FreeBSD kézikönyv

Kivonat

Üdvözöljük a FreeBSD világában! Ez a kézikönyv ismerteti a *FreeBSD 11.2-RELEASE*, ill. a *FreeBSD 12.0-RELEASE* telepítését és használatát a mindennapokban. A kézikönyv tartalmán számos független fejlesztõ *folyamatosan dolgozik*. Emiatt elképzelhetõ, hogy bizonyos fejezetek már elavultak és aktualizálásra szorulnak. Amennyiben úgy érezzük, hogy segíteni tudnánk a projekt munkájában, értesítsük a fejlesztõket a FreeBSD Dokumentációs Projekt levelezési lista címén! Ezen dokumentum legfrissebb változata mindig elérhetõ a FreeBSD honlapjáról (a korábbi változatok pedig megtalálhatóak a http://docs.FreeBSD.org/doc/ címen). Ezenkívül még rengeteg más formátumban és tömörítve is letölthetõ a FreeBSD FTP szerverérõl vagy a tükröoldalak egyikérõl. Amennyiben a kézikönyv nyomtatott változatára lenne szükségünk, megvásárolhatjuk a FreeBSD Mall-ból. Ha pedig keresni szeretnénk benne, azt a funkciót itt érhetjük el.

Fordította: Páli Gábor, utolsó ellenõrzés: 2010.11.28.

Tartalomjegyzék

Előszó . . . 11
Kiknek szánjuk ezt a könyvet . . . 11
Változtatások a harmadik kiadás óta . . . 11
Változtatások a második kiadás (2004) óta . . . 11
Változtatások az elsõ kiadás (2001) óta . . . 12
A könyv felépítése . . . 13
A könyvben alkalmazott konvenciók . . . 17
Köszönetnyilvánítás . . . 18
I: Bevezetés . . . 19
1. Bemutatkozás . . . 20
1.1. Áttekintés . . . 20
1.2. Üdvözöljük a FreeBSD-ben! . . . 20
1.3. A FreeBSD Projektrõl . . . 23
2. A FreeBSD telepítése . . . 29
2.1. Áttekintés . . . 29
2.2. Hardverkövetelmények . . . 29
2.3. A telepítés elõtt elvégzendõ feladatok . . . 30
2.4. A telepítés megkezdése . . . 38
2.5. A sysinstall bemutatása . . . 45
2.6. Lemezterület lefoglalása . . . 51
2.7. A telepítendõ összetevõk kiválasztása . . . 66
2.8. A telepítés eszközének kiválasztása . . . 69
2.9. A telepítés véglegesítése . . . 71
2.10. A telepítés után . . . 73
2.11. Hibakeresés . . . 114
2.12. Telepítési útmutató haladóknak . . . 118
2.13. Saját telepítõeszköz elkészítése . . . 120
3. A UNIX alapjai . . . 127
3.1. Áttekintés . . . 127
3.2. Virtuális konzolok és terminálok . . . 127
3.3. Engedélyek . . . 131
3.4. A könyvtárak elrendezése . . . 136
3.5. A lemezek szervezése . . . 139
3.6. Állományrendszerek csatlakoztatása és leválasztása . . . 146
3.7. Folyamatok . . . 149
3.8. Démonok, jelzések és a futó programok leállítása . . . 151
3.9. Parancsértelmezõk . . . 153
3.10. Szövegszerkesztõk . . . 156

3.11. Eszközök és eszközleírók . . . 156
3.12. Bináris formátumok . . . 157
3.13. Bõvebben olvashatunk... . . . 159
4. Alkalmazások telepítése: csomagok és portok . . . 162
4.1. Áttekintés . . . 162
4.2. Az alkalmazások telepítésének összefoglalása . . . 162
4.3. A számunkra szükséges alkalmazások felkutatása . . . 164
4.4. A csomagrendszer használata . . . 166
4.5. A Portgyûjtemény használata . . . 169
4.6. Telepítés utáni teendõk . . . 180
4.7. Teendõ a sérült portokkal . . . 181
5. Az X Window System . . . 182
5.1. Áttekintés . . . 182
5.2. Az X áttekintése . . . 182
5.3. Az X11 telepítése . . . 185
5.4. Az X11 beállítása . . . 186
5.5. Betûtípusok használata az X11-ben . . . 193
5.6. Az X bejelentkeztetõ képernyõje . . . 197
5.7. Munkakörnyezetek . . . 200
II: Gyakori feladatok . . . 205
6. II. Rész Gyakori feladatok . . . 206
6.1. Áttekintés . . . 206
6.2. Böngészõk . . . 206
6.3. Irodai eszközök . . . 211
6.4. Dokumentum-megjelenítõk . . . 214
6.5. Pénzügyi szoftverek . . . 216
6.6. Összefoglalás . . . 218
7. Multimédia . . . 219
7.1. Áttekintés . . . 219
7.2. A hangkártya beállítása . . . 220
7.3. MP3 . . . 224
7.4. Videók lejátszása . . . 227
7.5. TV kártyák beállítása . . . 236
7.6. Lapolvasók . . . 238
8. A FreeBSD rendszermag testreszabása . . . 244
8.1. Áttekintés . . . 244
8.2. Miért készítsünk saját rendszermagot? . . . 244
8.3. A rendszerünkben levõ hardverek összeszedése . . . 245
8.4. Meghajtók, alrendszerek és modulok . . . 246
8.5. Saját rendszermag készítése és telepítése . . . 247
8.6. A konfigurációs állomány . . . 250

8.7. Ha valamilyen hiba történne . . . 267
9. Nyomtatás . . . 269
9.1. Áttekintés . . . 269
9.2. Bevezetés . . . 269
9.3. Kezdeti beállítások . . . 270
9.4. Magasszintû nyomtatóbeállítás . . . 285
9.5. A nyomtatók használata . . . 318
9.6. Más nyomtatási rendszerek . . . 327
9.7. Hibakeresés . . . 327
10. Bináris Linux kompatibilitás . . . 332
10.1. Áttekintés . . . 332
10.2. Telepítés . . . 332
10.3. A Mathematica® telepítése . . . 336
10.4. A Maple™ telepítése . . . 338
10.5. A MATLAB® telepítése . . . 341
10.6. Az Oracle® telepítése . . . 344
10.7. Az SAP® R/3® telepítése . . . 347
10.8. Témák haladóknak . . . 370
III: Rendszeradminisztráció . . . 373
11. Beállítás és finomhangolás . . . 374
11.1. Áttekintés . . . 374
11.2. Kezdeti beállítások . . . 374
11.3. A mag beállítása . . . 376
11.4. Az alkalmazások beállítása . . . 377
11.5. Szolgáltatások indítása . . . 377
11.6. A `cron` segédprogram beállítása . . . 379
11.7. Az rc használata FreeBSD alatt . . . 381
11.8. A hálózati kártyák beállítása . . . 383
11.9. Virtuális címek . . . 390
11.10. Konfigurációs állományok . . . 391
11.11. Finomhangolás a sysctl használatával . . . 396
11.12. A lemezek finomhangolása . . . 397
11.13. A rendszermag korlátainak finomhangolása . . . 401
11.14. A lapozóterület bõvítése . . . 405
11.15. Energia- és erõforrásgazdálkodás . . . 406
11.16. A FreeBSD ACPI támogatásának használata és nyomonkövetése . . . 407
12. A FreeBSD rendszerindítási folyamata . . . 416
12.1. Áttekintés . . . 416
12.2. A rendszerindítás problémája . . . 416
12.3. A boot manager és az indulás fokozatai . . . 417
12.4. Kapcsolat a rendszermaggal a rendszerindítás folyamán . . . 424

12.5. Eszköz útmutatók (device.hints) . . . 425
12.6. Init: A folyamatirányítás elindítása . . . 426
12.7. A leállítási folyamat . . . 427
13. Felhasználók és hozzáférések alapvető kezelése . . . 428
13.1. Áttekintés . . . 428
13.2. Bevezetés . . . 428
13.3. Az adminisztrátori hozzáférés . . . 430
13.4. Rendszerhozzáférések . . . 430
13.5. Felhasználói hozzáférések . . . 430
13.6. A hozzáférések módosítása . . . 431
13.7. A felhasználók korlátozása . . . 435
13.8. Csoportok . . . 438
14. Biztonság . . . 440
14.1. Áttekintés . . . 440
14.2. Bevezetés . . . 440
14.3. A FreeBSD védelme . . . 442
14.4. DES, Blowfish, MD5 és a Crypt . . . 451
14.5. Egyszeri jelszavak . . . 452
14.6. A TCP kapcsolatok burkolása . . . 456
14.7. KerberosIV . . . 459
14.8. Kerberos5 . . . 468
14.9. OpenSSL . . . 477
14.10. VPN IPsec felett . . . 480
14.11. OpenSSH . . . 487
14.12. Az állományrendszerek hozzáféréseit vezérlõ listák . . . 493
14.13. A külsõ programok biztonsági problémáinak figyelése . . . 495
14.14. A FreeBSD biztonsági figyelmeztetései . . . 497
14.15. A futó programok nyilvántartása . . . 499
15. A jail alrendszer . . . 501
15.1. Áttekintés . . . 501
15.2. A jail alrendszerhez kapcsolódó fogalmak . . . 501
15.3. Bevezetés . . . 502
15.4. A jailek létrehozása és vezérlése . . . 503
15.5. Finomhangolás és karbantartás . . . 505
15.6. A jailek alkalmazása . . . 506
16. Kötelező hozzáférés-vezérlés (MAC) . . . 514
16.1. Áttekintés . . . 514
16.2. A fejezet fontosabb fogalmai . . . 515
16.3. A MAC ismertetése . . . 516
16.4. Bõvebben a MAC címkéirõl . . . 518
16.5. A védelem megtervezése . . . 523

16.6. A modulok beállítása . . . 524
16.7. A seeotheruids MAC-modul . . . 524
16.8. A bsdextended MAC-modul . . . 525
16.9. Az ifoff MAC-modul . . . 526
16.10. A portacl MAC-modul . . . 526
16.11. A partition MAC-modul . . . 528
16.12. A többszintû biztonsági MAC-modul . . . 529
16.13. A Biba MAC-modul . . . 531
16.14. A LOMAC MAC-modul . . . 533
16.15. A Nagios elzárása a MAC rendszerrel . . . 534
16.16. A felhasználók korlátozása . . . 538
16.17. A hibák elhárítása a MAC rendszerben . . . 539
17. Biztonsági események vizsgálata . . . 541
17.1. Áttekintés . . . 541
17.2. A fejezet fontosabb fogalmai . . . 542
17.3. A vizsgálat támogatásának telepítése . . . 542
17.4. A vizsgálat beállítása . . . 543
17.5. A vizsgálati alrendszer használata . . . 546
18. Háttértárak . . . 550
18.1. Áttekintés . . . 550
18.2. Az eszközök elnevezései . . . 550
18.3. Lemezek hozzáadása . . . 551
18.4. RAID . . . 554
18.5. USB tárolóeszközök . . . 559
18.6. Lézeres tárolóeszközök (CD-k) létrehozása és használata . . . 561
18.7. Lézeres tárolóeszközök (DVD-k) létrehozása és használata . . . 569
18.8. Hajlékonylemezek létrehozása és használata . . . 575
18.9. Szalagok létrehozása és használata . . . 576
18.10. Biztonsági mentés hajlékonylemezekre . . . 579
18.11. Mentési stratégiák . . . 581
18.12. Alapvetõ tudnivalók a biztonsági mentésrõl . . . 582
18.13. Hálózat, memória és állomány alapú állományrendszerek . . . 586
18.14. Az állományrendszerek pillanatképei . . . 589
18.15. Az állományrendszerek kvótái . . . 591
18.16. A lemezpartíciók titkosítása . . . 594
18.17. A lapozóterület titkosítása . . . 602
19. GEOM: A moduláris lemezszervező rendszer . . . 605
19.1. Áttekintés . . . 605
19.2. A GEOM bemutatása . . . 605
19.3. RAID0 - Csíkozás . . . 605
19.4. RAID1 - Tükrözés . . . 607

19.5. Eszközök hálózati illesztése a GEOM-ban . . . 611
19.6. A lemezes eszközök címkézése . . . 611
19.7. Naplózó UFS GEOM-on keresztül . . . 615
20. Támogatott állományrendszerek . . . 617
20.1. Áttekintés . . . 617
20.2. A Z állományrendszer (ZFS) . . . 617
21. A Vinum kötetkezelő . . . 627
21.1. Áttekintés . . . 627
21.2. Kicsik a lemezeink . . . 627
21.3. A hozzáférési idõk szûk keresztmetszetei . . . 627
21.4. Adatintegritás . . . 629
21.5. A Vinum objektumai . . . 630
21.6. Példák . . . 632
21.7. Az objektumok elnevezése . . . 638
21.8. A Vinum beállítása . . . 640
21.9. Rendszerindítás Vinum-kötetrõl . . . 642
22. Virtualizáció . . . 648
22.1. Áttekintés . . . 648
22.2. A FreeBSD mint vendég . . . 648
22.3. A FreeBSD mint gazda . . . 678
23. Honosítás - Az I18N/L10N használata és beállítása . . . 680
23.1. Áttekintés . . . 680
23.2. Az alapok . . . 680
23.3. A honosítás használata . . . 681
23.4. I18N programok fordítása . . . 688
23.5. A FreeBSD honosítása adott nyelvekre . . . 688
24. A FreeBSD frissítése és frissen tartása . . . 692
24.1. Áttekintés . . . 692
24.2. A FreeBSD frissítése . . . 692
24.3. A Portgyûjtemény frissítése a Portsnap használatával . . . 700
24.4. A dokumentáció frissítése . . . 701
24.5. A fejlesztõi ág követése . . . 707
24.6. A forrás szinkronizálása . . . 711
24.7. Az alaprendszer újrafordítása . . . 712
24.8. A források követése több géppel . . . 730
25. DTrace . . . 732
25.1. Áttekintés . . . 732
25.2. Eltérések az implementációban . . . 732
25.3. A DTrace támogatásának engedélyezése . . . 733
25.4. A DTrace használata . . . 734
25.5. A D nyelv . . . 737

IV: Hálózati kommunikáció . . . 738
26. Soros vonali kommunikáció . . . 739
26.1. Áttekintés . . . 739
26.2. Bevezetés . . . 739
26.3. Terminálok . . . 744
26.4. Betárcsázós szolgáltatások . . . 750
26.5. A betárcsázós szolgáltatások használata . . . 758
26.6. A soros vonali konzol beállítása . . . 762
27. A PPP és a SLIP . . . 772
27.1. Áttekintés . . . 772
27.2. A felhasználói PPP alkalmazása . . . 772
27.3. A rendszerszintû PPP alkalmazása . . . 786
27.4. PPP kapcsolatok hibaelhárítása . . . 794
27.5. A PPP használata Ethernet felett (PPPoE) . . . 798
27.6. PPP ATM felett (PPPoA) . . . 800
27.7. A SLIP használata . . . 804
28. Elektronikus levelezés . . . 814
28.1. Áttekintés . . . 814
28.2. Az elektronikus levelezés használata . . . 814
28.3. A sendmail beállítása . . . 817
28.4. A levéltovábbító ügynök megváltoztatása . . . 820
28.5. A hibák elhárítása . . . 822
28.6. Komolyabb témák . . . 826
28.7. SMTP és az UUCP . . . 829
28.8. Csak küldés beállítása . . . 831
28.9. Levelezés betárcsázós kapcsolattal . . . 832
28.10. Az SMTP hitelesítése . . . 833
28.11. Levelezõ kliensek . . . 835
28.12. A fetchmail használata . . . 843
28.13. A procmail használata . . . 844
29. Hálózati szerverek . . . 846
29.1. Áttekintés . . . 846
29.2. Az inetd"szuperszerver" . . . 846
29.3. A hálózati állományrendszer (NFS) . . . 851
29.4. Hálózati információs rendszer (NIS/YP) . . . 858
29.5. A hálózat automatikus beállítása (DHCP) . . . 878
29.6. Névfeloldás (DNS) . . . 882
29.7. Az Apache webszerver . . . 896
29.8. Állományok átvitele (FTP) . . . 902
29.9. Állomány- és nyomtatási szolgáltatások Microsoft® Windows® kliensek számára (Samba) . . . 904

29.10. Az órák egyeztetése az NTP használatával . . . 907
29.11. Távoli gépek naplózása syslogd használatával . . . 910
30. Tűzfalak . . . 915
30.1. Bevezetés . . . 915
30.2. Röviden a tûzfalakról . . . 915
30.3. Tûzfalak . . . 916
30.4. Az OpenBSD csomagszûrõje (PF) és az ALTQ . . . 916
30.5. Az IPFILTER (IPF) tûzfal . . . 920
30.6. IPFW . . . 942
31. Egyéb haladó hálózati témák . . . 963
31.1. Áttekintés . . . 963
31.2. Átjárók és az útválasztás . . . 963
31.3. Vezeték nélküli hálózatok . . . 971
31.4. Bluetooth . . . 993
31.5. Hálózati hidak . . . 1002
31.6. Linkek összefûzése és hibatûrése . . . 1009
31.7. Lemez nélküli mûködés . . . 1014
31.8. ISDN . . . 1021
31.9. Hálózati címfordítás . . . 1025
31.10. Párhuzamos vonali IP (PLIP) . . . 1030
31.11. Az IPv6 . . . 1032
31.12. Az Aszinkron adatátviteli mód (ATM) . . . 1037
31.13. A Közös cím redundancia protokoll (CARP) . . . 1039
V: Függelék . . . 1042
Függelék A: A FreeBSD beszerzése . . . 1043
A.1. CD és DVD kiadók . . . 1043
A.2. FTP oldalak . . . 1045
A.3. Anonim CVS . . . 1052
A.4. A CTM használata . . . 1055
A.5. A CVSup használata . . . 1059
A.6. CVS címkék . . . 1073
A.7. AFS oldalak . . . 1078
A.8. Rsync oldalak . . . 1079
Függelék B: Irodalomjegyzék . . . 1081
B.1. A FreeBSD-rõl szóló könyvek és folyóiratok . . . 1081
B.2. Felhasználói kézikönyvek . . . 1082
B.3. Rendszeradminisztrátori kézikönyvek . . . 1082
B.4. Programozói kézikönyvek . . . 1083
B.5. Az operációs rendszerek belsõ mûködésérõl . . . 1083
B.6. Biztonságról szóló írások . . . 1084
B.7. Hardverrel foglalkozó írások . . . 1084

B.8. UNIX® történelem . . . 1085
B.9. Magazinok és folyóiratok . . . 1085
Függelék C: Források az interneten . . . 1086
C.1. Levelezési listák . . . 1086
C.2. Usenet hírcsoportok . . . 1103
C.3. Világhálós szolgáltatások . . . 1105
C.4. E-mail címek . . . 1107
Függelék D: PGP-kulcsok . . . 1108
D.1. Tisztségviselők . . . 1108

Előszó

Kiknek szánjuk ezt a könyvet

A FreeBSD-t még nem ismerõk felfedezhetik, hogy a könyv elsõ része a FreeBSD telepítésének folyamatán vezeti keresztül a felhasználót, valamint érintõlegesen bemutatja az ezt alátámasztó UNIX®-os alapfogalmakat és szabályokat. Ennek a résznek a végigjárása nem kíván többet, csupán egy kis felfedezõ kedvet, illetve a menet közben bemutatott új fogalmak befogadását.

Ha túljutottunk rajta, a kézikönyv második, jóval terjedelmesebb része a FreeBSD-t használó rendszergazdák számára nyújt mindenféle témában minden részletre kiterjedõ referenciát. Ezek közül egyes fejezetek elvárnak némi elõzetes felkészülést, amelyet minden fejezet áttekintésében említeni is fogunk.

További információkért olvassuk át a Irodalomjegyzéket.

Változtatások a harmadik kiadás óta

A kézikönyv jelenleg interneten elérhetõ változata számtalan önkéntes által az utóbbi 10 évben végzett együttes erõfeszítéseinek eredményeit tükrözi. A 2004-ben két kötetben megjelentetett harmadik kiadás óta a következõ fontosabb változások történtek:

- DTrace: készült egy új fejezet a DTrace nevû teljesítmény-elemzõ eszközrõl.
- Támogatott állományrendszerek: ebben a fejezetben a FreeBSD és a különbözõ más rendszerekhez fejlesztett állományrendszerek viszonyát mutatjuk be, többek a Sun™ ZFS megoldását.
- Biztonsági események vizsgálata: ez a fejezet FreeBSD új biztonsági fejlesztéseit foglalja össze és mutatja be a használatukat.
- Virtualizáció: ebben az új fejezetben a FreeBSD rendszerhez és rajta elérhetõ különbözõ virtualizációs technológiákról szólunk.

Változtatások a második kiadás (2004) óta

A harmadik kiadás a FreeBSD Dokumentációs Projekt tagjainak két évi kemény munkájának gyümölcse. A nyomtatott változat már olyan nagyra nõtt, hogy két külön kötetben kellett kiadnunk. Az alábbi fontosabb változtatások jelentek meg ebben az új kiadásban:

- Beállítás és finomhangolás: a beállításra és finomhangolásra vonatkozó részeket bõvítettük az ACPI energia- és erõforrás gazdálkodásról szóló részekkel, a `cron` rendszerprogrammal, illetve még több, a rendszermag finomhangolását elõsegítõ opció leírásával.
- Biztonság: a biztonságról szóló részt bõvítettük a virtuális magánhálózatokról (VPN-ekrõl), állományrendszeri hozzáférés-vezérlési listákról (ACL-ek) szóló elemekkel, valamint biztonságtechnikai tanácsokkal.
- Kötelező hozzáférés-vezérlés (MAC): a kötelezõ hozzáférés-vezérlésrõl (MAC-rõl) szóló fejezet teljesen új ebben a kiadásban. Bemutatja, mi is az a MAC és hogyan hasznosítható egy FreeBSD-s

rendszer biztonságossá tételében.

- Háttértárak: a háttértárakat tartalmazó részt bõvítettük az USB-tárakról, állományrendszeri pillanatképeirõl, lemezkvótákról, állomány- és hálózat alapú állományrendszerekrõl, továbbá a titkosított partíciókról szóló részekkel.
- A Vinum kötetkezelő: a Vinum egy új fejezet ebben a kiadásban. Bemutatja a Vinum logikaikötet-kezelõ használatát, aminek segítségével eszközfüggetlen módon hozhatunk létre logikai lemezeket, szoftveres RAID-0, RAID-1 és RAID-5 konfigurációkat.
- Bekerült egy hibaelhárításról szóló rész a A PPP és a SLIP PPP és SLIP leírásához.
- Elektronikus levelezés: az elektronikus levelezést ismertetõ részt bõvítettük a különféle levéltovábbító rendszerekrõl, az SMTP hitelesítésrõl, UUCP protokollról, a fetchmail és procmail programokról szóló elemekkel, valamint egyéb, haladókat megcélzó témákkal.
- Hálózati szerverek: a hálózati szervereket ismertetõ rész egy teljesen új fejezet ebben a kiadásban. Benne megtalálható az Apache HTTP szerver, az ftpd szerver, illetve a Microsoft® Windows®-os kliensek számára megfelelõ Samba szerver beállítása. Az érthetõség kedvéért egyes részek átkerültek ide a Egyéb haladó hálózati témák, vagyis a haladó hálózati témákat tárgyaló fejezetbõl.
- Egyéb haladó hálózati témák: a haladó hálózati témákat tartalmazó részt kiegészítettük a FreeBSD és a Bluetooth® eszközök kapcsolatáról, a vezeték nélküli hálózatokról és az aszinkron adatátvitel módról (ATM-rõl) szóló ismeretekkel.
- Létrehoztunk egy szójegyzéket abból a célból, hogy a könyvben használt definíciók és szakkifejezések egyetlen központi helyen össze legyenek foglalva.
- Számos esztétikai javítást eszközöltünk a könyvben található ábrákon és táblázatokon.

Változtatások az elsõ kiadás (2001) óta

A második kiadás a FreeBSD Dokumentációs Projekt tagjainak két évi komoly munkájának eredménye. Az alábbi fontosabb változtatások jelennek meg ebben a kiadásban:

- Bekerült egy teljes tárgy- és névmutató.
- Mindegyik ASCII-ábrát grafikusak váltották fel.
- Mindegyik fejezet elejére odakerült egy általános áttekintés, ami egy rövid összefoglalást ad a fejezet tartalmáról, valamint közli az elolvasásához szükséges ismereteket.
- A tartalmat felosztottuk logikailag három részre: "Bevezetés", "Rendszeradminisztráció" és "Függelék".
- A A FreeBSD telepítéseet ("A FreeBSD telepítése") teljesen újraírtuk és sok-sok illusztráció is hozzáadásra került a könnyebb megértés érdekében.
- A A UNIX alapjai ("A UNIX® alapjai") kiegészült a futó programokról, démonokról és jelzésekrõl szóló további hasznos információkkal.
- A Alkalmazások telepítése. csomagok és portoket ("Alkalmazások telepítése") bõvítettük a bináris csomagkezelésrõl szóló további ismeretekkel.
- A Az X Window System ("Az X Window System") teljes újraíráson ment át, aminek folyamán igyekeztünk nagyobb hangsúlyt helyezni a modern asztali technológiák, mint pl. a KDE és

GNOME XFree86™ 4.X-en történõ használatának leírására.

- A A FreeBSD rendszerindítási folyamata ("A FreeBSD rendszerindítási folyamata") kibõvült.
- A Háttértárak ("Háttértárak") két, korábban külön levõ fejezet, a "Lemezek" és "Biztonsági mentések" összeolvasztásából jött létre. Úgy éreztük, a bennük helyet kapott témákat sokkal könnyebb úgy megérteni, ha egyetlen fejezetben tárgyaljuk ezeket. Egy (hardveres és szoftveres) RAID-rõl szóló rész is belekerült.
- A Soros vonali kommunikáció ("Soros vonali kommunikáció") teljes átszervezésre került, valamint a FreeBSD 4.X/5.X verziókhoz igazítottuk.
- A A PPP és a SLIP ("A PPP és a SLIP") lényegesen sokat fejlõdött.
- Számos új rész került a Egyéb haladó hálózati témákbe ("Egyéb haladó hálózati témák").
- A Elektronikus levelezés ("Elektronikus levelezés") kibõvült a sendmail beállításáról tartalmazó újabb információkkal.
- A Bináris Linux kompatibilitás ("Bináris Linux kompatibilitás") kiegészült az Oracle® és a SAP® R/3® telepítését bemutató részekkel.
- Az alábbi új témák kerültek tárgyalásra a második kiadásban:
  - Beállítás és finomhangolás (Beállítás és finomhangolás)
  - Multimédia (Multimédia)

A könyv felépítése

A könyvet négy logikailag elkülönülõ részre osztottuk fel. Az elsõ, *Bevezetés* címû részben bemutatjuk a FreeBSD telepítését és használatának alapjait. Elgondolásunk szerint az itt szereplõ fejezeteket sorban érdemes elolvasni, esetenként kihagyni azokat, amelyek már az olvasó számára ismert témákat dolgoznak fel. A második, *Gyakori feladatok* címû részben megismerhetjük a FreeBSD néhány gyakorta használt lehetõségét. Ez a rész, valamint az ezt követõ összes többi tetszõleges sorrendben olvasható. Mindegyik fejezet egy rövidke összefoglalással kezdõdik, amely ismerteti, az olvasótól milyen jellegû tapasztalatokat vár el a fejezet megértése. Célja, hogy segítsen az olvasónak megtalálni a számára érdekes témákat. A harmadik, *Rendszeradminisztráció* címû részben rendszergazdai feladatokat tárgyalunk. A negyedik, *Hálózati kommunikáció* címû részben hálózatok és szerverek üzemeltetésével kapcsolatos ismereteket foglaltunk össze. Végül, az ötödik rész tartalmazza a függeléket és az irodalomjegyzéket, hivatkozásokat.

***Bemutatkozás*: Bemutatkozás**

A FreeBSD bemutatkozik az új felhasználóknak. Szó esik a FreeBSD Projekt történetérõl, célkitûzéseirõl és a fejlesztési modelljérõl.

***A FreeBSD telepítése*: A FreeBSD telepítése**

Végigvezetjük a felhasználót a telepítési folyamat egészén. Bizonyos rendhagyó kérdések, mint például a soros konzolon keresztül történõ telepítés is terítékre kerülnek.

***A UNIX alapjai*: A UNIX® alapjai**

Sorra vesszük a FreeBSD operációs rendszer alapvetõ parancsait és lehetõségeit. Amennyiben már jártasak vagyunk valamilyen szinten a Linux® vagy más UNIX®-típusú rendszerek használatában, nyugodtan kihagyhatjuk ezt a fejezetet.

Alkalmazások telepítése. csomagok és portok: **Alkalmazások telepítése, csomagok és portok**

Megismerhetjük, miként tudunk külsõ cégek által fejlesztett alkalmazásokat telepíteni a FreeBSD "Portgyûjteményének" (FreeBSD Ports Collection) vagy a megszokott bináris csomagok használatán keresztül.

Az X Window System: **Az X Window System**

Általános bemutatásra kerül az X Window System, valamint az X11 használata a FreeBSD-n. Ezenkívül olvashatunk az elterjedtebb munkakörnyezetekrõl, mint pl. a KDE és a GNOME.

Asztali alkalmazások: **Asztali alkalmazások**

Felsoroljuk az ismertebb asztali alkalmazásokat: webböngészõket és alkalmazói programcsomagokat, és bemutatjuk, hogyan telepítsük ezeket FreeBSD-re.

Multimédia: **Multimédia**

Megtudhatjuk, hogyan állítsuk be a zene- és videolejátszást rendszerünkön. Emellett olvashatunk néhány multimédiás alkalmazás használatáról is.

A FreeBSD rendszermag testreszabása: **A FreeBSD rendszermag testreszabása**

Kifejtjük, miért lehet szükségünk egy új rendszermag konfigurálására, és részletesen végigjárjuk egy rendszermag konfigurációjának, fordításának és telepítésének lépéseit.

Nyomtatás: **Nyomtatás**

Ismertetjük, hogyan lehet nyomtatókat használni FreeBSD alatt, beleértve a munkalapok készítésének mikéntjét, a nyomtatóhasználat nyilvántartását és a kezdeti beállításokat.

Bináris Linux kompatibilitás: **Bináris Linux kompatibilitás**

Megismerhetjük a FreeBSD bináris Linux kompabilitásához kapcsolódó lehetõségeket. Ezenfelül részletekre is kitérõ telepítési útmutatót találhatunk különbözõ népszerû linuxos alkalmazásokhoz, mint például az Oracle®, SAP® R/3® és a Mathematica®.

Beállítás és finomhangolás: **Beállítás és finomhangolás**

Megismerhetjük a FreeBSD azon paramétereit, amelyek megfelelõ állításával a rendszergazdák a lehetõ legtöbbet képesek kihozni FreeBSD rendszerükbõl. Ezenkívül bemutatásra kerül a FreeBSD-ben használt számos konfigurációs állomány, valamint hogy ezeket hol találhatjuk meg.

A FreeBSD rendszerindítási folyamata: **A FreeBSD rendszerindítási folyamata**

Tartalmazza a FreeBSD rendszerindítási folyamatának leírását, és elmagyarázza, miként lehet ezt vezérelni a konfigurációs beállítások segítségével.

Felhasználók és hozzáférések alapvető kezelése: **Felhasználók és hozzáférések alapvetõ kezelése**

Bemutatja a felhasználói fiókok létrehozását és kezelését. Emellett megemlíti a felhasználókra érvényesíthetõ erõforrás-megszorításokat, illetve egyéb fiókkezelési feladatokat.

Biztonság: **Biztonság**

Bemutatásra kerül a FreeBSD rendszerünk biztonságossá tételére alkalmas számos különféle eszköz, többek közt a Kerberos, IPsec és az OpenSSH.

A jail alrendszer: **A jail alrendszer_**

Megtudhatjuk, hogyan mûködik az alkalmazások elszigeteléséért felelõs jail alrendszer, valamint miben emelkedik ki a FreeBSD-ben is megtalálható hagyományos "chroot" megoldással szemben.

Kötelező hozzáférés-vezérlés (MAC): ***Kötelezõ hozzáférés-vezérlés***

Megismerhetjük a kötelezõ hozzáférés-vezérlést (MAC-et), valamint azt, hogyan is tudjuk felhasználni egy FreeBSD-s rendszer biztonsága érdekében.

Biztonsági események vizsgálata: ***Biztonsági események vizsgálata***

Kiderül, mit jelent a FreeBSD-ben az események vizsgálata, illetve mindez hogyan telepíthetõ, konfigurálható és miként tudjuk a vizsgálatok adatait kielemezni vagy felügyelni.

Háttértárak: ***Háttértárak***

Bemutatásra kerül, miként kezelhetjük a háttértárolókat és állományrendszereket a FreeBSD-ben. Ide tartoznak a fizikai lemezek, RAID-tömbök, optikai és szalagos egységek, memória alapú lemezek és a hálózati állományrendszerek.

GEOM. a moduláris lemezszervező rendszer: ***GEOM, a moduláris lemezszervezõ rendszer***

Megismerhetjük a FreeBSD-ben jelenlevõ GEOM alrendszert és az általa támogatott különbözõ RAID-szintek beállítását.

Támogatott állományrendszerek: ***Támogatott állományrendszerek***

A FreeBSD operációs rendszer számára nem natív állományrendszerekkel foglalkozik, például a Sun™ Z állományrendszerével.

A Vinum kötetkezelő: ***A Vinum kötetkezelõ***

Megtudhatjuk, hogyan használjuk a Vinumot, a logikaikötet-kezelõt, amely eszközfüggetlen logikai lemezeket, szoftveres RAID-0, RAID-1 és RAID-5 konfigurációkat biztosít.

Virtualizáció: ***Virtualizáció***

Tartalmazza a virtualizációs rendszerek által felkínált lehetõségek bemutatását és használatát a FreeBSD-vel.

Honosítás. Az I18N/L10N használata és beállítása: ***Honosítás, az I18N/L10N használata és beállítása***

Bemutatja, hogyan használjuk a FreeBSD-t a rendszer és az alkalmazások szintjén az angoltól eltérõ nyelveken.

A FreeBSD frissítése és frissen tartása: ***A FreeBSD frissítése és frissen tartása***

Elmagyarázza, mik az alapvetõ különbségek a FreeBSD-STABLE, FreeBSD-CURRENT verziók, valamint a FreeBSD kiadások között. Bemutatja, mely felhasználók lehetnek azok, akik a legtöbbet tudnak profitálni egy fejlesztõi rendszer használatából, illetve körvonalazza ennek folyamatát. Továbbá röviden összefoglalja azokat az eszközöket, amelyekkel a felhasználók frissíthetik a rendszerüket a biztonsági és kritikus hibák javításakor.

DTrace: ***DTrace***

A Sun™ DTrace eszközének beállítását és használatát mutatja be. A segítségével megvalósított

dinamikus nyomkövetéssel lehetõségünk nyílik valós idejû elemzéseken keresztül felderíteni a különbözõ teljesítménybeli problémákat.

Soros vonali kommunikáció: ***Soros vonali kommunikáció***

Kifejti, hogyan csatlakoztassunk terminált vagy modemet a FreeBSD rendszerünkhöz, ha behívó vagy betárcsázós kapcsolatot szeretnénk létrehozni.

A PPP és a SLIP: ***A PPP és a SLIP***

Bemutatja, miként tudjuk PPP-n, SLIP-en és Etherneten keresztüli PPP-vel (PPPoE) összekapcsolni a FreeBSD-t távoli rendszerekkel.

Elektronikus levelezés: ***Elektronikus levelezés***

Megismerhetjük egy elektronikus levelezõ szerver különféle komponenseit, és elmélyedhetünk az egyik leghíresebb levelezõszerver-szoftver, a sendmail használatában és felületesebb konfigurálásában.

Hálózati szerverek: ***Hálózati szerverek***

Részletekbe menõen és konfigurációs példákkal mutatja be, miként tudunk hálózati állományrendszer kiszolgálónak, névszervernek, hálózati információs rendszer kiszolgálónak vagy idõszinkronizációs szervernek beállítani egy FreeBSD-s számítógépet.

Tűzfalak: ***Tûzfalak***

Kifejti a szoftveres tûzfalak mögött álló filozófiát, valamint részletesen tárgyalja a különbözõ, FreeBSD-n elérhetõ tûzfalak konfigurációját.

Egyéb haladó hálózati témák: ***Egyéb haladó hálózati témák***

Feldolgoz számos hálózati témát, beleértve az internet kapcsolat helyi hálózaton (LAN-on) keresztül történõ megosztását több számítógép között, haladó forgalomirányítási kérdéseket, vezeték nélküli hálózatok beállítását, Bluetooth®, ATM, IPv6 és sok minden mással kapcsolatos információkat.

A FreeBSD beszerzése: ***A FreeBSD beszerzése***

Felsorolja azokat a forrásokat, ahonnan a FreeBSD CD-n vagy DVD-n beszerezhetõ, valamint azokat a honlapokat, ahonnan letölthetõ vagy telepíthetõ a FreeBSD.

Irodalomjegyzék: ***Irodalomjegyzék***

A könyv sok tekintetben olyan témákat is érint, amelyek felkelthetik az olvasó érdeklõdését és ezek kapcsán bõvebb magyarázatra vágyik. Az irodalomjegyzékben ezért összeírtunk számos remek könyvet, amelyekre hivatkozunk is a fejezetekben.

Források az interneten: ***Erõforrások az interneten***

Tartalmazza a FreeBSD felhasználók számára elérhetõ azon fórumokat, ahová beküldhetik kérdéseiket, illetve szakmai jellegû társalgásokat folytathatnak.

PGP-kulcsok: ***PGP-kulcsok***

Az egyes FreeBSD fejlesztõk PGP-kulcsait sorolja fel.

A könyvben alkalmazott konvenciók

A könnyebb és egységesebb olvashatóság kedvéért az alábbi konvenciókat igyekeztünk követni a könyvben.

Tipográfiai konvenciók

Dõlt

A *dõlt* betûket állománynevek, URL-ek, kiemelt szövegek és a szakmai kifejezések elsõ elõfordulásakor használjuk.

`Írógépszerû`

Az `írógépszerû` betûket hibaüzenetek, parancsok, környezeti változók, portok, számítógépek, felhasználók, csoportok, eszközök nevei, változók és kódrészletek esetén használjuk.

Félkövér

A félkövér betûket alkalmazások, parancsok és billentyûk megnevezésénél használjuk.

Felhasználói bevitel

A billentyûket **félkövérrel** írjuk, hogy kiemelkedjenek a szöveg többi részébõl. Az egyszerre megnyomni kívánt billentyûk kombinációját a + jelöléssel adjuk meg, mint például:

`Ctrl` + `Alt` + `Del`

Ez azt jelenti, hogy a felhasználónak a `Ctrl`, `Alt` és `Del` billentyûket egyszerre kell lenyomnia.

Azokat a billentyûket, amelyeket egymás után kell lenyomni, vesszõvel választjuk el, például:

`Ctrl` + `X`, `Ctrl` + `S`

Ez tehát azt jelenti, hogy a felhasználónak elõször a `Ctrl` és `X` billentyûket, majd a `Ctrl` és `S` billentyûket kell egyszerre lenyomnia.

Példák

A E:\> kijelzéssel kezdõdõ példák egy MS-DOS® parancsot jelölnek. Ha másképpen nem említjük, ezeket a parancsokat a modern Microsoft® Windows®-okban található "Parancssorból" kell kiadni.

```
E:\> tools\fdimage floppies\kern.flp A:
```

A # kijelzéssel kezdõdõ példák a FreeBSD-ben rendszeradminisztrátori jogokat igénylõ parancsok kiadását jelentik. Ehhez bejelentkezhetünk a `root` felhasználóval, vagy felvethetjük a rendszeradminisztrátori jogokat a saját felhasználói fiókunkból a su(1) használatával is.

```
# dd if=kern.flp of=/dev/fd0
```

A % kijelzéssel kezdõdõ példák olyan parancsra utalnak, amelyeket egy normál felhasználói fiókból érdemes kiadni. Hacsak másképpen nem jelezzük, a C-shell szintaxisát használjuk a környezeti változók és egyéb parancsok megadásakor.

```
% top
```

Köszönetnyilvánítás

A könyv, amit itt most olvashatunk, több száz ember együttes munkájának eredménye a világ minden tájáról. Akár csak elgépeléseket javítottak, vagy komplett fejezeket adtak hozzá, minden hozzájárulás hasznosnak bizonyult.

Emellett sok cég anyagilag is támogatta a könyv fejlõdését, lehetõvé téve ezáltal, hogy a szerzõk teljes munkaidõben dolgozhassanak rajta, pénzt kapjanak az írásaikért stb. Leginkább a BSDi (amelyet késõbb felvásárolt a Wind River Systems) adott teljes munkaidõs fizetést a FreeBSD Dokumentációs Projekt tagjainak a könyv gondozásához, amely végül az elsõ nyomtatott kiadás megjelentetéséhez vezetett 2000 márciusában (ISBN 1-57176-241-8). A Wind River Systems ezt követõen további szerzõket is finanszírozott a nyomtatási-szedési infrastruktúra továbbfejlesztéséhez és a könyv tartalmának bõvítéséhez. Ennek eredménye lett a második nyomtatott kiadás, amely 2001 novemberében jelent meg (ISBN 1-57176-303-1). 2003 - 2004 folyamán a FreeBSD Mall, Inc. támogatott anyagilag számos hozzájárulót a kézikönyvet illetõ munkájáért, a harmadik nyomtatott kiadásra történõ elõkészítésben.

Part I: Bevezetés

A FreeBSD kézikönyv ezen része azoknak a felhasználóknak és rendszergazdáknak szól, akik még nem ismerik a FreeBSD-t. A fejezetek:

- Bemutatják a FreeBSD-t.
- Végigvezetnek a telepítés folyamatán.
- Ismertetik a UNIX® alapjait.
- Megmutatják, hogyan telepítsük a FreeBSD-hez elérhetõ megannyi külsõ alkalmazást.
- Megismerhetjük az X-et, a UNIX®-os ablakozórendszert, és részleteiben is láthatjuk, miként konfiguráljunk be egy munkakörnyezetet, amellyel kényelmesebbé válik a munka.

A fejezetek megírása során arra törekedtünk, hogy minél kevesebb hivatkozást tegyünk a könyv késõbb következõ részeire, így ennek köszönhetõen a kézikönyv ezen része anélkül olvasható, hogy közben folyamatosan elõre-hátra kellene lapozgatnunk benne.

Chapter 1. Bemutatkozás

1.1. Áttekintés

Köszönjük, hogy érdeklõdik a FreeBSD iránt! A fejezet a FreeBSD Projektet több különbözõ vonatkozásban mutatja be: a történetét, a céljait, a fejlesztési modelljét és így tovább.

A fejezet elolvasása során megismerjük:

- hogyan viszonyul a FreeBSD más operációs rendszerekhez;
- a FreeBSD Projekt történetét;
- a FreeBSD Projekt célkitûzéseit;
- a FreeBSD nyílt forráskódú fejlesztési modelljének alapjait;
- és természetesen: hogyan is keletkezett a "FreeBSD" név.

1.2. Üdvözöljük a FreeBSD-ben!

A FreeBSD egy 4.4BSD-Lite alapú operációs rendszer Intel® (x86 és Itanium®), AMD64, Alpha™, Sun UltraSPARC® számítógépekre. Jelenleg is portolás alatt áll további architektúrákra. Olvashatunk a FreeBSD történetérõl vagy éppen az aktuális kiadásáról. Ha szeretnénk hozzájárulni a Projekt fejlõdéséhez (forráskód, hardver vagy pénz), olvassuk el a Hozzájárulás a FreeBSD-hez címû cikket (angolul).

1.2.1. Mire képes a FreeBSD?

A FreeBSD számos figyelemre méltó tulajdonságot tudhat magáénak. Ezek közül néhány:

- A *preemptív ütemezés* dinamikusan szabályozható prioritások segítségével biztosítja a számítógép felhasználók és alkalmazások közti finom és igazságos megosztását, akár a legnagyobb terhelés esetén is.
- *Többfelhasználós rendszerként* lehetõvé teszi, hogy sokan tudják a FreeBSD-t egyszerre többféle dologra is használni. Például, ez azt jelenti, hogy a rendszerhez csatlakoztatott különbözõ perifériák, mint például a nyomtatók és szalagos egységek, megfelelõen szétoszthatóak a felhasználók között vagy éppen a hálózaton, és az egyes erõforrásokhoz a felhasználók vagy azok egy csoportja csak korlátozott módon férhetnek hozzájuk, elkerülve ezzel a rendszer számára létfontosságú erõforrások túlterhelését.
- A *TCP/IP hálózati protokoll* gyors és megbízható implementációja, illetve a legfontosabb ipari szabványok, mint az SCTP, DHCP, NFS, NIS, PPP, SLIP, IPsec és IPv6 támogatása. Ezáltal egy FreeBSD-s számítógép könnyedén képes együttmûködni más rendszerekkel vagy akár vállalati szerverként is üzemelni. Megbirkózik az NFS (Network File System, távoli állományelérés) és az elektronikus levelezés megszervezésével ugyanúgy, ahogy a vállalatunk internetes elvárásaival a WWW, FTP és forgalomirányítási protokollokon keresztül és tûzfal iránti (biztonsági) igényeivel is.
- A *memóriavédelem* megvalósítása gondoskodik róla, hogy az alkalmazások (vagy a felhasználók)

ne zavarják egymást. Az egyik alkalmazás összeomlása nincs kihatással a rendszerben futó összes többire.

- A FreeBSD egy *32 bites* operációs rendszer (az Alpha, Itanium®, AMD64 és UltraSPARC® architektúrákon pedig *64 bites*), amelyet már a kezdetektõl fogva annak terveztek.
- A *X Window System* ipari szabványa (X11R7) alapján szolgáltatja a grafikus felhasználói felületet (GUI) bármelyik VGA-kártyán és monitoron, illetve annak teljes forráskódja is elérhetõ.
- *Bináris szintû kompatibilitás* a Linuxra, SCO-ra, SVR4-re, BSDI-re és NetBSD-re készített programok nagy részével.
- *Futtatásra kész* alkalmazások ezrei érhetõek el a FreeBSD *port-* és _csomag_gyûjteményében. Miért bújnánk az internetet értük, ha mindent egy helyen is megtalálhatunk?
- További *könnyen portolható* alkalmazások ezrei állnak rendelkezésre az interneten. A FreeBSD forráskódja kompatibilis a legtöbb elterjedt kereskedelmi UNIX® rendszerével, aminek köszönhetõen az alkalmazások nagy része csak kevés módosítást igényel a fordításhoz, már amennyiben erre egyáltalán szükség van.
- Az igény szerinti lapozással mûködõ *virtuális memória* és "egyesített VM/puffer gyorsítótár" úgy lett kialakítva, hogy hatékonyan kiszolgálja a nagyobb étvágyú alkalmazásokat, miközben a többi felhasználó számára továbbra is reakcióképes marad.
- Az *SMP* támogatása a több processzorral rendelkezõ számítógépek számára.
- *C*, *C++* és *Fortran* fejlesztõi eszközök széles tárháza használható. Kutatáshoz és fejlesztéshez más egyéb programozási nyelvek is elérhetõek a portok és csomagok segítségével.
- Az egész rendszer _forráskód_jának megléte lehetõvé teszi, hogy a legnagyobb fokú irányítást élvezhessük a környezetünk felett. Miért is bíznánk magunkat egy zárt rendszert fejlesztõ cégre, mikor lehetne egy igazán nyílt rendszerünk?
- Nagy mennyiségû *internetes dokumentáció*.
- *Még sok minden más!*

A FreeBSD Kaliforniai Egyetem (Berkeley) Számítógépes rendszerek kutatócsoportja által fejlesztett 4.4BSD-Lite kiadásán alapszik és ápolja a BSD-rendszerek fejlesztésének jellegzetes hagyományait. Túl a kutatócsoport kivételes munkáján, a FreeBSD Projekt több ezernyi órát szentelt arra, hogy a legtöbbet hozza ki a rendszerbõl mind a teljesítményt, mind pedig a valós életben felbukkanó terhelési helyezetekben történõ helytállást illetõen. Ahogy a legnagyobb piaci óriások igyekeznek egy hasonló képességû, teljesítményû és megbízhatóságó PC-s operációs rendszert kifejleszteni, úgy a FreeBSD már *most* felajánlja ezeket!

Kizárólag csak a képzeletünk szabhat gátat annak, hogy mire is tudjuk használni a FreeBSD-t. Szoftverfejlesztéstõl kezdve, a gyári automatizáláson és készletnyilvántartáson át a mûholdas antennák tájolásáig szinte mindenre: ha ezt eddig egy kereskedelmi UNIX®-szal is meg tudtuk tenni, akkor nagyon valószínû, hogy a FreeBSD-vel is képesek leszünk erre! A FreeBSD ezen felül nagyban profitál a világban található különbözõ kutatóközpontok és egyetemek által fejlesztett, kiváló minõségû alkalmazások ezreibõl, melyek gyakorta olcsón vagy ingyen elérhetõek. Kereskedelmi alkalmazások is egyre nagyobb számban képviseltetik magukat minden nap.

Mivel a FreeBSD forráskódja általánosan elérhetõ, a rendszer szinte tetszõleges mértékben testreszabható a különleges elvárásokat támasztó alkalmazások vagy projektek számára. Ez a

nagyobb kereskedelmi fejlesztõk operációs rendszereivel majdnem teljesen elképzelhetetlen. Íme csupán néhány példája azon alkalmazásoknak, melyek jelenleg is FreeBSD-t használnak:

- *Internetes szolgáltatások:* A FreeBSD-be épített szilárd TCP/IP alapú hálózatkezelés különféle internetes szolgáltatások számára teszi ideális platformmá:
  - FTP szerverek
  - World Wide Web szerverek (hagyományos vagy biztonságos [SSL])
  - IPv4 és IPv6 forgalomirányítás
  - Tûzfalak és NAT ("IP maszkolás"), átjárók
  - Elektronikus levelezõ szerverek
  - USENET hírrendszer és üzenõfal
  - Sok minden más...

  A FreeBSD használatához kezdetben elegendõ egy olcsó 386-os PC, melyet a vállalkozásunk fejlõdésével szépen fel tudunk hozni egy RAID-del ellátott négyprocesszoros Xeon rendszerig.

- *Oktatás:* Esetleg informatikával vagy mûszaki informatikával foglalkozik? Nem is lehetne jobban a FreeBSD által felkínált élményeken kívül máshogy megismerkedni elsõkézbõl az operációs rendszerek, számítógépes architektúrák és hálózatok mûködésével! Rengeteg szabadon használható mûszaki, matematikai és grafikai tervezõ programcsomag könnyíti meg azok munkáját is, akik számára a számítógép legfõképpen *más* feladatok elvégzésére hivatott!
- *Kutatás:* Miután a teljes FreeBSD rendszer forráskódja bárki számára elérhetõ, tökéletes kiindulási pontot ad az operációs rendszerek témakörében vagy a számítástudomány egyéb ágaiban végzendõ kutatásokhoz. A FreeBSD nyílt természete ezenkívül lehetõvé teszi egymástól távol levõ csoportok közös együttmûködését is anélkül, hogy a résztvevõknek aggódnia kellene a különleges licencszerzõdések vagy a nyílt fórumokon felmerülõ korlátozások miatt.
- *Hálózatépítés:* Szüksége van egy új útválasztóra? Esetleg egy névszerverre (DNS)? Egy tûzfalra, mely távoltartja a nemkívánatos egyéneket a belsõ hálózattól? A FreeBSD pillanatok alatt átváltoztatja a sarokban porosodó 386-os vagy 486-os PC-nket egy kifinomult csomagszûrési képességekkel bíró forgalomirányító eszközzé.
- *X Window munkaállomás:* A FreeBSD a szabadon használható X11 szerverrel együtt remek választás egy olcsó X terminál kiépítéséhez. Eltérõen egy szokványos X terminháltól, a FreeBSD azonban igény szerint sok alkalmazás helyi futtatását is képes megoldani, ezzel megszabadítva minket a központi szerver használatának kényszerétõl. A FreeBSD viszont akár "lemez nélkül" is el tud indulni, aminek révén az egyes munkaállomások karbantartása még olcsóbbá és könnyebbé válik.
- *Szoftverfejlesztés:* Az alap FreeBSD rendszer fejlesztõeszközök tömkelegével, többek közt a híres GNU C/C++ fordítóval és nyomkövetõvel érkezik.

A FreeBSD CD-n, DVD-n és FTP-n keresztül elérhetõ forráskód és bináris formátumban is. A FreeBSD beszerzésével kapcsolatos bõvebb információkért olvassuk el az A FreeBSD beszerzéseet.

1.2.2. Ki használja a FreeBSD-t?

A FreeBSD egyaránt remek eszköz- és termékfejlesztõi platformként funkcionál a világ legnagyobb informatikai cégeinél, többek közt:

- Apple
- Cisco
- Juniper
- NetApp

A FreeBSD mindezek mellett több nagyobb internetes oldal alapját képzi, mint például:

- Yahoo!
- Yandex
- Apache
- Rambler
- Sina
- Pair Networks
- Sony Japan
- Netcraft
- NetEase
- Weathernews
- TELEHOUSE America
- Experts Exchange

és még sokan mások.

1.3. A FreeBSD Projektrõl

A most következõ rész egy-két háttérinformációt tár fel a Projektrõl, többek között a történetét, céljait és a benne alkalmazott fejlesztési modellt.

1.3.1. A FreeBSD rövid története

A FreeBSD Projekt valamikor 1993 kezdetérõl eredeztethetõ, és részben a "Nem hivatalos 386BSD Patchkit"-bõl nõtt ki, a patchkit 3 legutolsó koordinátorának, Nate Williamsnek, Rod Grimesnak és nekem köszönhetõen.

Eredeti célunk a 386BSD köztes állapotainak rögzítése lett volna, amitõl olyan problémák megoldását reméltük, melyeket a patchkitek gyártása önmagában egyszerûen nem tudott megoldani. Néhányan még talán emlékeznek is a Projekt kezdeti munkaneveire: "386BSD 0.5" vagy "386BSD Interim", melyek pontosan erre a tényre hivatkoztak.

A 386BSD eredetileg Bill Jolitz operációs rendszere volt, amely ennél a pontnál már közel egy éve

senki sem tartott karban. Mivel a hozzá tartozó patchkit pedig napról napra duzzadt, egyre kényelmetlenebbé vált a karbantartása. Ezért egyhangúan úgy döntöttünk, segítünk Billnek azzal, hogy idõnként létrehozunk egy "letisztított" változatot. Ez a próbálkozásunk csúnyán kudarcba fulladt, amikor Bill Jolitz hirtelen meggondolta magát és visszalépett a Projekt támogatásától. Semmilyen egyértelmû útmutatást nem adott arra, hogy mit csináljunk helyette.

Nem tartott sokáig eldöntenünk, hogy ez a cél továbbra is megéri a fáradtságot, még Bill segítsége nélkül is, ezért felvettük a "FreeBSD" nevet, melyet David Greenmannek köszönhetünk. Kezdeti feladatainkat a rendszer akkori felhasználóival tartott egyeztetések után állítottuk fel. Miután teljesen tisztán láthatóvá vált, hogy a Projekt a megvalósulás útján van, felvettem a kapcsolatot a Walnut Creek-kel, terjesztési mód után nézve azok számára, akik nem tudtak akkoriban könnyedén hozzáférni az internethez. A Walnut Creek nem csak támogatta a FreeBSD CD-n történõ terjesztését, hanem még egy számítógépet és egy gyors internetkapcsolatot is a Projekt rendelkezésére bocsátott. A Walnut Creek szinte példátlan mértékû, egy akkoriban teljesen ismeretlen projektbe vetett hite nélkül nagyon nehezen lenne elképzelhetõ, hogy a FreeBSD olyan messzire és olyan gyorsan jutott volna el, ahol ma tart.

Az elsõ CD-lemezen (és széles körben az interneten is megjelenõ) változat a FreeBSD 1.0 volt, amely 1993 decemberében jelent meg. A Berkeley-rõl származó 4.3BSD-Lite ("Net/2") szalagokon található források alapján készült, kiegészítve a 386BSD-bõl és a Szabad Szoftver Alapítványtól (Free Software Foundation, FSF) származó komponensekkel. Elsõ kiadásként igen méltányos sikert könyvelhetett el, melyet a még inkább sikeres FreeBSD 1.1-gyel folytattunk 1994 májusában.

Nagyjából ekkortájt néhány váratlan sötét felhõ bukkant fel az égbolton, ahogy a Novell és a Berkeley hosszantartó pereskedése lezárult a Berkeley Net/2 szalagjainak jogi formáját illetõen. Ennek eredményeképpen a Berkeley elfogadta, hogy a Net/2 nagy része "jelzáloggal terhelt" és a Novell tulajdona, aki pedig valamivel korábban az AT&T-tõl szerezte. Ezért cserébe a Berkeley megkapta a Novell "áldását" a 4.4BSD-Lite kiadásra, és amikor az véglesen kijön, megszûnik a rajta levõ jelzálog. Emiatt az összes Net/2 felhasználónak erõsen javasolt volt váltani. Ez érintette magát a FreeBSD-t is, és így a Projekt 1994 júliusáig kapott határidõt, hogy leállítsa a Net/2 alapú termékeinek szállítását. A megegyezés értelmében a Projekt kiadhatott még egy utolsó kiadást a határidõ elõtt, amely végül a FreeBSD 1.1.5.1 lett.

A FreeBSD-nek ekkor szembesülnie kellett azzal a nehéz feladattal, hogy lényegében újra fel kellett találnia magát, a teljesen új és meglehetõsen hiányos 4.4BSD-Lite bitjeitõl elindulva. A "Lite" (egyszerûsített) kiadások abban az értelemben számítottak egyszerûbbnek, hogy a Berkeley kutatói (a különbözõ jogi követelések miatt) eltávolították a ténylegesen beindítható rendszerhez szükséges programrészek nagyobb részét, ill. a 4.4-es verzió Intel processzorokra készített portja nagyon is befejezetlen volt. A Projektnek egészen 1994 novemberéig tartott, hogy megtegye ezt a lépést, ugyanis ekkor jelent meg a FreeBSD 2.0 az interneten és (december vége felé) CD-n. Annak ellenére, hogy még némileg érdes maradt bizonyos helyeken, ez a kiadás jelentõs sikereket ért el. Ezt követte 1995 júniusában a sokkalta stabilabb és könnyebben telepíthetõ FreeBSD 2.0.5.

A FreeBSD 2.1.5-öt 1996 augusztusában adtuk ki, mely akkora népszerûségnek örvendett az internet-szolgáltatók és kereskedelmi közösségek körében, hogy a 2.1-STABLE elágazásból egy újabb kiadást készítettünk. Ez volt a FreeBSD 2.1.7.1, amely 1997 februárjában jelent meg és ezzel együtt a 2.1-STABLE fejlesztését is zárta. Most már csak karbantartást végzünk rajta, és csak a biztonsági és egyéb kritikus hibajavítások kerülnek bele (RELENG_2_1_0).

A FreeBSD 2.2 fejlesztése 1996 novemberében ágazott le az akkori fejlesztõi ("-CURRENT") ágból, mint a RELENG_2_2-es ág. Ebbõl az elsõ teljes kiadás (2.2.1) 1997 áprilisában jelent meg. A 2.2-es ág mentén további kiadások 1997 nyarán és õszén készültek, melyek közül az utolsó (2.2.8) 1998 novemberében jelent meg. Az elsõ hivatalos 3.0-ás kiadás 1998 októberében jött ki, ami egyúttal a 2.2-es ág befejezésének kezdetét jelentette.

A fejlesztési fa 1999. január 20-án került ismét elágaztatásra, melynek eredménye a 4.0-CURRENT és 3.X-STABLE ágak lettek. A 3.X-STABLE ágban a 3.1 1999. február 15-én, a 3.2 1999. május 15-én, a 3.3 1999. szeptember 16-án, a 3.4 1999. december 20-án és a 3.5 2000. június 24-én jelent meg, melyet pár nappal késõbb egy kisebb alverzió, a 3.5.1 követett, a Kerberosra vonatkozó friss biztonsági javításokkal. Ez lett egyben a 3.X ág utolsó kiadása.

Egy másik fontos elágaztatás 2000. március 13-án történt, mellyel életre kelt a 4.X-STABLE ág. Ebbõl aztán számos kiadás született: a 4.0-RELEASE 2000 márciusában mutatkozott be, az utolsó 4.11-RELEASE pedig 2005 januárjában látott napvilágot.

A várva várt 5.0-RELEASE 2003. január 19-én került bejelentésre. Közel háromévnyi munka eredményeképpen ez a kiadás indította meg a FreeBSD-t a többprocesszoros rendszerek és az alkalmazások szálkezelésének fejlettebb támogatásának útján, valamint az UltraSPARC® és `ia64` platformok támogatása is itt jelent meg elõször. Ezt a kiadást az 5.1 követte 2003 júniusában. A hozzá tartozó -CURRENT ágból az utolsó kiadás az 5.2.1-RELEASE volt, amely 2004 februárjában mutatkozott be.

A 2004 augusztusában, a RELENG_5 ág létrehozását a 5.3-RELEASE követte, és egyben a 5-STABLE ág kezdetét is jelezte. A legújabb 5.5-RELEASE 2006 májusában jött ki. A RELENG_5 ágból már nem fog készülni több kiadás.

A fejlesztési fa ezután 2005 júliusában ágazott el ismét, ezúttal a RELENG_6 ágnak adott életet. A 6.0-RELEASE az 6.X ág elsõ kiadásaként 2005 novemberében jelent meg. A legújabb 6.4-RELEASE 2008 november hónapjában jelentkezett. A RELENG_6 ágból már nem készülnek további kiadások.

A RELENG_7 ág 2007 októberében jött létre. Ebbõl az elsõ kiadás 2008 februárjában a 7.0-RELEASE volt. A legfrissebb 11.2-RELEASE kiadás June 28, 2018 hónapban készült el. A RELENG_7 ágból további kiadások is várhatóak.

A fejlesztési fából 2009 augusztusában ismét levált egy ág, amely ezúttal a RELENG_8 volt. A 8.0-RELEASE, a 8.X ág elsõ kiadása 2009 novemberében jelent meg. A legfrissebb 12.0-RELEASE December 11, 2018 hónapban jött ki. A RELENG_8 ágból várhatóak további kiadások.

Jelen pillanatban a hosszabb távú fejlesztések a 9.X-CURRENT (törzs) ágban kapnak helyet, és a 9.X-bõl készült idõközönkénti pillanatkiadások folyamatosan elérhetõek CD-n (és természetesen interneten keresztül is) a pillanatkiadásokat tároló szerverrõl.

1.3.2. A FreeBSD Projekt céljai

A FreeBSD Projekt célja, hogy olyan szoftvereket kínáljon, amelyek tetszõlegesen, bármilyen célra felhasználhatóak, mindenféle megkötések nélkül. Sokunk jelentõs energiát fektet a programokba (és a Projektbe) és minden bizonnyal egyikünk sem utasítana vissza semmilyen anyagi ellenszolgáltatást se most, se késõbb, de egyáltalán nem ragaszkodunk hozzá. Hisszük, hogy elsõdleges "küldetésünk" olyan programok és programrészletek készítése bárki számára és

bármilyen célra, melyeket a lehetõ legszélesebb körben alkalmaznak és a lehetõ legtöbb hasznot hajtják. Ez, úgy érzem, az egyik legalapvetõbb célja a szabad szoftvereknek, és ez az, amit mi is lelkesen magunkénak vallunk.

A forrásfánkban található GNU General Public License (GPL) vagy a Library General Public License (LGPL) alá esõ kódok hozzáférhetõségére ezzel szemben némileg több megszorítás vonatkozik, legalább is inkább ami a hozzáférhetõséget illeti. Mivel a GPL-es szoftverek kereskedelmi használata további bonyodalmakat vethet fel, ha lehetõségünk adódik rá, inkább a sokkal enyhébb BSD licenccel rendelkezõ szoftvereket választjuk.

1.3.3. A FreeBSD fejlesztési modellje

A FreeBSD fejlesztése egy nagyon nyitott és rugalmas folyamat, szó szerint a világ minden tájáról érkezõ többszáznyi segítségbõl építkezik, ahogy az látható is a résztvevõink listáján. A FreeBSD fejlesztési infrastruktúrája lehetõvé teszi, hogy ez a többszáznyi résztvevõ az interneten keresztül mûködjön együtt. Folyamatosan várjuk az új fejlesztõket és ötleteket, és mindazok, akik komolyabban érdeklõdnek a Projekt iránt, egyszerûen felvehetik velünk a kapcsolatot a FreeBSD technical discussions levelezési lista címén. Egy FreeBSD announcements levelezési lista is elérhetõ azok számára, akik értesíteni kívánják a többi FreeBSD felhasználót munkájuk fõbb eredményeirõl.

A FreeBSD Projektrõl és annak fejlesztési modelljérõl hasznos tudni az alábbiakat, függetlenül attól, hogy egyedül vagy másokkal szoros együttmûködésben dolgozunk:

Az SVN és CVS repositoryk

Sok éven keresztül a FreeBSD központi forrásfáját CVS-en (Concurrent Versions System) keresztül tartották karban, amely egy, a FreeBSD-vel is érkezõ, szabadon elérhetõ verziókezelõ rendszer. 2008 júniusában a Projekt az SVN (Subversion) használatára váltott. Ez a váltás szükségszerû volt, mivel a CVS által okozott technikai nehézségek gyorsan elõjöttek a forrásfa és a hozzá tartozó metainformációk szapora növekedésével. Noha a központi repository most már SVN-alapú, a kliensoldali CVSup és csup alkalmazások továbbra is a korábbi infrastruktúrával dolgoznak, ahogy eddig is - az SVN respositoryban végzett változtatások ehhez automatikusan átkerülnek CVS alá. Jelen pillanatban egyedül csak a központi forrásfa használja ezt a megoldást, a dokumentáció, a weboldalak és a Portgyûjtemény forrásai továbbra is CVS alól üzemelnek. Az elsõdleges CVS repository egy Santa Clara-i (California, USA) számítógépen található, ahonnan a világban található rengeteg tükörszerverre másolódik. Az SVN-fa, mely tartalmazza a -CURRENT és -STABLE ágakat, könnyen lemásolható a saját számítógépünkre is. Ennek részleteirõl bõvebben a A forrásfa szinkronizálása c. szakaszban olvashatunk.

A committerek listája

A hivatalos fejlesztõk (*committerek*) azok az emberek, akik a CVS-fához írási joggal rendelkeznek, tehát módosítást hajthatnak végre a FreeBSD forrásaiban (a "committer" kifejezés a cvs(1) `commit` parancsából származik, amelyet arra használunk, hogy felvigyük a módosításainkat a CVS repository-ba). Javaslatainkat legjobban a send-pr(1) használatával tudjuk a committerek elé tárni. Ha valamiért ez mégsem mûködne, megpróbálhatjuk õket elérni közvetlenül a FreeBSD committer's mailing list címére küldött e-maillel.

A FreeBSD Core Team

Ha a FreeBSD Projekt egy vállalat lenne, akkor a *FreeBSD Core Teamje* (irányító csoportja)

foglalná magában a vezetõséget. Ennek a csoportnak elsõdleges feladata, hogy fenntartsa a Projekt egészének kondícióját és gondoskodjon róla, hogy a megfelelõ irányba haladjon. Az irányító csoportnak ugyanígy feladata a megbízható és odaadó committerek tömörítése és az új tagok beszervezése, ha a csoportból kilépne valaki. A jelenlegi Core Team tagjait 2008 júliusában választották meg. A választásokat kétévente tartják.

Ebben a csoportban egyes tagoknak ezenfelül még bizonyos területekre felügyelniük is kell. Ez azt jelenti, hogy felelõsek a rendszer valamelyik nagyobb részének az elõírásoknak megfelelõ mûködéséért. A FreeBSD fejlesztõk teljes felsorolása és a hozzájuk tartozó területek megtalálhatóak A résztvevõk listjában.

A Core Team legtöbb tagja pusztán önkéntesen vesz részt a FreeBSD fejlesztésében és nem származik a projektbõl semmilyen anyagi haszna. Emiatt a "részvétel" nem tévesztendõ össze a "garantált támogatással". A "vezetõségre" vonatkozó hasonlat nem teljesen pontos abban az értelemben, hogy ezek az emberek tulajdonképpen egy kívülálló szempontjából ésszerûtlen döntést hoztak azzal, hogy a FreeBSD támogatására áldozták az életüket!

Külsõ résztvevõk

Végül, de nem utoljára, következzen a fejlesztõk legnagyobb csoportja: õk maguk a felhasználók, akik rendszeres visszajelzéseket és hibajavításokat küldenek. A FreeBSD kevésbé központosított fejlesztésében elsõsorban a FreeBSD technical discussions levelezési lista segítségével lehet felvenni a fonalat, ahol ezeket a témákat tárgyalják meg. A FreeBSD-hez kapcsolódó különféle levelezési listákról többet a Források az interneten ben olvashatunk.

A FreeBSD résztvevõinek listája hosszú és még most is növekszik; miért nem próbálunk mi is visszaadni valamit a FreeBSD-nek?

Nem csak programozással lehet segíteni a Projektet: a megoldandó feladatok listáját megtalálhatjuk a FreeBSD Projekt honlapján.

Röviden összefoglalva, a fejlesztési modellünk egymáshoz lazán kapcsolódó koncentrikus körökként szervezõdik. Ez a központosított modell a FreeBSD-*felhasználók* kényelmét szolgálandó lett kialakítva, akik így könnyedén tudnak követni egyetlen központi kódbázist, azonban megvan a lehetõségük a részvételre is! Minden vágyunk egy olyan megbízható operációs rendszer kialakítása, amihez nagy mennyiségû könnyen telepíthetõ és használható alkalmazás tartozik - ez a modell ennek elérésére nagyon is megfelelõ.

A haladás ütemének fenntartása érdekében mindössze csak annyit kérünk a leendõ FreeBSD fejlesztõinktõl, hogy legyenek legalább annyira elszántak, mint a jelenlegi tagjaink!

1.3.4. Az aktuális FreeBSD kiadások

A FreeBSD egy szabadon elérhetõ, teljes forráskóddal érkezõ 4.4BSD-Lite alapú kiadás Intel i386™, i486™, Pentium®, Pentium® Pro, Celeron®, Pentium® II, Pentium® III, Pentium® 4 (vagy azzal kompatibilis), Xeon™, DEC Alpha™ és Sun UltraSPARC® alapú számítógépekre. Elsõsorban a Berkeley Számítógépes rendszerek kutatócsoportjának szoftverein alapszik, számos javítással a NetBSD, OpenBSD, 386BSD és a Szabad Szoftver Alapítvány munkásságának köszönhetõen.

A FreeBSD 2.0 1994 végi megjelenése óta a FreeBSD teljesítménye, megbízhatósága és tudása drasztikusan megnövekedett. A legnagyobb változtatás az újjáalakított, összevont VM/állomány puffer gyorsítótárral rendelkezõ virtuális memória alrendszer, amely nem csak a teljesítményt növeli, hanem csökkenti a FreeBSD memóriaigényét is, jobban elfogadhatóvá téve ezzel az 5 MB-os minimumot. A további fejlesztések között találjuk a teljes NIS szerver és kliens támogatást, az átviteli TCP támogatását, az igény szerint tárcsázó PPP-t, a beépített DHCP támogatást, a továbbfejlesztett SCSI alrendszert, az ISDN támogatást, az ATM, FDDI, Fast és Gigabit Ethernet (1000 Mbit) hálózati csatolók támogatását, a legfrissebb Adaptec gyártmányú vezérlõk fejlesztett támogatását és a többezernyi hibajavítást.

Az alapeszközök mellé a FreeBSD felkínálja többezernyi ismert és keresett program portjaiból álló gyûjteményét. Ebben a pillanatban is már több, mint 37000 port érhetõ el! A portok listája a HTTP (WWW) szerverektõl, a játékokon, nyelveken és sok mindenen keresztül a szövegszerkesztõkig terjed. Az egész Portgyûjtemény közelítõleg 3 GB tárhelyet kíván, minden portot az eredeti forráshoz viszonyított "különbségként" tárol. Ennek következtében a portok frissítése sokkal könnyebb és nagyban csökkenti a korábbi, 1.0-ás Portgyûjteménynél kialakult tárigényeket. Egy port lefordításához egyszerûen csak be kell lépnünk a telepíteni kívánt program könyvtárába és ki kell adnunk a `make install` parancsot, a többit a rendszer elvégzi. Minden egyes telepítendõ port teljes forrása dinamikusan vagy CD-rõl vagy pedig FTP-n keresztül töltõdik le, így csak a ténylegesen telepítendõk lefordításához elegendõ tárhelyre van szükség. Majdnem mindegyik port elérhetõ elõre lefordított "csomag" formájában azok számára, akik nem kívánják lefordítani a portokat, és melyeket egy egyszerû parancs (`pkg_add`) segítségével telepíteni is tudják. A csomagokról és portokról a Alkalmazások telepítése: csomagok és portokben tudhatunk meg többet.

A FreeBSD telepítésérõl és használatáról most már számos további nagyon hasznos dokumentumot találhatunk bármelyik FreeBSD-s számítógép /usr/shared/doc könyvtárában. A helyileg telepített kézikönyveket bármilyen HTML-t megjeleníteni képes böngészõvel el tudjuk olvasni az alábbi URL-eken:

A FreeBSD kézikönyv

/usr/shared/doc/handbook/index.html

A FreeBSD GYIK

/usr/shared/doc/faq/index.html

Az aktuális (leginkább frissített) verziók megtekinthetõek a http://www.FreeBSD.org/ címen.

Chapter 2. A FreeBSD telepítése

2.1. Áttekintés

A FreeBSD telepítéséhez egy könnyen használható szöveges telepítõprogram, a sysinstall használható. Ez a FreeBSD alapértelmezett telepítõprogramja, habár ezt a különféle gyártók kedvük szerint lecserélhetik. Ebben a fejezetben bemutatjuk a FreeBSD sysinstall segítségével történõ telepítését.

A fejezet elolvasása során megismerjük:

- hogyan készítsünk telepítõlemezeket a FreeBSD-hez;
- a FreeBSD miként hivatkozza és osztja fel a merevlemezeinket;
- hogyan indítsuk el a sysinstall programot;
- milyen kérdéseket tesz fel nekünk a sysinstall, mire gondol, hogyan is kell azokat megválaszolni.

A fejezet elolvasásához ajánlott:

- a telepítendõ FreeBSD verzióhoz tartozó támogatott hardvereket felsoroló lista átolvasása és benne a saját hardvereszközeink megkeresése.

Általánosan elmondható, hogy a most következõ telepítési utasítások az i386™ ("PC kompatibilis") architektúrájú számítógépekre vonatkoznak. Ahol erre szükség van, ott más platformokra vonatkozó utasítások is szerepelhetnek. Habár ezt a leírás igyekszünk a lehetõ legjobban naprakészen tartani, elképzelhetõ, hogy felfedezhetünk kisebb eltéréseket a telepítõben és az itt leírtak közt. Ezért ezt a fejezetet inkább egy általános útmutatónak javasoljuk, nem pedig egy szó szerint értelmezendõ kézikönyvként.

2.2. Hardverkövetelmények

2.2.1. Minimális konfiguráció

A FreeBSD telepítéséhez szükséges minimális konfiguráció FreeBSD verziónként és architektúránként eltérõ.

A minimális konfigurációt a FreeBSD honlapján a kiadásokról szóló oldalon, az "Installation Notes" részben találhatjuk meg. Ezt a következõ szakaszokban foglaljuk össze. A FreeBSD telepítésének módszerétõl függõen szükségünk lehet egy hajlékonylemezes (floppy) vagy CD-ROM meghajtóra, esetleg egy hálózati kártyára. Ezt a Készítsünk egy rendszerindító lemezt ban tárgyaljuk.

FreeBSD/i386 és FreeBSD/pc98

A FreeBSD/i386 és FreeBSD/pc98 egyaránt egy 486 vagy jobb processzort és legalább 24 MB memóriát igényel. A legkisebb telepítéshez legalább 150 MB szabad lemezterület szükséges.

Régebbi konfigurációk esetén nem egy gyorsabb processzor, hanem inkább több memória beszerzése, illetve több lemezterület felszabadítása a fontosabb.

FreeBSD/alpha

Az Alpha támogatás a FreeBSD 7.0 beindulásával eltávolításra került. A FreeBSD 6.*X* sorozat az utolsó, amely valamilyen támogatást ajánl ehhez az architektúrához. Ezzel kapcsolatban részletesebben a kiadásokkal kapcsolatos információkat tartalmazó oldalon olvashatunk a FreeBSD honlapján.

FreeBSD/amd64

Két típusú processzor képes futtatni a FreeBSD/amd64 verzióját. Az elsõ ezek közül az AMD64 processzorok, beleértve az AMD Athlon™64, AMD Athlon™64-FX, AMD Opteron™ vagy újabb processzorokat.

A FreeBSD/amd64 verzióját kihasználni képes processzorok másik csoportja az Intel® EM64T architektúrájára épülõ processzorok. Ilyen processzor például az Intel® Core™ 2 Duo, Quad és Extreme processzorcsaládok, valamint az Intel® Xeon™ 3000, 5000 és 7000 sorozatszámú processzorai.

Ha nVidia nForce3 Pro-150 alapú géppel rendelkezünk, ki *kell* kapcsolnunk a BIOS-ban az IO APIC használatát. Ha nem találnánk ilyen beállítást, akkor helyette magát az ACPI-t kell kikapcsolnunk. A Pro-150 chipsetnek vannak bizonyos hibái, amelyekre eddig még nem sikerült megfelelõ megoldást találnunk.

FreeBSD/sparc64

A FreeBSD/sparc64 telepítéséhez egy támogatott platformra van szükségünk (lásd: Támogatott hardverek).

A FreeBSD/sparc64 telepítéséhez egy egész lemezre lesz szükségünk, mivel a rendszer jelenleg nem képes megosztani azt más operációs rendszerekkel.

2.2.2. Támogatott hardverek

A FreeBSD minden kiadásához mellékelik a támogatott hardverek listáját "FreeBSD Hardware Notes" címmel. Ez a dokumentum többnyire a HARDWARE.TXT nevû állomány, amelyet a rendszer CD-n vagy FTP-n keresztül elérhetõ változatának gyökerében vagy a sysinstall dokumentációkat tartalmazó menüjében találhatunk meg.

2.3. A telepítés elõtt elvégzendõ feladatok

2.3.1. Készítsünk leltárt a számítógépünkrõl

A FreeBSD telepítése elõtt érdemes összeszedni, pontosan mi minden is található a számítógépünkben. A FreeBSD telepítõrutinjai mutatni fogják a különbözõ komponensek (merevlemezek, hálózati kártyák, CD-meghajtók és a többi) modelljét és gyártóját. A FreeBSD

ezenkívü megpróbálja kideríteni a megjelenõ eszközök pontos konfigurációját is, beleértve a használt IRQ és IO portok kiosztását. A PC-s hardverek különféle szeszélyei miatt azonban ez az iménti folyamat nem minden esetben megbízható, ezért elõfordulhat, hogy helyesbíteni kell a FreeBSD által megállapított értékeket.

Ha már van a gépünkön egy másik operációs rendszer, például Windows® vagy Linux®, akkor mindenképpen hasznos lehet az általa felkínált eszközökkel lekérdezni a hardvereink beállításait. Ha nem lennénk biztosak benne, hogy az adott bõvítõkártyákat pontosan milyen beállításokkal is használjuk, nézzük meg ezeket magán a kártyán. A népszerû IRQ értékek általában a 3, 5 és 7, valamint az IO portok számát általában tizenhatos számrendszerben szerepeltetik, például 0x330.

Javasoljuk, hogy nyomtassuk ki vagy írjuk le ezeket a paramétereket a FreeBSD telepítése elõtt. Ehhez rendezzük ezeket egy táblázatban, valahogy így:

Táblázat 1. Példa egy eszközleltárra

Eszköz neve	IRQ	IO portok	Megjegyzés
Elsõ merevlemez	-	-	Mérete 40 GB, gyártmánya Seagate, elsõdleges IDE master
CD-ROM meghajtó	-	-	Elsõdleges IDE slave
Második merevlemez	-	-	Mérete 20 GB, gyártmánya IBM, másodlagos IDE master
Elsõ IDE vezérlõ	14	0x1f0	
Hálózati kártya	-	-	Intel® 10/100
Modem	-	-	3Com® 56K-s faxmodem, COM1
…	…	…	…

Ahogy elkészítettük a számítógépünk alkatrészeit tartalmazó listát, vessük ezeket össze a telepítendõ FreeBSD kiadás által megkövetelt eszközökkel.

2.3.2. Mentsük le az adatainkat

Amennyiben a FreeBSD telepítéséhez használt számítógép számunkra értékes adatokat tárol, igyekezzünk lementeni ezeket, és a FreeBSD tényleges telepítése elõtt gyõzõdjünk is meg róla, hogy a mentés sikeres volt. A FreeBSD telepítõrutinjai természetesen megerõsítést fognak kérni bármilyen adat lemezre írása elõtt, azonban ha egyszer már elindítottuk a folyamatot, már semmit sem tudunk visszafordítani.

2.3.3. Döntsük el a FreeBSD telepítésének helyét

Ha a FreeBSD telepítéséhez az egész merevlemezünket fel akarjuk használni, akkor még nincs miért izgatnunk magunkat - nyugodtan átléphetjük ezt a szakaszt.

Amikor viszont a FreeBSD-t más operációs rendszerek mellé szeretnénk telepíteni, ismernünk kell,

miként is helyezkednek el az adatok a lemezeken, és hogy ez miként is érint bennünket.

A lemezek kiosztása a FreeBSD/i386 esetén

A PC-k által használt lemezek különálló darabokra tagolhatóak. Ezeket a darabokat *partíciók*nak nevezzük. Mivel azonban a FreeBSD maga is tárol partíciókat, ezért ez az elnevezés pillanatok alatt megtévesztõvé válhat, ezért ezeket a lemezdarabokat a FreeBSD lemezslice-oknak vagy egyszerûen csak slice-oknak hívja. Például a PC-s lemezpartíciókkal dolgozó, `fdisk` nevû FreeBSD-s segédprogram partíciók helyett is slice-okra hivatkozik. A PC lemezenként alapvetõen csak négy partíciót enged meg. Ezeket a partíciókat nevezik *elsõdleges partícióknak*. Ettõl a korlátozástól egy új típus, a *kiterjesztett partíció* létrehozásával szabadultak meg, amivel így négynél több partíció is készíthetõ. Lemezenként egyetlen ilyen kiterjesztett partíció található, de ezen belül speciális, ún. *logikai partíciók* hozhatóak létre.

Minden partíciónak van egy *partíció-azonosítója*, melyet a partíción található adatok típusának megállapítására használnak. A FreeBSD partícióinak azonosítója a `165`.

Általánosságban véve minden operációs rendszer így azonosítja a partíciókat. Például a DOS és annak leszármazottai, mint például a Windows®, minden elsõdleges és logikai partícióhoz egy C: -tõl induló *meghajtó-betûjelet* társít.

A FreeBSD-t egy elsõdleges partícióra kell telepíteni. A FreeBSD az összes adatát, beleértve minden általunk létrehozott állományt is, ezen az egyetlen partíción fogja elhelyezni. Ha viszont több lemezünk van, többen is, vagy akár mindegyiken létrehozhatunk FreeBSD-s partíciókat. A FreeBSD telepítésekor azonban legalább egy ilyen partíciónak használhatónak kell lennie. Ez lehet elõre megtisztított üres partíciói is, vagy akár egy olyan partíció, amelyen már nem használt adatok vannak.

Ha már mindegyik partíciónk betelt, akkor a többi operációs rendszer által felkínált eszközök (például MS-DOS®-ban vagy Windows®-ban az `fdisk`) valamelyikével elõször fel kell közülük szabadítanunk egyet a FreeBSD számára.

Amennyiben akadna egy használható partíció, akkor használjuk azt. Ekkor azonban elõfordulhat, hogy ehhez elõször a meglévõk közül össze kell majd zsugorítanunk valamelyiket.

A FreeBSD legkisebb telepíthetõ változata nagyjából 100 MB lemezterületet igényel. Azonban ez egy *nagyon* kicsi változat és szinte semmi helyet nem hagy a saját állományainknak. Sokkal valósághûbb, ha grafikus felület nélkül nagyjából 250 MB-ot mondunk, és legalább 350 MB-ot a grafikus felület használata esetén. Ha ezeken felül további szoftvereket is telepíteni kívánunk, még több helyre lesz szükségünk.

Amikor a FreeBSD számára akarunk helyet csinálni, vagy partíciókat akarunk átméretezni, használjuk például a PartitionMagic® nevû kereskedelmi szoftvert, vagy esetleg egy olyan szabad szoftvert, mint például a GParted. Ismereteink szerint a PartitionMagic® és a GParted is használható az NTFS partíciókkal. A GParted számos live linuxos disztribúción megtalálható, ilyen többek közt a SystemRescueCD.

Gondok lehetnek azonban a Microsoft® Vista által használt partíciókkal. Ezért nem árt, ha az átméretezésekor a kezünk ügyében van a Vista telepítõ CD-je. Természetesen, mint minden lemezkarbantási mûvelet esetén, ilyenkor is határozottan ajánlott biztonsági mentéseket készíteni.

Az említett eszközök helytelen használata megsemmisítheti a lemezeinken tárolt adatokat, ezért a használatuk elõtt gondoskodjunk friss, mûködõképes biztonsági mentésekrõl.

Példa 1. Meglevõ partíció használata a méret megváltoztatása nélkül

Tegyük fel, hogy a számítógépünkben egyetlen 4 GB méretû lemez van, amelyen megtalálható a Windows® valamelyik verziója, és ezt a lemezt korábban két, egyaránt 2 GB méretû meghajtóra osztottuk, a C:-re és D:-re. 1 GB adatunk van a C: meghajtón és fél GB a D:-n.

Mindez tehát azt jelenti, hogy a lemezünkön két partíció található, betûjelenként egy. Ha átmásoljuk a D: meghajtón levõ adatainkat a C: meghajtóra, akkor ezzel felszabadíthatjuk a FreeBSD számára a második partíciót.

Példa 2. Meglevõ partíció zsugorítása

Tegyük fel, hogy a számítógépünkben egyetlen 4 GB méretû lemez van, amelyet teljes egészében a Windows® valamelyik példánya foglal el. A Windows® telepítése során ezért minden bizonnyal egyetlen nagy partíciót hoztunk létre, amely a C: betûjelet kapta és a mérete 4 GB. Jelen pillanatban másfél GB helyet használunk a lemezen, és szeretnénk a FreeBSD számára 2 GB helyet felszabadítani.

A FreeBSD telepítéséhez a következõk valamelyikét kell tennünk:

1. Mentsük le a Windows®-os adatainkat, telepítsük újra a Windows®-t úgy, hogy egy 2 GB méretû partíciót választunk neki a telepítése során.
2. A partíció összezsugorítására használjuk az elõbb említett alkalmazásokat, például a PartitionMagic®-et.

2.3.4. Szedjük össze a hálózati beállításainkat

Amennyiben a FreeBSD telepítésének részeként hálózatra is szándékozunk csatlakozni (például egy FTP vagy NFS szerverrõl akarunk telepíteni), ismernünk kell a hálózatra vonatkozó beállításainkat is. A telepítõ rá fog kérdezni ezekre az információkra, amelyek megadása után a FreeBSD a telepítés befejezéséhez csatlakozni tud majd a hálózatra.

Csatlakozás Ethernet-hálózaton, kábel- vagy DSL-modemen keresztül

Ha egy Ethernet-hálózathoz, vagy magához az internethez csatlakozunk egy DSL- vagy kábelmodemen keresztül, akkor az alábbi adatokra lesz szükségünk:

1. IP-cím
2. Az alapértelmezett átjáró IP-címe
3. A gépünk neve
4. DNS (névfeloldó) szerverek IP-címei
5. Hálózati maszk

Ha nem ismerjük ezeket, érdeklõdjünk a rendszergazdától vagy a szolgáltatónktól. Elképzelhetõ az is, hogy mindezen információkat *DHCP* segítségével, automatikusan kapjuk meg. Ezt is mindenképpen jegyezzük fel.

Kapcsolódás modemmel

Ha az internet-szolgáltatónkhoz hagyományos modemen keresztül csatlakozunk, akkor is tudjuk telepíteni a FreeBSD-t interneten keresztül, azonban ez nagyon sokáig tarthat.

Ehhez tudnunk kell:

1. Az internet-szolgáltatónk behívószámát
2. A soros (COM) port számát, amelyen keresztül a modem kapcsolódik a gépünkhöz
3. Az internet-szolgáltatónktól kapott felhasználói nevet és jelszót

2.3.5. Olvassuk el FreeBSD hibajegyzékét

Habár a FreeBSD Projekt igyekszik a FreeBSD minden egyes kiadását a lehetõ legmegbízhatóbban felkészíteni, hibák óhatatlanul is maradnak bennük. Nagyon ritka esetekben ezek a hibák magára a telepítés folyamatára is kihathatnak. Amint ezeket a problémákat sikerül felderíteni és javítani, rögvest megjelennek a FreeBSD honlapján található hibajegyzékben (angolul). A telepítés elõtt ezért mindig ajánlott átolvasni ezt a dokumentumot, így megbizonyosodunk róla, hogy semmilyen utólag felmerült probléma nem akadályozza munkánkat.

Az összes kiadáshoz tartozó információ, beleértve az egyes kiadások hibajegyzékeit is, a FreeBSD honlapjáról a kiadásokra vonatkozó információkat tartalmazó részen érhetõ el (angolul).

2.3.6. Szerezzük be a FreeBSD telepítéséhez szükséges állományokat

A FreeBSD telepítése az alábbi helyek bármelyikén megtalálható állományok felhasználásával történik:

Lokálisan:

- CD vagy DVD
- Ugyanazon a számítógépen levõ MS-DOS® partíció
- Pendrive (USB-flash-tároló)
- SCSI- vagy QIC-szalag
- Floppylemezek

Hálózaton keresztül:

- FTP oldalról, tûzfalon keresztül vagy szükség szerint HTTP proxy használatával
- NFS szerverrõl
- Párhuzamos vagy soros vonali kapcsolaton keresztül

Ha megvásároltuk a FreeBSD telepítõ CD-jét vagy DVD-jét, akkor már mindennel rendelkezünk a telepítéshez. Lépjünk bátran tovább a következõ szakaszra (Készítsünk egy rendszerindító lemezt)!

Ha eddig még nem szereztük volna be a FreeBSD telepítéséhez szükséges állományokat, ugorjunk a Saját telepítõeszköz elkészítésehoz, ahol megtudhatjuk, hogyan készítsük elõ a FreeBSD telepítését az imént felsorolt helyzetekben. A szakasz elolvasása után pedig jöjjünk vissza ide, majd folytassuk az olvasást a Készítsünk egy rendszerindító lemeztban.

2.3.7. Készítsünk egy rendszerindító lemezt

A FreeBSD telepítése úgy kezdõdik, hogy a számítógépünkkel a FreeBSD telepítõjét indítjuk el - ez viszont nem egy olyan program, amit más operációs rendszerben el tudunk indítani. A számítógépünk általában a merevlemezünkre telepített operációs rendszert indítja el, azonban beállítható úgy is, hogy az indulásához egy ún. "rendszerindító" (bootolható) floppy lemezt használjon. Napjaink számítógépei azonban a CD-meghajtóban levõ CD-krõl vagy USB lemezrõl is el tudnak indulni.

Ha CD-n vagy DVD-n megvan a FreeBSD telepítõje (akár megvettük, akár éppen magunk készítettük) és a számítógépünk tud CD-rõl vagy DVD-rõl rendszert indítani (a BIOS-ban van egy "Boot Order" vagy hozzá hasonló nevû beállítás), akkor kihagyhatjuk ezt a szakaszt. A FreeBSD CD- és DVD image-ek kiírásával egy rendszerindításra alkalmas lemezt kapunk, amirõl minden további elõkészület nélkül telepíthetünk.

Rendszerindításra alkalmas pendrive-ot az alábbi lépések mentén tudunk készíteni:

1. Az image állomány letöltése

   A pendrive-okhoz készült image állományok a ISO-IMAGES/ könyvtárból tölthetõek le, `ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/architektúra/ISO-IMAGES/verzió/FreeBSD-12.0-RELEASE-architektúra-memstick.img` néven. Az *architektúra* és *verzió* helyére a telepítendõ architektúrát és verziószámot helyettesítsük be. Ennek megfelelõen tehát például a FreeBSD/i386 12.0-RELEASE változata a ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/ISO-IMAGES/12.0/FreeBSD-12.0-RELEASE-i386-memstick.img címrõl érhetõ el.

   A pendrive image .img kiterjesztéssel rendelkezik. A ISO-IMAGES/ könyvtár általában több különféle állományt tartalmaz, ezek közül kell választanunk a FreeBSD telepítendõ változatának, és sok esetben a telepítéshez rendelkezésre álló hardver típusának megfelelõen.

   A következõ lépés megkezdése elõtt *készítsünk biztonsági mentést* a pendrive tartalmáról, mivel minden rajta levõ adat *törlõdni* fog.

2. A pendrive elõkészítése

   Az itt található példában a rendszerindításhoz és így a mûvelet végrehajtásához a /dev/da0 nevû eszközt fogjuk használni. Ezt ne felejtsük el helyettesíteni a rendszerünkön erre a célra használt eszköz nevével, máskülönben kárt tehetünk az adatainkban.

A `kern.geom.debugflags` változó értékének megfelelõ beállításával engedélyezzük a céleszközön a Master Boot Record írását.

```
#  sysctl kern.geom.debugflags=16
```

3. Az image pendrive-ra írása

   Az .img kiterjesztésû állományt *nem* egyszerûen a pendrive-ra kell másolni, ez a lemez teljes tartalmát magában foglalja. Ennek megfelelõen *nem* egyszerûen állományokat kell másolnunk az egyik lemezrõl a másikra. Helyette a dd(1) parancs segítségével írjuk az image állomány tartalmát közvetlenül a lemezre.

   ```
   # dd if=FreeBSD-12.0-RELEASE-i386-memstick.img of=/dev/da0 bs=64k
   ```

Rendszerindításra alkalmas floppy lemezt az alábbi lépések mentén tudunk készíteni:

1. A rendszerindító lemezek image-einek beszerzése

   A FreeBSD 8.0 kiadásától kezdõdõen megszûnik a floppy lemezek támogatása. Helyette telepítsünk pendrive-ról, amelyrõl fentebb olvashatunk, vagy egyszerûen használjunk CD-t vagy DVD-t.

   A rendszerindító lemezek a telepítõeszköz floppies/ könyvtárában találhatóak, illetve letölthetõek az `ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/architektúra/változat-RELEASE/floppies/` helyrõl. Az *architektúra* és *változat* helyére természtesen írjuk be a telepíteni kívánt architektúrát és verziót. Így például a FreeBSD/i386 12.0-RELEASE rendszerindító lemezei az ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/12.0-RELEASE/floppies/ címrõl érhetõek el.

   A floppyk image-ei .flp kiterjesztésûek. A floppies/ könyvtár számos különféle image-et tartalmaz, ezek közül leginkább a telepítendõ FreeBSD változat, valamint emellett olykor konkrétan a hardver határozza meg a használandót. Az esetek túlnyomó részében négy floppyra lesz szükségünk: boot.flp, kern1.flp, kern2.flp és kern3.flp. A lemezek image-eit illetõ legfrissebb információkat ugyanazon a könyvtáron belül szereplõ README.TXT állományban olvashatjuk (angolul).

   Az FTP-hez használt programunkat az image-ek letöltése során ne felejtsük el *bináris (binary)* átviteli módban használni. Egyes böngészõk hajlamosak ugyanis *szöveges (text vagy ASCII)* átviteli módot használni, ami viszont csak abból vehetõ észre, hogy nem tudjuk a lemezekrõl elindítani a rendszert.

2. A floppyk elõkészítése

   Mindegyik letöltendõ image-hez elõ kell készíteni egy-egy hajlékonylemezt. Nagyon fontos,

hogy ezek a lemezek teljesen hibátlanok legyenek. Errõl a legkönnyebben úgy gyõzõdhetünk meg, ha a lemezeket magunk formázzuk, és nem bízunk a különféle elõreformázott (preformatted) floppykban. A Windows®-ban található formázó segédprogram sem árul el nekünk semmit a lemezeken található hibás részekrõl, egyszerûen csak "rossznak" (bad) jelöli meg és figyelmen kívül hagyja ezeket. Határozottan ajánljuk, hogy amennyiben a telepítésnek ezt a módját választjuk, mindig használjunk teljesen új floppykat.

Ha megpróbáljuk telepíteni a FreeBSD-t, és a telepítõprogram összeomlik, lefagy vagy bármilyen furcsaságot mûvel, elsõként mindenképpen a floppykra gyanakodhatunk. Ilyenkor írjuk ki az image-eket új lemezekre és próbálkozzunk újra a telepítéssel.

3. Az image állományok írása a floppykra

   Az .flp kiterjesztésû állományok *nem* a lemezre másolható hagyományos állományok, hanem a lemezek teljes tartalmának képei, ezért ezeket egyszerûen *nem* másolhatjuk egyik lemezrõl a másikra. Az image-ek közvetlen lemezreírásához ehelyett kifejezetten erre a célra alkalmas eszközöket kell használnunk.

   Azok számára, akik a floppykat MS-DOS®/Windows® rendszerû számítógépeken kívánják elkészíteni, mellékeltünk egy `fdimage` nevû segédprogramot.

   Ha a CD-meghajtónk betûjele például E: és a telepítõ CD-n található image-eket szeretnénk kiírni vele, akkor ezt a parancsot kell kiadnunk:

   ```
   E:\> tools\fdimage floppies\boot.flp A:
   ```

   Ezután ismételten adjuk ki az iménti parancsot minden egyes használni kívánt .flp állományra, azonban elõtte mindig tegyünk be egy újabb floppyt, és a ráírt image-ek neveivel folyamatosan címkézzük fel a lemezeket. A megadott parancsot természetesen mindig írjuk át a konkrét .flp állományok tényleges elérési útvonalainak megfelelõen. Ha nincs CD-nk, akkor az `fdimage` programot az FreeBSD FTP oldalán található tools könyvtárból is letölthetjük.

   Amikor a lemezeket egy UNIX® rendszeren készítenénk el (például egy másik FreeBSD rendszeren), akkor a dd(1) parancs is használható az image állományok közvetlen lemezreírásához. FreeBSD alatt így néz ki a paraméterezése:

   ```
   # dd if=boot.flp of=/dev/fd0
   ```

   FreeBSD-n a /dev/fd0 az elsõ hajlékonylemezes meghajtóra hivatkozik (tehát az A: betûjelû meghajtóra). Ennek megfelelõen a /dev/fd1 jelenti a B: meghajtót és így tovább. Más UNIX® változatok esetleg más neveket használhatnak a hajlékonylemezes meghajtók megnevezésére, ezért errõl érdemes ilyenkor tájékozódni az adott rendszerhez tartozó dokumentációban.

Most már készen állunk a FreeBSD telepítésére!

2.4. A telepítés megkezdése

Alapértelmezés szerint a telepítés egészen addig nem fog semmit sem írni a lemezekre, amíg a következõ üzenet fel nem bukkan:

```
Last Chance: Are you SURE you want continue the installation?

If you're running this on a disk with data you wish to save then WE
STRONGLY ENCOURAGE YOU TO MAKE PROPER BACKUPS before proceeding!

We can take no responsibility for lost disk contents!
```

A szöveg fordítása:

```
Utolsó esély: BIZTOSAN folytatni kívánja a telepítést?

Ha olyan lemezre szeretne telepíteni, amelyen fontos adatok
találhatóak, HATÁROZOTTAN JAVASOLJUK, hogy a továbblépés elõtt
KÉSZÍTSEN RÓLUK MEGBÍZHATÓ BIZTONSÁGI MÁSOLATOT!

Nem vállalunk semmilyen felelõsséget az elveszett adatokért!
```

A telepítõbõl tehát a fenti, végsõ figyelmeztetés elõtt bármikor ki lehet lépni anélkül, hogy a merevlemezünkön levõ adatokat veszélyeztetnénk. Ha úgy érezzük, hogy valamit véletlenül rosszul állítottunk volna be a telepítés során, ekkor még minden komolyabb kár okozása nélkül kikapcsolhatjuk a számítógépünket.

2.4.1. A rendszer indítása

Rendszerindítás i386™-on

1. Kezdjünk egy kikapcsolt számítógéppel.
2. Kapcsoljuk be a számítógépet. Az indulása során látnunk kell egy olyan opciót, amivel be tudunk lépni a rendszer beállításait tartalmazó menübe, avagy a BIOS-ba. Ezt többnyire a `F2`, `F10`, `Del` vagy a `Alt` + `S` lenyomásával érhetjük el. Ezek közül használjuk a képernyõn megjelenõ billentyûket. Elõfordulhat, hogy induláskor a számítógépünk semmilyen szöveget, csak egy képet mutat. Ilyenkor általában a `Esc` billentyû megnyomására eltûnik a kép és láthatóvá válnak a számunkra fontos üzenetek.
3. Miután beléptünk a menübe, keressük meg azt a beállítást, amely a rendszerindításhoz használt eszközt határozza meg. Ennek a neve sokszor "Boot Order" (rendszerindítási sorrend) vagy valami hozzá hasonló. Itt mindenféle eszköz felsorolását találjuk: `Floppy`,

`CDROM`, `First Hard Disk` (elsõ merevlemezes meghajtó) és így tovább.

Ha CD-rõl akarjuk a telepítést elindítani, akkor akkor a `CDROM` eszközt válasszuk. Ha bármilyen kétség merülne fel bennünk, keressük meg ezt a beállítást a számítógéphez és/vagy az alaplaphoz kapott kézikönyvben.

Igényeink szerint végezzük el a beállítást, majd mentsük el és lépjünk ki. Most indítsuk újra a számítógépet.

4. Ha a Készítsünk egy rendszerindító lemezt ban leírtak szerint rendszerindító pendrive-ot készítettünk, akkor bekapcsolás elõtt csatlakoztassuk a számítógéphez.

   Ha CD-rõl indítjuk a telepítést, akkor kapcsoljuk be a számítógépet és az elindulása után igyekezzünk minél hamarabb betenni a lemezt a meghajtóba.

   A FreeBSD 7.3 és az azt megelõzõ változatokban a Készítsünk egy rendszerindító lemezt ban leírtak szerint elõkészített floppy-ról is el tudjuk kezdeni a telepítést. Ezek egyike lesz az elsõ rendszerindító lemez, a boot.flp. Helyezzük ezt a lemezt a meghajtóba, és indítsuk el vele a számítógépet.

   Ha minden próbálkozásunk ellenére a számítógépünk a megszokott módon indul és a meglevõ operációs rendszert tölti be, akkor a következõkkel lehet a gond: .. A lemezeket nem raktuk be eléggé korán. Hagyjuk benn ezeket és próbáljuk meg ismét újraindítani a számítógépet. .. Nem állítottuk be jól a BIOS-t. Próbáljuk meg egészen addig újra végrehajtani az elõzõ lépést, amíg a megfelelõ beállítást el nem találjuk. .. A BIOS nem támogatja a kiválasztott eszközrõl történõ rendszerindítást.

5. A FreeBSD megkezdi az indulását. Ha CD-rõl indítjuk, akkor valami ehhez hasonlót fogunk látni (a konkrét verzióra vonatkozó adatokat itt most kihagytuk):

```
Booting from CD-Rom...
645MB medium detected
CD Loader 1.2

Building the boot loader arguments
Looking up /BOOT/LOADER... Found
Relocating the loader and the BTX
Starting the BTX loader

BTX loader 1.00 BTX version is 1.02
Console: internal video/keyboard
BIOS CD is cd0
BIOS drive C: is disk0
BIOS drive D: is disk1
BIOS 639kB/261056kB available memory

FreeBSD/i386 bootstrap loader, Revision 1.1

Loading /boot/defaults/loader.conf
```

```
/boot/kernel/kernel text=0x64daa0 data=0xa4e80+0xa9e40 syms
=[0x4+0x6cac0+0x4+0x88e9d]
\
```

Amikor floppyról indítjuk a rendszert, ehhez hasonlóval találkozhatunk (itt sem szerepelnek most verzióadatok):

```
Booting from Floppy...
Uncompressing ... done

BTX loader 1.00  BTX version is 1.01
Console: internal video/keyboard
BIOS drive A: is disk0
BIOS drive C: is disk1
BIOS 639kB/261120kB available memory

FreeBSD/i386 bootstrap loader, Revision 1.1

Loading /boot/defaults/loader.conf
/kernel text=0x277391 data=0x3268c+0x332a8 |

Insert disk labelled "Kernel floppy 1" and press any key...
```

Kövessük a képernyõn megjelenõ utasítást ("Helyezze be a "Kernel floppy 1" címkéjû lemezt és nyomjon meg egy billentyût..."), tehát vegyük ki a boot.flp image-hez tartozó lemezt és tegyük be helyette a kern1.flp image-hez tartozó lemezt, majd nyomjuk le az `Enter` billentyût. Várjuk meg amíg a rendszer megkezdi az indulást az elsõ lemezrõl, majd az utasításoknak megfelelõen folyamatosan tegyük be a soron következõ lemezeket.

6. Miután elindítottuk a rendszert CD-rõl, pendrive-ról vagy floppy-ról, a rendszerindítási folyamat be fogja hozni a FreeBSD rendszertöltõjének menüjét:

Ábra 1. FreeBSD rendszerbetöltõ menüje

Várjuk ki a tíz másodperces szünetet vagy egybõl nyomjuk le az `Enter` billentyût.

Rendszerindítás sparc64-en

A legtöbb sparc64 alapú rendszert úgy állították be, hogy automatikusan lemezrõl induljon. A FreeBSD telepítéséhez azonban hálózaton keresztül vagy CD-rõl kell indítanunk a rendszert, ezért módosítanunk kell a PROM (az OpenFirmware) beállításait.

Mindehhez indítsuk újra a rendszert és várjuk meg, amíg feltûnik a rendszerindító üzenet. A konkrét üzenet nagyban függ a számítógép típusától, azonban valami ilyesmi lesz:

```
Sun Blade 100 (UltraSPARC-IIe), Keyboard Present
Copyright 1998-2001 Sun Microsystems, Inc.  All rights reserved.
OpenBoot 4.2, 128 MB memory installed, Serial #51090132.
Ethernet address 0:3:ba:b:92:d4, Host ID: 830b92d4.
```

Amikor megpróbálja a rendszert elindítani a lemezrõl, a PROM parancssorának bekéréshez nyomjuk le a billentyûzeten az `L1` + `A` vagy a `Stop` + `A` billentyûket, esetleg a soros konzolon keresztül küldjünk egy `BREAK` parancsot (például a tip(1) vagy cu(1) man oldalakon szereplõ `~#` parancs használatával). Körülbelül így néz ki:

```
ok     ①
ok {0} ②
```

① Ez a fajta parancssor csak az egy processzorral rendelkezõ rendszereken jelenik meg.

② Ez a fajta parancssor többprocesszoros (SMP) rendszereken jelenik meg, ahol a szám az éppen

aktív processzor sorszámát jelöli.

Most helyezzük a CD-t a meghajtóba, és a PROM parancssorában pedig gépeljük be `boot cdrom` parancsot.

2.4.2. Az eszközkeresés eredményeinek vizsgálata

A képernyõn megjelenõ utolsó pár száz sor mindig eltárolódik, késõbb tetszõlegesen átvizsgálhatóak.

A puffer tartalmának átnézéséhez nyomjuk le a `Scroll Lock` billentyût, amivel bekapcsoljuk a korábban megjelent üzenetek közti visszalépést. Itt a nyílbillentyûk, vagy a `PageUp` és `PageDown` billentyûk használhatóak a kiírások átböngészéséhez. A `Scroll Lock` ismételt lenyomásával kiléphetünk ebbõl a módból.

Tegyük most mi is ezt, és nézzük az összes olyan üzenetet, amely a rendszermag indulása során keletkezett. A Példa az eszközkeresés eredményeireban látható szövegekhez hasonlóakat fogunk találni, habár ez a számítógépben található konkrét eszközöktõl függõen eltérõ lehet.

Példa az eszközkeresés eredményeire

```
avail memory = 253050880 (247120K bytes)
Preloaded elf kernel "kernel" at 0xc0817000.
Preloaded mfs_root "/mfsroot" at 0xc0817084.
md0: Preloaded image </mfsroot> 4423680 bytes at 0xc03ddcd4

md1: Malloc disk
Using $PIR table, 4 entries at 0xc00fde60
npx0: <math processor> on motherboard
npx0: INT 16 interface
pcib0: <Host to PCI bridge> on motherboard
pci0: <PCI bus> on pcib0
pcib1:<VIA 82C598MVP (Apollo MVP3) PCI-PCI (AGP) bridge> at device 1.0 on pci0
pci1: <PCI bus> on pcib1
pci1: <Matrox MGA G200 AGP graphics accelerator> at 0.0 irq 11
isab0: <VIA 82C586 PCI-ISA bridge> at device 7.0 on pci0
isa0: <iSA bus> on isab0
atapci0: <VIA 82C586 ATA33 controller> port 0xe000-0xe00f at device 7.1 on pci0
ata0: at 0x1f0 irq 14 on atapci0
ata1: at 0x170 irq 15 on atapci0
uhci0 <VIA 83C572 USB controller> port 0xe400-0xe41f irq 10 at device 7.2 on pci
0
usb0: <VIA 83572 USB controller> on uhci0
usb0: USB revision 1.0
uhub0: VIA UHCI root hub, class 9/0, rev 1.00/1.00, addr1
uhub0: 2 ports with 2 removable, self powered
pci0: <unknown card> (vendor=0x1106, dev=0x3040) at 7.3
dc0: <ADMtek AN985 10/100BaseTX> port 0xe800-0xe8ff mem 0xdb000000-0xeb0003ff ir
q 11 at device 8.0 on pci0
dc0: Ethernet address: 00:04:5a:74:6b:b5
miibus0: <MII bus> on dc0
```

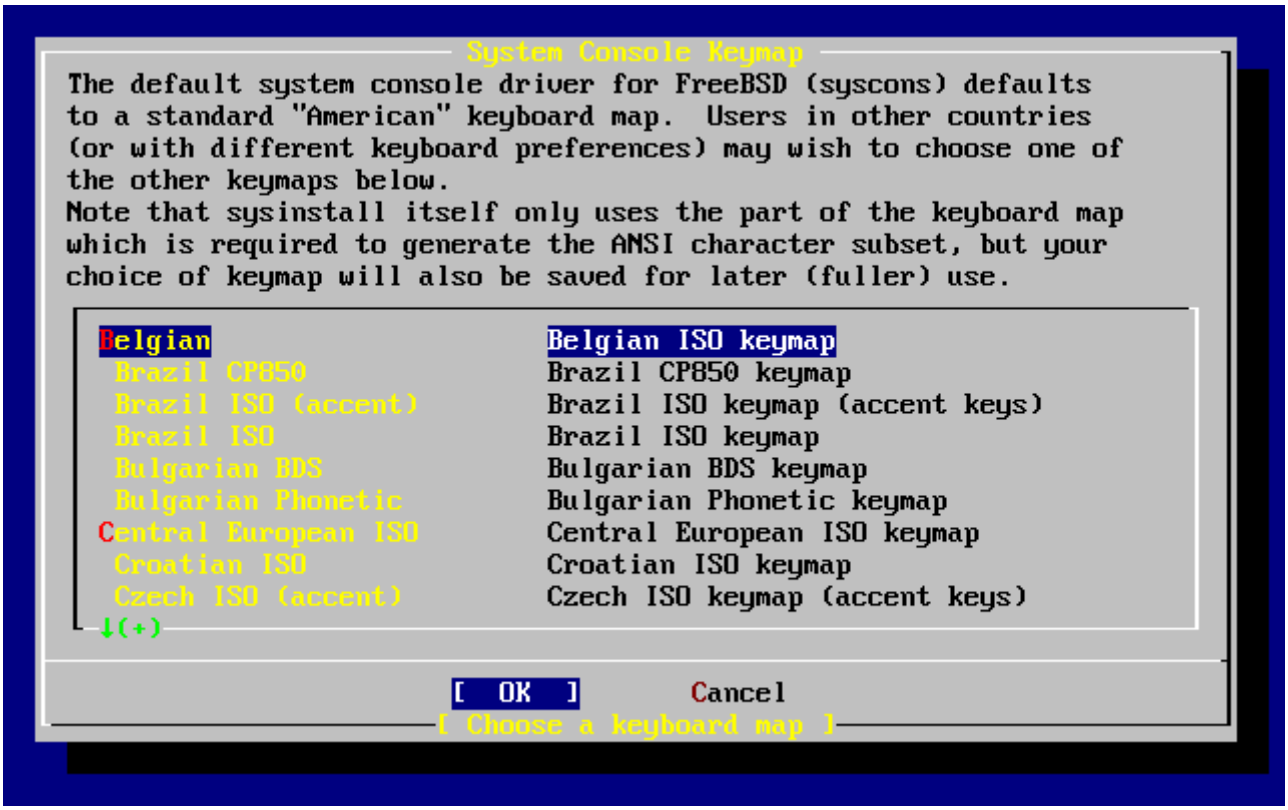
```
ukphy0: <Generic IEEE 802.3u media interface> on miibus0
ukphy0: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
ed0: <NE2000 PCI Ethernet (RealTek 8029)> port 0xec00-0xec1f irq 9 at device 10.
0 on pci0
ed0 address 52:54:05:de:73:1b, type NE2000 (16 bit)
isa0: too many dependant configs (8)
isa0: unexpected small tag 14
orm0: <Option ROM> at iomem 0xc0000-0xc7fff on isa0
fdc0: <NEC 72065B or clone> at port 0x3f0-0x3f5,0x3f7 irq 6 drq2 on isa0
fdc0: FIFO enabled, 8 bytes threshold
fd0: <1440-KB 3.5" drive> on fdc0 drive 0
atkbdc0: <Keyboard controller (i8042)> at port 0x60,0x64 on isa0
atkbd0: <AT Keyboard> flags 0x1 irq1 on atkbdc0
kbd0 at atkbd0
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: model Generic PS/@ mouse, device ID 0
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on isa0
sc0: <System console> at flags 0x100 on isa0
sc0: VGA <16 virtual consoles, flags=0x300>
sio0 at port 0x3f8-0x3ff irq 4 flags 0x10 on isa0
sio0: type 16550A
sio1 at port 0x2f8-0x2ff irq 3 on isa0
sio1: type 16550A
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
pppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode
ppc0: FIFO with 16/16/15 bytes threshold
plip0: <PLIP network interface> on ppbus0
ad0: 8063MB <IBM-DHEA-38451> [16383/16/63] at ata0-master UDMA33
acd0: CD-RW <LITE-ON LTR-1210B> at ata1-slave PIO4
Mounting root from ufs:/dev/md0c
/stand/sysinstall running as init on vty0
```

Figyelmesen olvassuk át az üzeneteket, és bizonyosodjuk meg róla, hogy a FreeBSD minden számunkra fontos eszközt felismert. Ha nem látunk egy eszközt, akkor azt valószínûleg nem találta meg. Egy saját rendszermag létrehozásával azonban fel tudunk ismertetni olyan eszközöket is, amelyek támogatása eredetileg nem szerepel a GENERIC rendszermagban. Ilyenek például a hangkártyák.

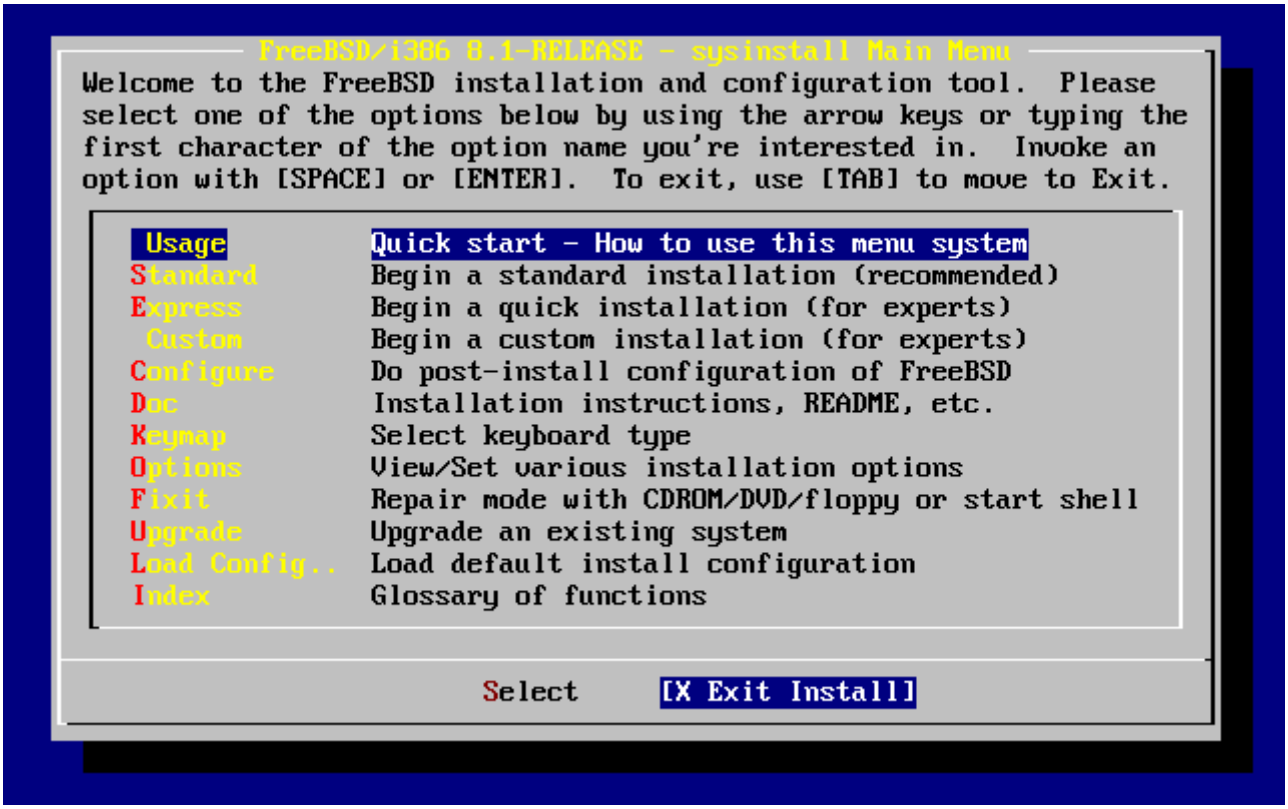
A FreeBSD 6.2 vagy késõbbi változataiban az eszközök felkutatása után a Az ország kiválasztásaban láthatóak következnek. Itt a nyílbillentyûk segítségével választhatjuk ki az országot (country), térséget (region) vagy csoportot (group). Az `Enter` lenyomása után pillanatok alatt beállítódik az országunk. Ha meg akarjuk ismételni az iménti beállítást, pillanatok alatt ki tudunk lépni a sysinstall programból.

Ábra 2. Az ország kiválasztása

Ha országként United States (Egyesült Államok) került beállításra, akkor a szabványos amerikai billentyûzet-kiosztás állítódik be. A többi ország esetében az alábbi menü jelenik meg. A kurzormozgató billentyûk segítségével ekkor keressük meg ki a számunkra megfelelõ kiosztást, és az `Enter` billentyû lenyomásával válasszuk ki.



Ábra 3. A billentyûzet típusának kiválasztása



Ábra 4. Kilépés a sysinstall programból

A telepítõprogram fõképernyõjén válasszuk ki a nyílbillentyûkkel az Exit Install ("Kilépés a telepítésbõl") menüpontot. Erre a következõ üzenet fog megjelenni:

```
                User Confirmation Requested
      Are you sure you wish to exit? The system will reboot

                        [ Yes ]    No
```

Az üzenet fordítása:

```
              Felhasználói megerősítés szükséges
      Valóban ki akar lépni? A rendszer ezt követően újra fog
      indulni

                        [ Igen ]    Nem
```

Ha a **[yes]** választ adjuk és a CD-t az újraindításkor is a meghajtóban hagyjuk, akkor a telepítõprogram még egyszer el fog indulni.

Ha floppyról indítottuk volna a rendszert, az újraindítás elõtt vegyük ki a boot.flp image-et tartalmazó lemezt.

2.5. A sysinstall bemutatása

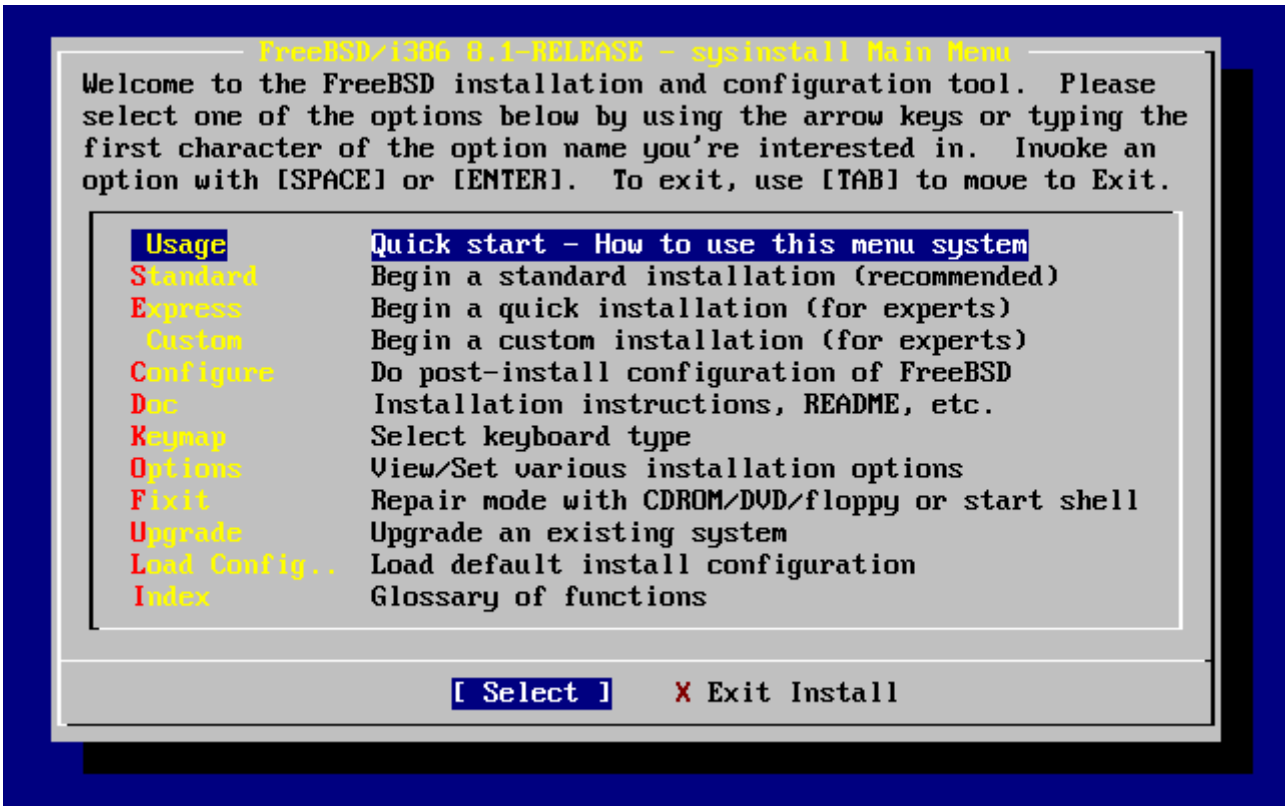
A sysinstall a FreeBSD Projekt által fejlesztett telepítõprogram. Konzol alapú, menükre és

képernyõkre oszlik, amelyeken a beállításokat és a telepítési folyamat irányítását tudjuk elvégezni.

A sysinstall menürendszerét több más billentyû mellett legfõképpen a nyílbillentyûkkel, az `Enter`, `Tab` és a `Szóköz` billentyûkkel kezelhetjük. Ezek és az általuk elvégezhetõ feladatok részletes leírása a sysinstall használatáról szóló információk között található.

Ennek megtekintéséhez elõször gyõzõdjünk meg róla, hogy a A "Usage" kiválasztása a sysinstall fõmenüjében által illusztrált helyzetnek megfelelõen kiválasztottuk a Usage ("Használat") menüpontot és a **[Select]** ("Kiválaszt") feliratú gombon állunk, majd nyomjuk le az `Enter` billentyût.

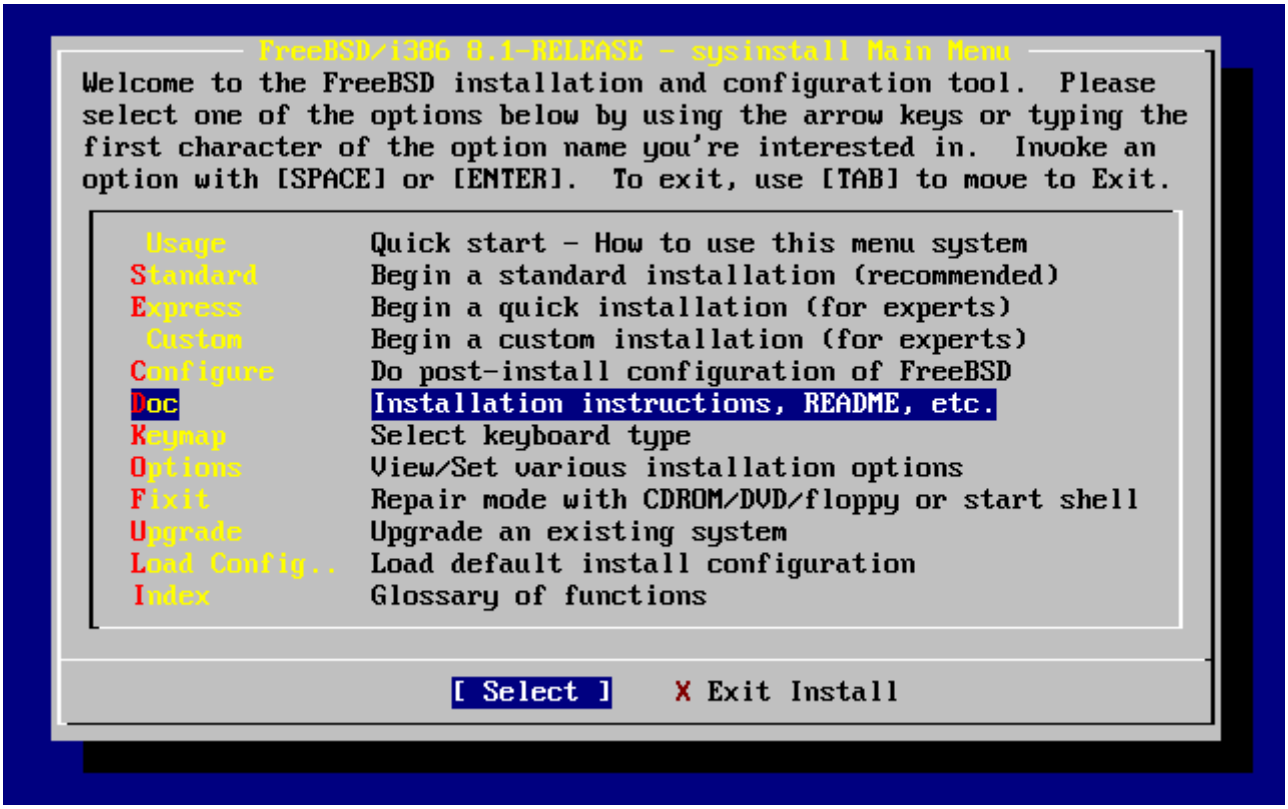
Ezt követõen megjelenik a menürendszer használatát bemutató leírás. Miután végigolvastuk, a fõmenübe az `Enter` billentyû lenyomásával tudunk visszajutni.



Ábra 5. A "Usage" kiválasztása a sysinstall fõmenüjében

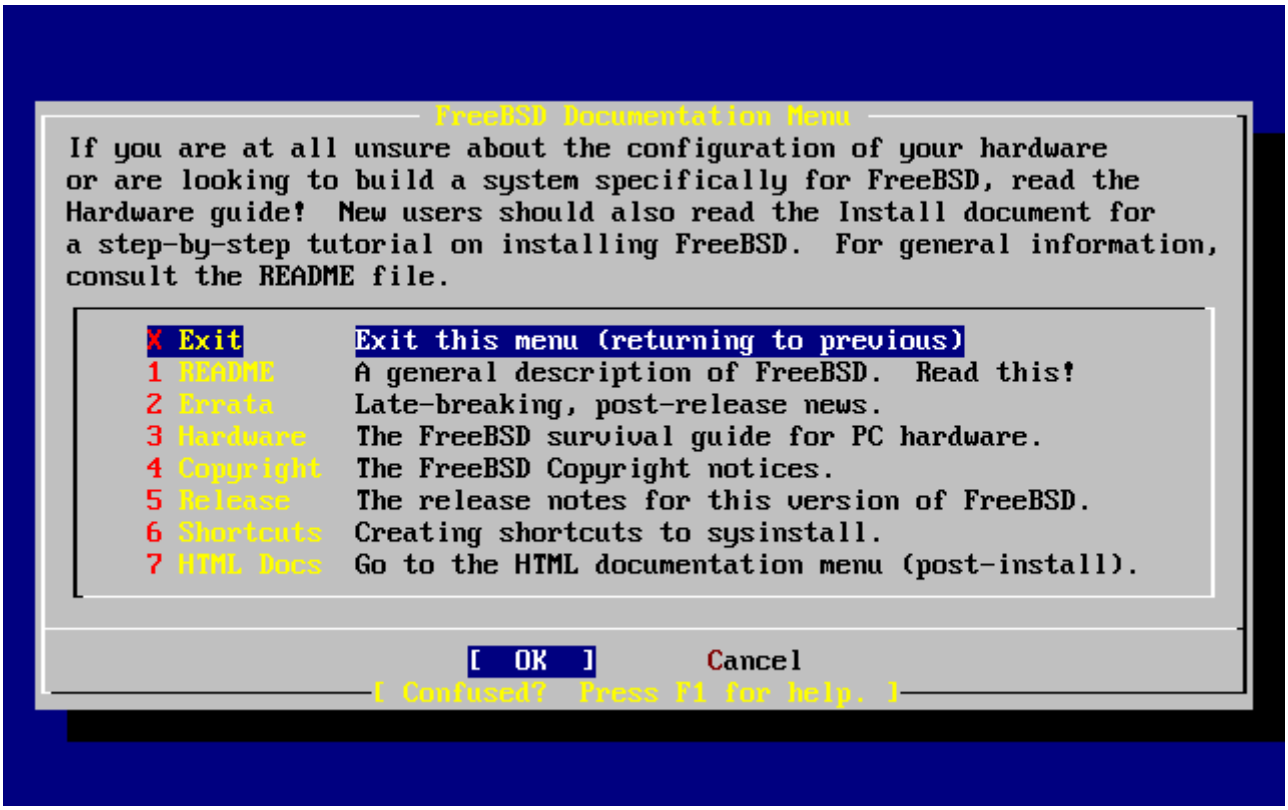
2.5.1. A dokumentációs menü kiválasztása

A fõmenüben a nyílbillentyûkkel válasszuk a Doc feliratú menüpontot és nyomjuk meg az `Enter` billentyût.



Ábra 6. A dokumentációs menü kiválasztása

Ezzel megjelenik a dokumentációs menü.



Ábra 7. A sysinstall dokumentációs menüje

Feltétlenül olvassuk el az itt található leírásokat.

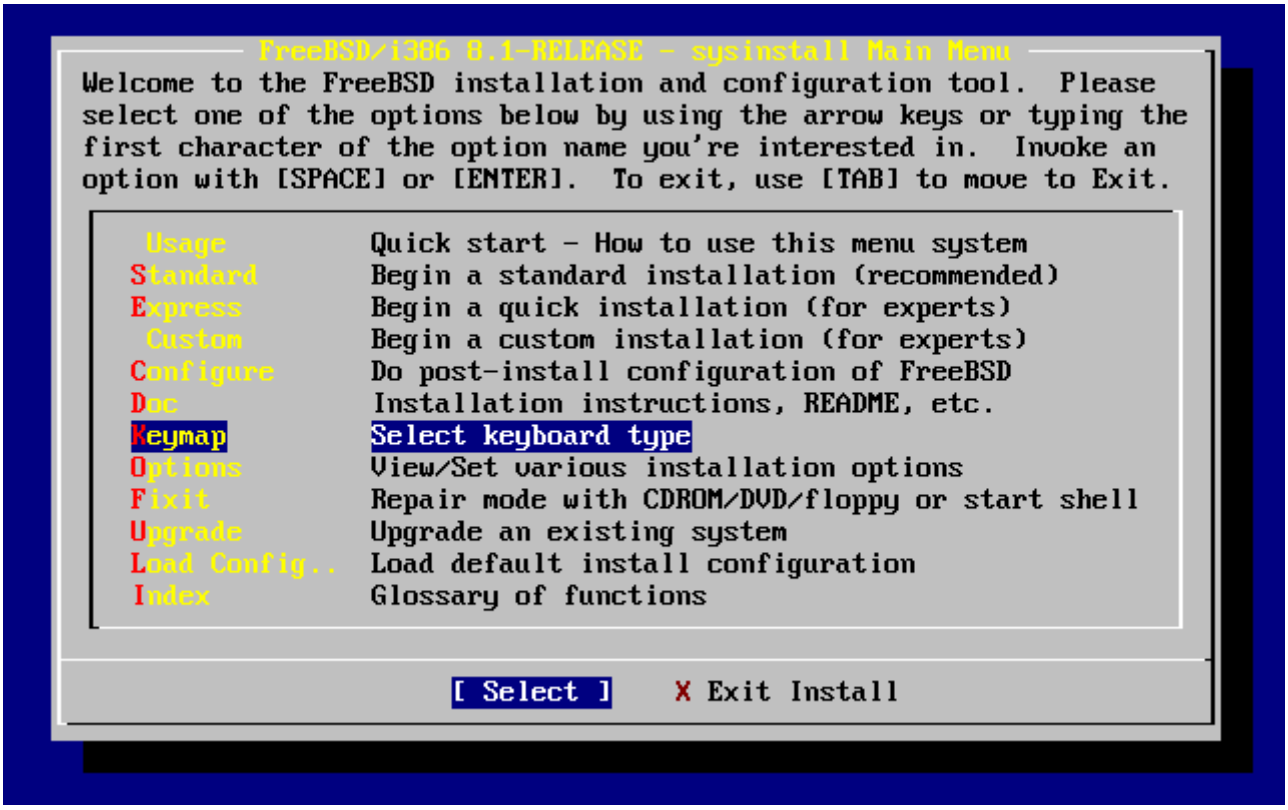
A dokumentumok elolvasásához elõször válasszunk közülük a nyílbillentyûkkel, majd nyomjuk meg az `Enter` billentyût. A dokumentum elolvasása után az `Enter` lenyomásával tudunk visszatérni a

dokumentációs menübe.

A dokumentációs menübõl a fõmenübe úgy tudunk kilépni, ha a nyílbillentyûkkel kiválasztjuk az Exit ("Kilépés") menüpontot és megnyomjuk az `Enter` billentyût.

2.5.2. A billentyûkiosztás menüjének kiválasztása

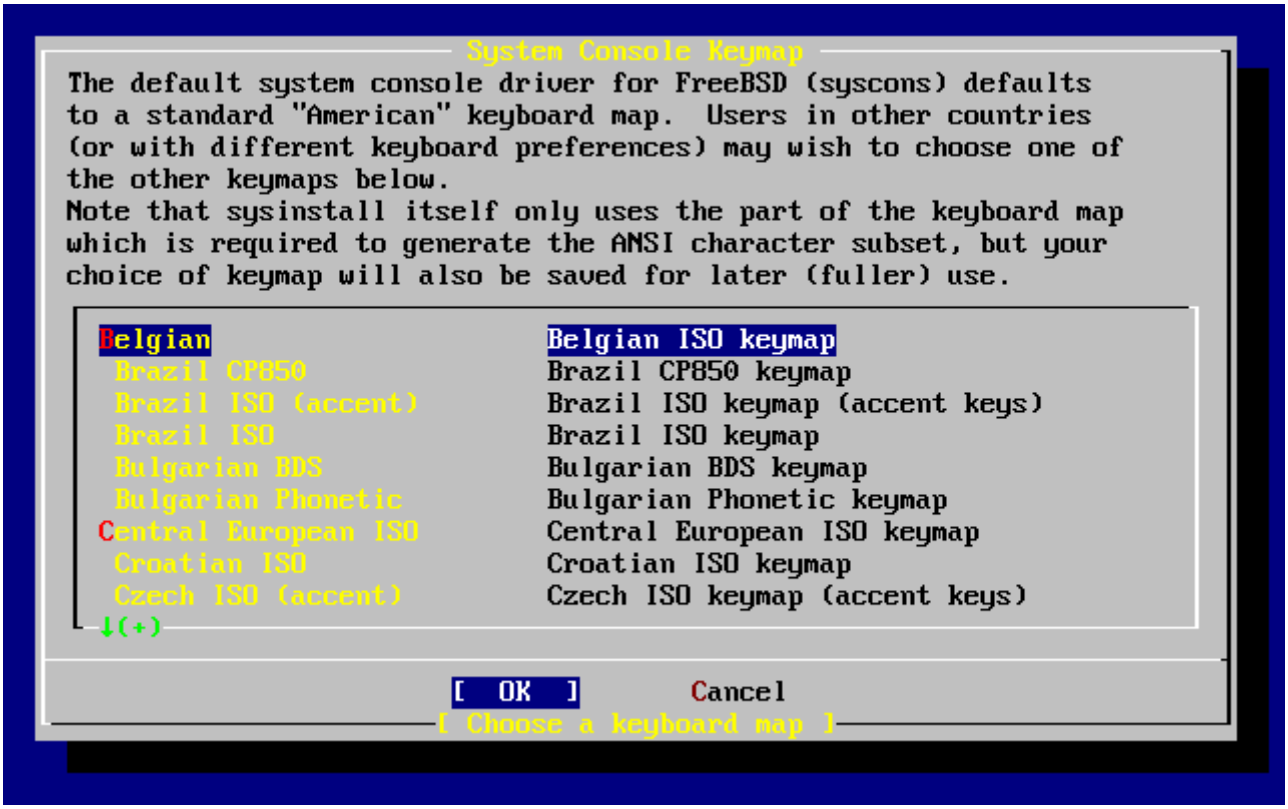
A billentyûzetkiosztás megváltoztatásához válasszuk ki a nyílbillentyûk segítségével a Keymap menüpontot a menübõl és nyomjuk meg az `Enter` billentyût. Erre természetesen csak akkor lesz szükségünk, ha nem szabványos vagy nem angol billentyûzetet használunk.



Ábra 8. A sysinstall fõmenüje

A különbözõ billentyûkiosztásoknak megfelelõ menüpontok a fel/le nyílak és a `Szóköz` billentyû segítségével választhatóak ki. A `Szóköz` ismételt lenyomásával töröljük a választásunkat. A befejezéshez válasszuk ki a nyilakkal a **[OK]** gombot és nyomjuk le az `Enter` billentyût.

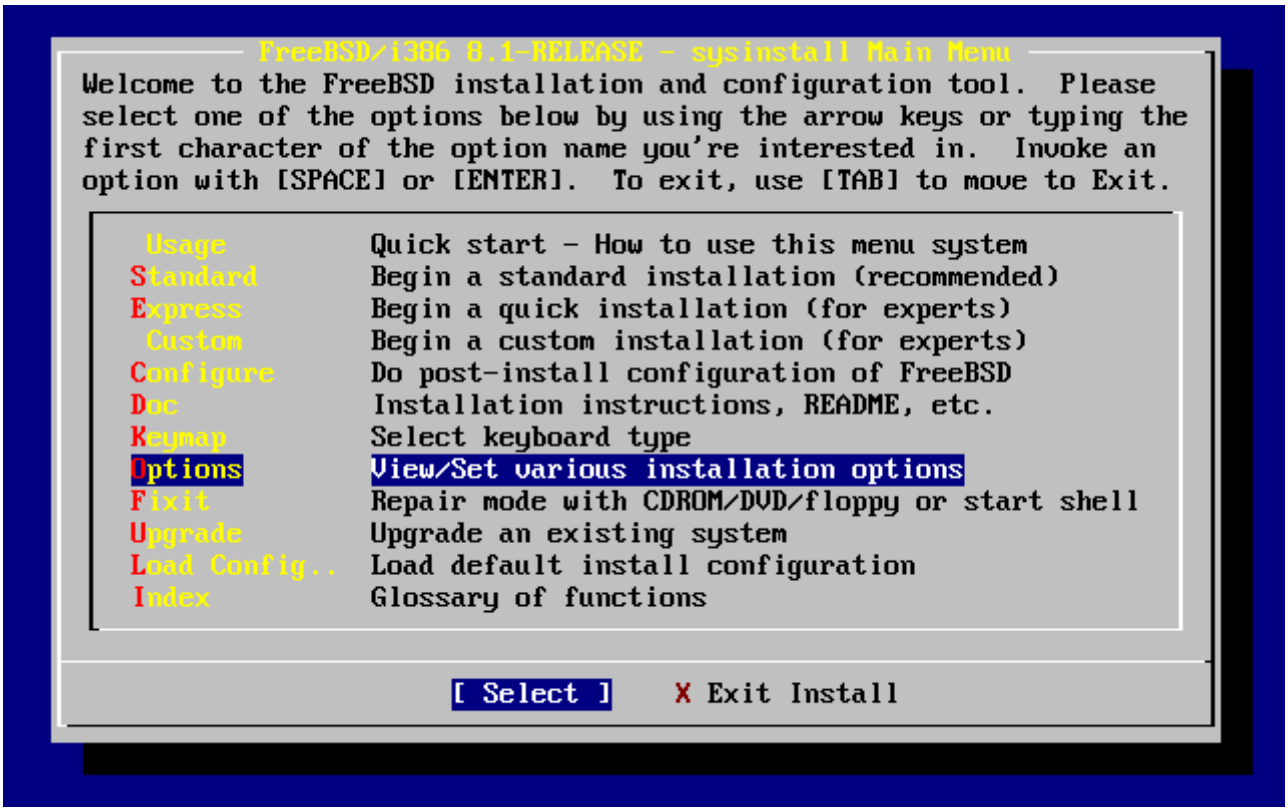
A mellékelt képen a lista egy része látható csupán. Ha a `Tab` billentyûvel a **[Cancel]** gombot választjuk, akkor az alapértelmezett billentyûkiosztást kapjuk és visszakerülünk a fõmenübe.



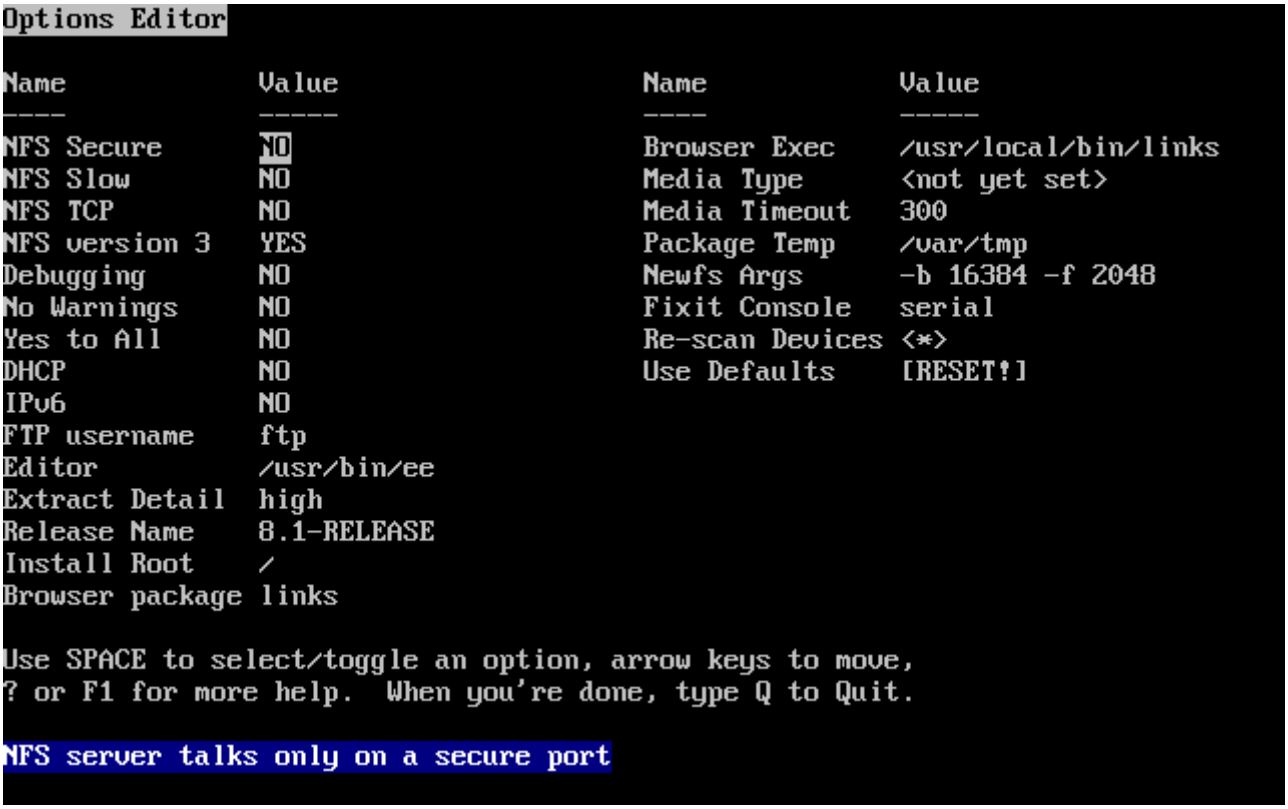
Ábra 9. A sysinstall billentyûkiosztást beállító menüje

2.5.3. A telepítés beállításai tartalmazó képernyõ

Válasszuk az Options ("Beállítások") menüpontot, majd nyomjuk le az `Enter` billentyût.



Ábra 10. A sysinstall fõmenüje



Ábra 11. A sysinstall beállításai

Az itt szereplõ alapértelmezett értékek a legtöbb felhasználó számára minden további nélkül megfelelnek, nem szükséges a megváltoztatásuk. A kiadás neve ("release name") mezõ értéke a telepítendõ verziótól függõen változhat.

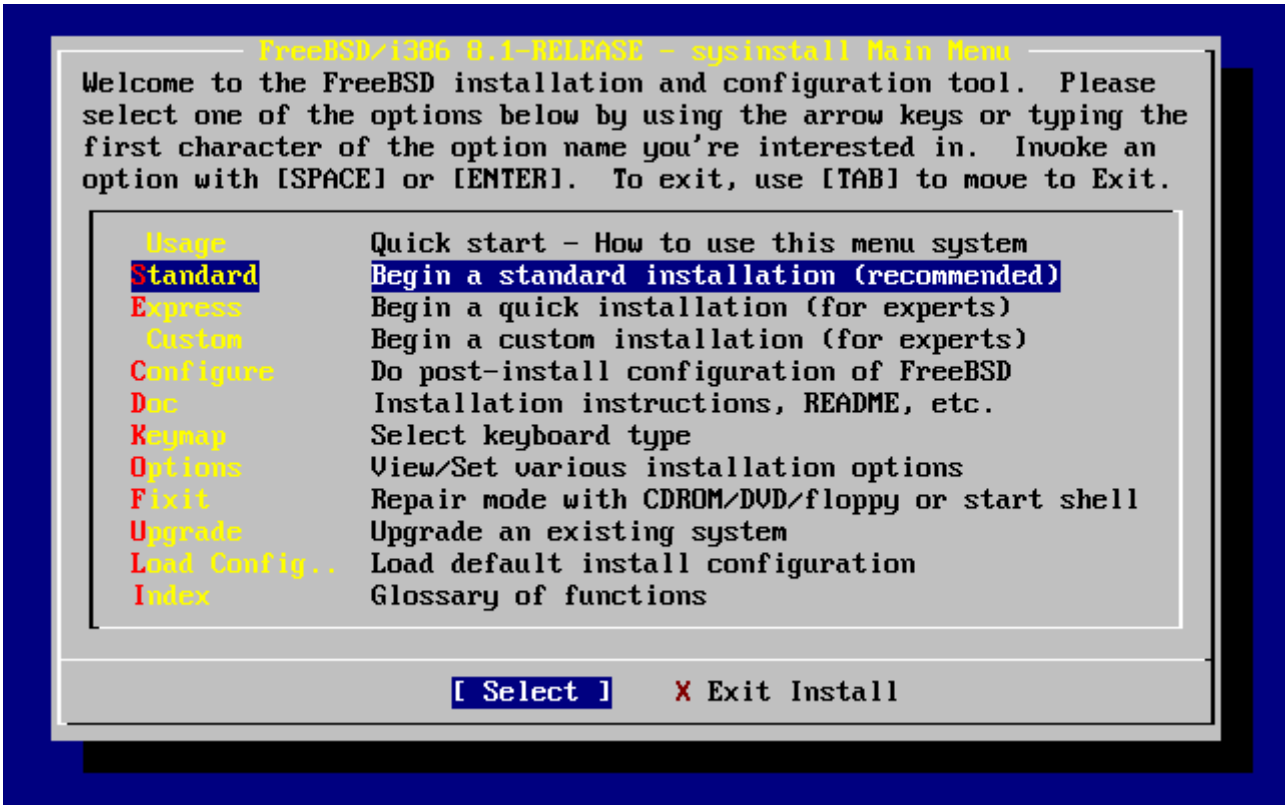
A kiválasztott mezõ rövid leírása a képernyõ alján, kékkel kiemelten jelenik meg. A Use Defaults ("Az alapértelmezések használata") beállítás az alapértelmezésére állítja vissza az összes értéket.

Az F1 lenyomásával elolvashatjuk a különbözõ beállításokhoz tartozó súgót.

A Q billentyûvel visszatérhetünk a fõmenübe.

2.5.4. Egy szabványos telepítés megkezdése

A Standard ("Szabványos") elnevezésû menüpont által felkínált telepítési módszer ajánlott a UNIX®-szal vagy a FreeBSD-vel most ismerkedõk számára. A telepítés megkezdéséhez a nyilakkal válasszuk ki a Standard menüpontot, majd nyomjuk meg az Enter billentyût.



Ábra 12. Egy szabványos telepítés megkezdése

2.6. Lemezterület lefoglalása

Elsõ feladatunk lemezterületet foglalni a FreeBSD számára, majd megcímkézni azt, hogy a sysinstall elõ tudja készíteni. Ehhez tisztában kell lennünk azzal, hogy a FreeBSD milyen formában is keresi az adatokat a lemezünkön.

2.6.1. A BIOS meghajtószámozása

Egy témára különösen tekintettel kell lennünk mielõtt telepítenénk és beállítanánk a FreeBSD-t a rendszerünkön, fõleg abban az esetben, ha több merevlemezünk is van.

Egy BIOS-függõ operációs rendszert, például MS-DOS®-t vagy Windows®-t futattó PC esetén a BIOS az operációs rendszer beleegyezésével képes elvonatkoztatni a lemezek megszokott sorrendjétõl. Ennek köszönhetõen a felhasználó nem csak az ún. "primary master" (elsõdleges master) merevlemezes meghajtótól tudja elindítani a rendszert. Ez kifejezetten kényelmes megoldás az olyan felhasználók számára, akik az elsõvel teljesen megegyezõ második merevlemez megvásárlásával kialakították a rendszerük egyszerû és egyben a legolcsóbb biztonsági mentését, amire a Ghost vagy XCOPY programokkal tudnak rendszeres másolatokat készíteni. Így, ha az elsõdleges meghajtó tönkremegy vagy vírus támadja meg, esetleg az operációs rendszer egy hiba miatt használhatatlanná teszi, akkor a BIOS-t utasíthatjuk a meghajtók logikai cseréjére és ezzel könnyen helyre tudjuk állítani. Olyan, mintha a ház felnyitása nélkül felcseréltük volna a lemezeket bekötõ kábeleket.

A SCSI-vezérlõkkel szerelt drágább rendszerek gyakran tartalmaznak olyan BIOS-bõvítéseket, amelyeken keresztül a SCSI-lemezek ugyanígy tetszõlegesen átrendezhetõek, egészen hét meghajtóig.

Az ilyen lehetõségek használatához szokott felhasználókat azonban könnyen csalódás érheti, amikor a FreeBSD nem az elvárásaiknak megfelelõen cselekszik. A FreeBSD ugyanis nem használja a BIOS-t és nem ismeri a "BIOS logikai meghajtókiosztását". Ez meghökkentõ eredményekre vezethet, fõleg akkor, amikor paramétereiket tekintve a meghajtók fizikailag teljesen megegyeznek és ráadásul egymás másolatait tartalmazzák.

A FreeBSD telepítése elõtt mindig állítsuk vissza a BIOS-ban a meghajtók eredeti sorrendjét, és a használatához hagyjuk is így ezt a beállítást. Ha valamiért mégis meg kellene cserélnünk a meghajtókat, akkor ezentúl válasszuk a nehezebb utat: nyissuk ki a gépházat és kössük át a kábeleket, tegyük át a jumpereket mi magunk.

Vili fogott egy öreg Winteles számítógépet, hogy készítsen belõle egy FreeBSD-s rendszert Frédinek. Vili ehhez beszerel egy SCSI-meghajtót, ami így nullás SCSI-egység lesz, majd telepíti rá a FreeBSD-t.

Frédi nekilát használni a rendszert, azonban pár nap elteltével tapasztalja, hogy az öregecske SCSI-meghajtó számos apróbb hibát jelez, és ezért szól Vilinek.

Néhány nappal késõbb Vili eldönti, ideje pontot tenni az ügy végére, ezért a raktárban levõ SCSI-lemezek köztül elhoz az eredetivel egy teljesen megegyezõt. Az elõzetes felületellenõrzés eredményei szerint a meghajtó tökéletesen mûködik, ezért Vili beszerelni ezt a meghajtót a négyes SCSI-egységként, majd lemásolja a nullás meghajtó tartalmát a négyesre. Miután beszerelte a tökéletesen üzemelõ új meghajtót, Vili úgy határoz, ideje megkezdeni a használatát, ezért beállítja a SCSI BIOS-át, hogy a rendszer a nullás helyett ezentúl a négyes egységrõl induljon. A FreeBSD elindul és mindenki örül.

Frédi ezután folytatja megszokott munkáját, majd Vili és Frédi úgy gondolják, itt az ideje az újabb izgalmaknak - frissítsünk a FreeBSD egy újabb változatára. Vili ekkor eltávolítja a nullás SCSI-egységet, mivel már egyébként is kezdett tönkremenni, és kicseréli egy másik teljesen azonos lemezes meghajtóra. Vili ezt követõen Frédi internetrõl letöltött varázslatos floppyjainak segítségével feltelepíti a FreeBSD új verzióját az új nullás SCSI-egységre. A telepítés minden gond nélkül lezajlik.

Frédi próbálgatja is a FreeBSD új változatát néhány napig, és számára ez elegendõ bizonyíték ahhoz, hogy a munkahelyén is használja. Ideje hát átmásolni a régi munkáit, ezért Frédi csatlakoztatja a (korábbi FreeBSD változat legfrissebb változatát tartalmazó) négyes SCSI-egységet. Frédin azonban hirtelen aggodalom tör ki, hiszen a négyes SCSI-egységen sehol sem találja munkája féltett eredményeit.

Hova tûntek azok a komisz adatok?

Amikor Vili másolatot készített az eredeti nullás SCSI-egységrõl a négyes SCSI-egységre, a négyes egység egy "új klón" lett. Amikor a rendszerindításhoz Vili átrendezte a meghajtókat a SCSI BIOS-ban, azzal csak magát csapta be, ugyanis a FreeBSD továbbra is a nullás SCSI-egységrõl indult el! A BIOS által kiválasztott meghajtóról az effajta beállítások hatására ugyan behozható a rendszerindító és -betöltõ programok egy része, de amikor a FreeBSD rendszermagja átveszi a vezérlést, a BIOS által meghatározott sorrendiség figyelmen kívül marad és a FreeBSD visszatér a meghajtók eredeti rendezéséhez. Tehát ebben az esetben a rendszer továbbra is az eredeti nullás SCSI-egységrõl folytatja a mûködést, és Frédi összes adata itt található, nem pedig a négyes SCSI-egységen. A négyes SCSI-egységrõl futó rendszer illuziója így mindössze az emberi elvárások

szüleménye.

Örömmel említjük meg, hogy egyetlen byte-nyi adat sem sérült meg vagy pusztult el a jelenség felfedezése során. A korábbi nullás SCSI-egységet még sikerült megmenteni a szemétdombról és Frédi összes munkája visszakerült (és Vili most már el tud számolni nulláig).

Habár a tanmesénkben SCSI-meghajtókról esett szó, ugyanez fennáll az IDE-meghajtókra is.

2.6.2. Slice-ok létrehozása az FDisk használatával

Itt még semmilyen változtatás nem kerül lemezre. Ha úgy érezzük, hogy valamit rosszul csináltunk és újra el akarjuk kezdeni a telepítést, a menük segítségével büntetlenül távozhatunk a sysinstallból és újra próbálkozhatunk, vagy az `U` billentyû lenyomásával aktiválhatjuk az Undo ("Visszacsinál") funkciót. Ha véletlenül összezavarodtunk volna és nem találunk kilépési lehetõséget, akkor bármikor ki tudjuk kapcsolni a számítógépet.

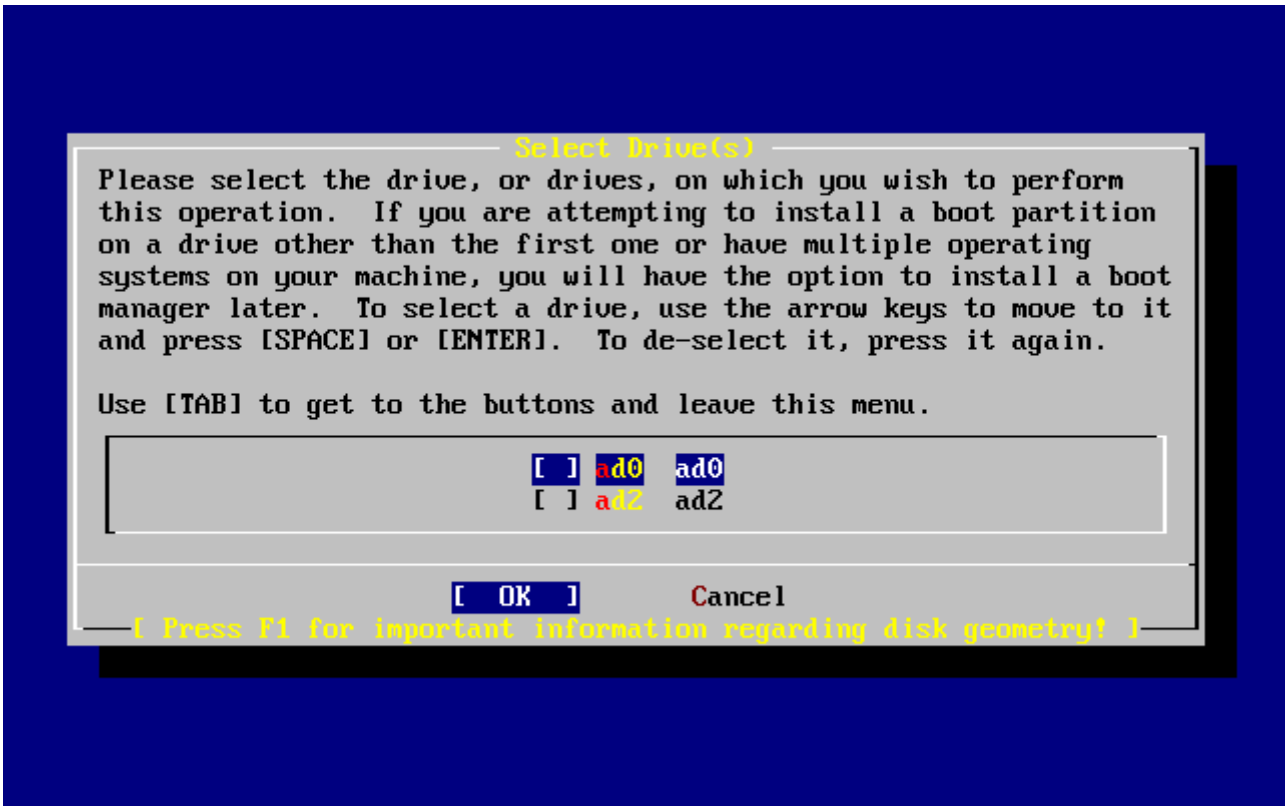
A sysinstallban a szabványos telepítés megkezdésekor az alábbi üzenet jelenik meg:

```
                                Message
In the next menu, you will need to set up a DOS-style ("fdisk")
partitioning scheme for your hard disk. If you simply wish to devote
all disk space to FreeBSD (overwriting anything else that might be on
the disk(s) selected) then use the (A)ll command to select the default
partitioning scheme followed by a (Q)uit. If you wish to allocate only
free space to FreeBSD, move to a partition marked "unused" and use the
(C)reate command.
                              [  OK  ]

                    [ Press enter or space ]
```

Az üzenet fordítása:

```
                                Üzenet
A most következő menüben össze kell állítanunk a merevlemezünk
DOS-szerű ("fdiskes") partícióit. Amennyiben egyszerűen csak át
akarjuk adni az összes lemezterületet a FreeBSD számára (ezzel
felülírva mindent, ami a kiválasztott lemezeken található), akkor
az alapértelmezett partíció-kiosztás kiválasztásához használjuk az
(A)ll (Mind), majd utána a (Q)uit (Kilépés) parancsokat. Ha viszont
csak az éppen szabad területet szánjuk a FreeBSD-nek, lépjünk egy
"unused" ("üres") feliratú partícióra és használjuk a (C)reate
(Létrehozás) parancsot.

                              [  OK  ]

           [ Nyomja le az Enter vagy a Szóköz billentyűt ]
```

Az utasításnak megfelelõen nyomjuk le az `Enter` billentyût. Ezután a rendszermag által az eszközök felkutatása során megtalált összes merevlemezes meghajtót láthatjuk. A A meghajtó kiválasztása az FDisk számára egy két IDE-lemezzel rendelkezõ rendszert mutat be, amelyeknek nevei rendre ad0 és ad2.



Ábra 13. A meghajtó kiválasztása az FDisk számára

Feltûnhet, hogy itt nem szerepel az ad1. Vajon miért maradt ki?

Képzeljük el, mi történne, ha két IDE-csatolós merevlemezünk lenne: az egyik az elsõ IDE-vezérlõn, a másik pedig a második IDE-vezérlõn lenne master. Ha a FreeBSD a megtalálásuk szerint ad0 és ad1 nevekkel számozná ezeket, attól még minden remekül mûködhetne.

Ha azonban beszerelnénk egy harmadik lemezt, például egy slave eszközt kapcsolnánk az elsõ IDE-vezérlõre, akkor már ez lenne a ad1, és ennek megfelelõen a korábban ad1 megnevezésû meghajtó pedig az ad2. Mivel az állományrendszerek felkutatására általában az eszközneveket (mint amilyen a ad1s1a) használják, ezért ilyenkor azt tapasztalhatnánk, hogy bizonyos állományrendszerek helytelenül jelennek meg, ezért meg kell változtatnunk a FreeBSD ezeket érintõ beállításait.

A probléma megoldására a rendszermag beállítható úgy, hogy az IDE-lemezeket a kapcsolódásuk szerint azonosítsa, ne pedig a megtalálásuk sorrendje szerint. Ezzel a kialakítással a második IDE-vezérlõn található master lemez *mindig* az ad2 eszköz lesz, tehát még olyankor is, amikor egyáltalán nincs a rendszerünkben ad0 vagy ad1 eszköz.

Ez a beállítás alapértelmezés a FreeBSD rendszermagjában, és ez magyarázza, hogy az iménti ábra miért csak ad0 és ad2 eszközöket mutat. Tehát a képen szereplõ számítógép mind a két IDE-vezérlõjének master csatornáján található egy-egy IDE-lemez, a slave csatornákon pedig nincs egy sem.

Itt válasszuk ki azt a lemezt, amelyre a FreeBSD-t telepíteni kívánjuk, majd nyomjuk meg a **[OK]**

gombot. Erre az Átlagos Fdisk partíciók szerkesztés elõtt által bemutatott képernyõvel elindul az FDisk.

Az FDisk képernyõje három részre osztható.

Az elsõ részben, amely a képernyõ felsõ két sorát foglalja össze, láthatjuk az éppen kiválasztott lemez adatait: a FreeBSD szerinti nevét, a paramétereit és az összméretét.

A második részben láthatjuk a lemezen megtalálható slice-okat: hol kezdõdnek (Offset) és hol érnek véget (End); mekkorák (Size); a FreeBSD milyen névvel hivatkozik rájuk (Name); milyen leírás (Description) és altípus (Subtype) tartozik hozzájuk. A példában két kicsi üres slice-ot láthatunk, ami a PC-k lemezkiosztására jellemzõ. Ezenkívül felfedezhetünk egy nagyobb méretû FAT típusú slice-ot is, amely az MS-DOS® / Windows® világban szinte minden bizonnyal a C: betûjelet viseli, valamint egy kiterjesztett slice-ot is, amely az MS-DOS® / Windows® számára további meghajtókat is tartalmazhat.

A harmadik részben az FDisk mûködtetésére használható parancsok láthatóak.

```
Disk name:      ad0                              FDISK Partition Editor
DISK Geometry:  16383 cyls/16 heads/63 sectors = 16514064 sectors (8063MB)

Offset       Size(ST)        End      Name  PType       Desc  Subtype    Flags

        0             63             62         -      6     unused        0
       63        4193217        4193279    ad0s1      2        fat       14    >
  4193280           1008        4194287         -      6     unused        0    >
  4194288       12319776       16514063    ad0s2      4   extended       15    >




The following commands are supported (in upper or lower case):

A = Use Entire Disk    G = set Drive Geometry   C = Create Slice   F = `DD' mode
D = Delete Slice       Z = Toggle Size Units    S = Set Bootable   I = Wizard m.
T = Change Type        U = Undo All Changes     Q = Finish


Use F1 or ? to get more help, arrow keys to select.
```

Ábra 14. Átlagos Fdisk partíciók szerkesztés elõtt

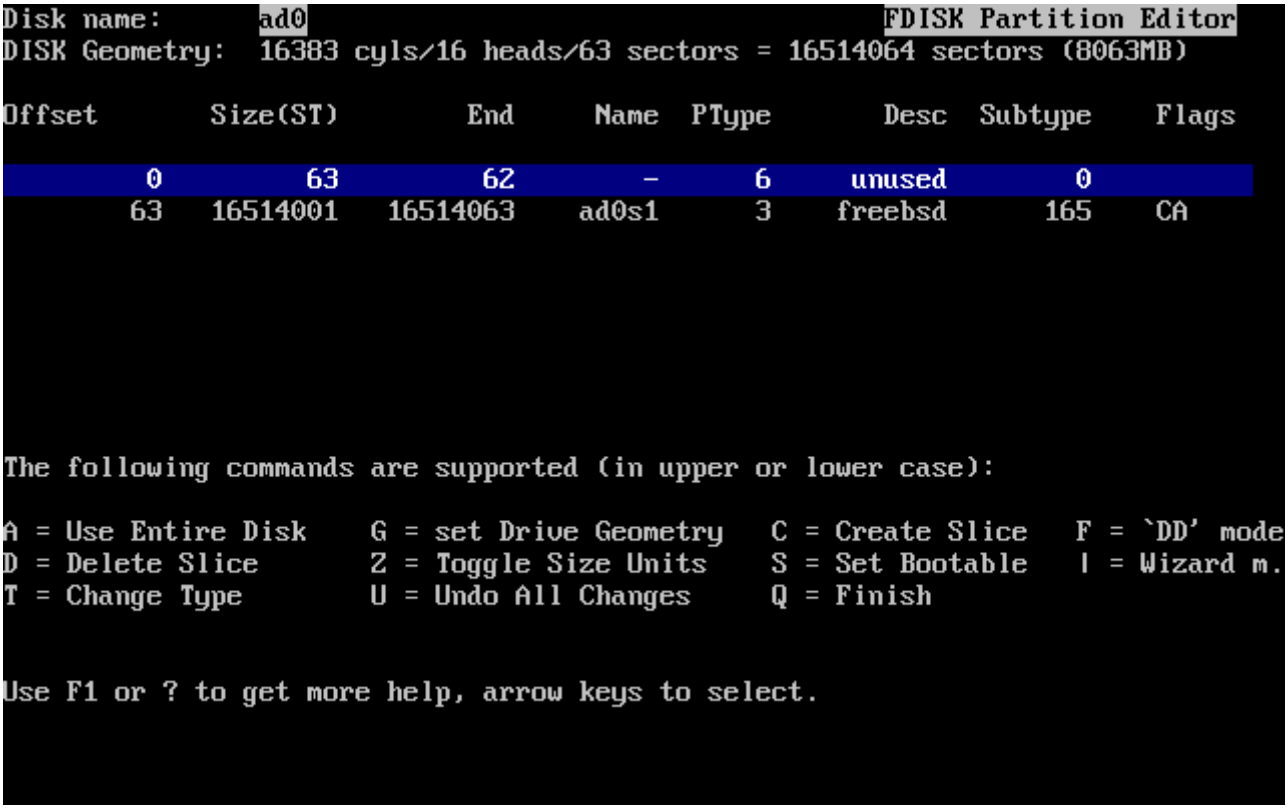
A most következõ teendõink attól függenek, hogy miként is akarjuk felosztani a lemezünket.

Ha az egész lemezt a FreeBSD használatára áldozzuk (és amikor majd megerõsítjük a sysinstall számára a továbblépést, a lemezen így minden más adat törlõdni fog), akkor nyomjuk le az `A` billentyût, amely megfelel a Use Entire Disk (Az egész lemez használata) menüpontnak. A létezõ slice-ok eltávolításra kerülnek és helyettük megjelenik egy `unused` (üres) jelzésû kis méretû terület (elvégre PC-rõl beszélünk), valamint egy nagyobb slice a FreeBSD számára. Ha így jártunk el, akkor válasszuk ki nyilakkal a frissen létrejött FreeBSD slice-ot és az `S` billentyû lenyomásával jelöljük be indíthatónak (bootable). A képernyõ ekkor a Particionálás az Fdisk "Using Entire Disk" funkciójával által mutatotthoz fog erõsen hasonlítani. A `Flags` (Beállítások) oszlopban láthatjuk az `A` jelzést,

amelybõl kiderül, hogy az adott slice *aktív*, tehát róla tud indulni a rendszer.

Ha a FreeBSD számára egy meglevõ slice törlésével szeretnénk helyet csinálni, akkor ehhez válasszuk ki nyílbillentyûkkel a használni kívánt slice-ot és nyomjuk le a `D` billentyût. Ezután nyomjuk le a `C` billentyût is, amire felbukkan a létrehozandó slice méretét kérdezõ ablak. Adjuk meg a számunkra megfelelõ méretet a számunkra megfelelõ formában, majd zárjuk le az `Enter` lenyomásával. Az ablakban szereplõ alapértelmezett érték a létrehozható lehetõ legnagyobb méretû slice-ot adja meg, ami vagy a legnagyobb összefüggõ üres terület, vagy pedig az egész merevlemez összterülete lehet.

Ha már korábban készítettünk elõ helyet a FreeBSD-nek (például egy PartitionMagic® vagy egy hozzá hasonló alkalmazás segítségével), akkor csak elegendõ az új slice létrehozásához megnyomnunk a `C` billentyût. Ekkor szintén megkérdezésre kerül a létrehozandó slice mérete.



Ábra 15. Particionálás az Fdisk "Using Entire Disk" funkciójával

Amikor befejeztük, nyomjuk le a `Q` billentyût. Ekkor a sysinstall elmenti a beállított értékeket, azonban a lemezre ekkor még nem kerülnek ki.

2.6.3. A rendszerválasztó telepítése

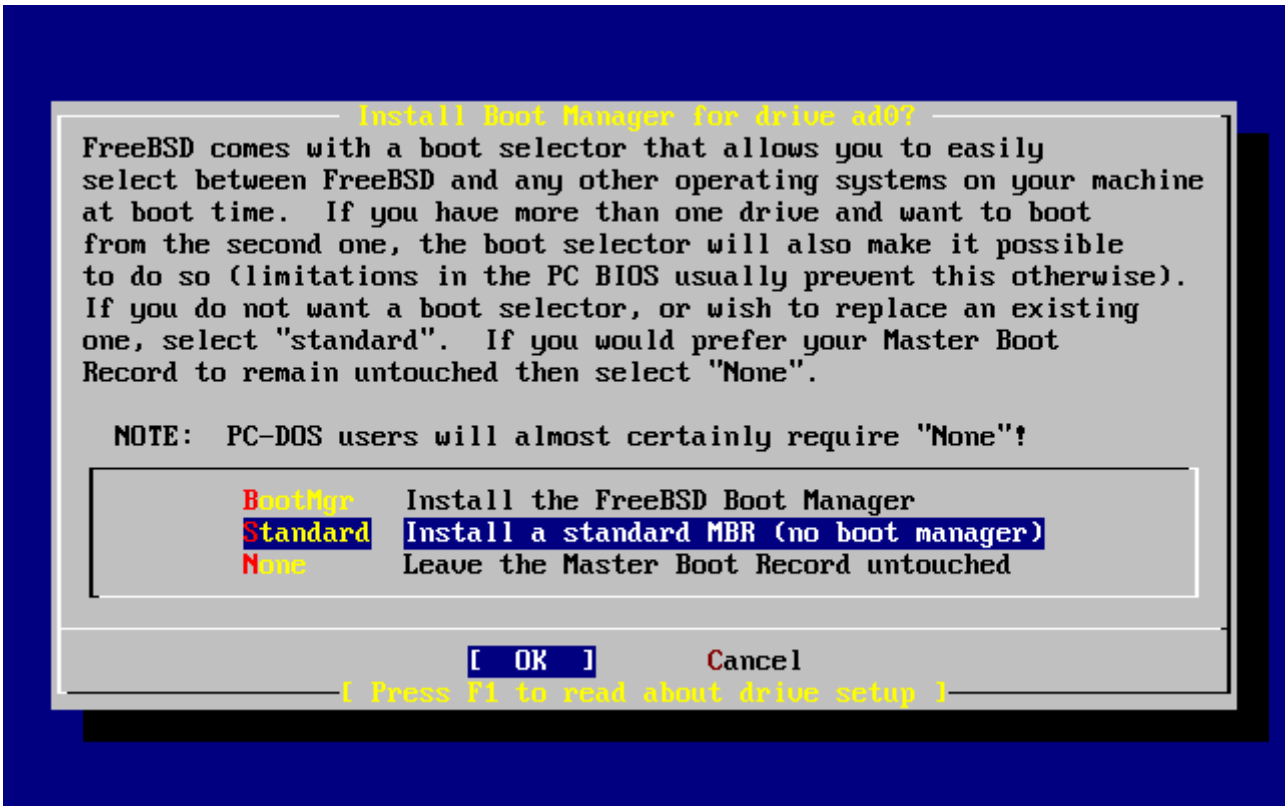
Mindezek után lehetõségünk nyílik telepíteni egy rendszerválasztót (boot manager). Általában véve akkor van szükségünk a FreeBSD rendszerválasztójának telepítésére, ha:

- Egynél több meghajtónk van, és közülük nem az elsõ meghajtóra telepítjük a FreeBSD-t.
- A FreeBSD-t ugyanazon a lemezen más operációs rendszerek mellé telepítjük, és szeretnénk választhatóvá tenni, hogy a számítógép indításakor a FreeBSD vagy a többi operációs rendszer induljon-e el.

Amennyiben a FreeBSD lesz az egyetlen operációs rendszer a gépünkön és az elsõ merevlemezes

meghajtóra telepítjük, akkor a Standard (Szabványos) rendszerválasztó tökéletesen megteszi. Ha viszont a FreeBSD indításához egy másik rendszerválasztót szeretnénk használni, válasszuk a None (Nincs) opciót.

Válasszunk, majd nyomjuk le az `Enter` billentyût!



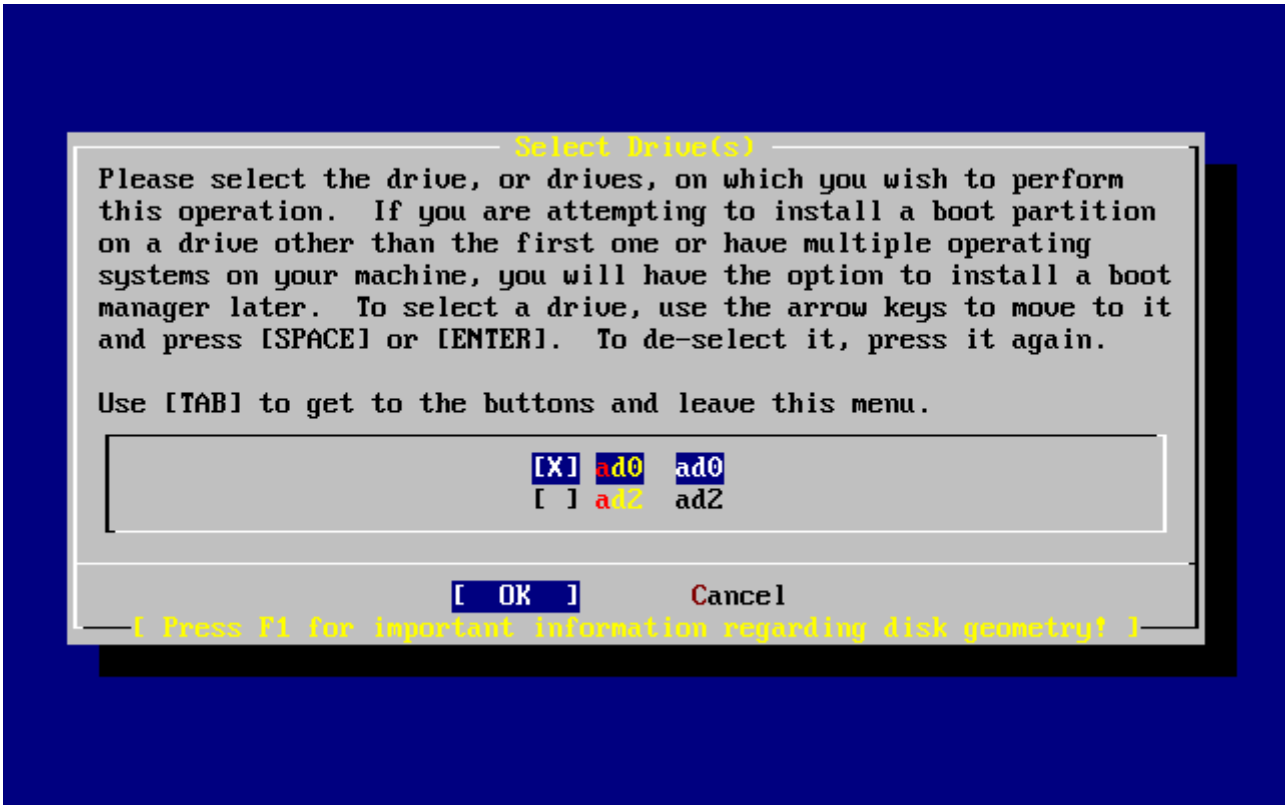
Ábra 16. A sysinstall rendszerválasztókat tartalmazó menüje

Az `F1` billentyû lenyomásán keresztül elérhetõ súgóképernyõn olvashatunk az egy merevlemezen több operációs rendszer használatával kapcsolatos problémákról.

2.6.4. Slice-ok létrehozása egy másik meghajtón

Ha egynél több meghajtónk van, a program a rendszerválasztó képernyõje után ismét visszatér a meghajtók kiválasztásához. Amennyiben a FreeBSD-t egy másik meghajtóra is telepíteni szeretnénk, itt válasszuk ki azt és ismételjük meg vele az imént az FDisk programmal végzett felosztási folyamatot.

Amikor a FreeBSD-t nem az elsõ meghajtóra telepítjük, akkor a FreeBSD rendszerválasztóját mind a két meghajtóra telepíteni kell.



Ábra 17. Kilépés a meghajtóválasztó menübõl

A `Tab` billentyûvel tudunk váltani a legutoljára kiválasztott meghajtó, a **[OK]** és a **[Cancel]** gombok között.

Az **[OK]** gombra álláshoz nyomjuk le egyszer a `Tab`ot, majd a telepítés folytatásához nyomjuk le az `Enter` billentyût.

2.6.5. Partíciók létrehozása a Disklabel segítségével

A következõ lépésként létre kell hoznunk partíciókat a frissen létrehozott slice-okban. Ne felejtsük el, hogy minden partíció rendelkezik egy `a`-tól `h`-ig terjedõ betûjellel, amelyek közül a `b`, `c` és `d` jelzésûeknek külön szerepe van, amire tekintettel kell lennünk.

Bizonyos alkalmazások kedvelnek egyes partíciókiosztási sémákat, különösen az egynél több lemezen elhelyezkedõ partíciókat. Azonban az elsõ FreeBSD telepítésünk során még nem annyira fontos koncentrálnunk a lemezünk hatékony felosztására. Sokkal inkább fontosabb, hogy elõször egyszerûen csak telepítsük a FreeBSD-t és tanuljuk meg a használatát. Amikor már jobban ismerni fogjuk az operációs rendszert, a partíciók kiosztásának megváltoztatásához mindig újra tudjuk telepíteni a FreeBSD-t.

Ebben a sémában négy partíció szerepel - egy a lapozóállománynak és három az állományrendszereknek.

Táblázat 2. Az elsõ lemez partícióinak kiosztása

Partíció	Állomá nyrends zer	Méret	Leírás
a	/	1 GB	Ez a rendszerindításhoz használt, más néven a gyökér állományrendszer (root filesystem). Minden további állományrendszer ehhez csatlakozik valahol. Ennek az állományrendszernek 1 GB méret elfogadható, mivel nem fogunk túlságosan sok adatot tárolni rajta, a FreeBSD telepítõje is csak nagyjából 128 MB adatot fog ide tenni. Az így fennmaradó lemezterület felhasználható átmeneti adatok tárolására, illetve a / könyvtárban helyet ad a FreeBSD késõbbi változatainak terjeszkedéséhez is.
b	-	RAM mérete x 2-3	A rendszer lapozóállománya a b partíción tárolódik. Itt a megfelelõ méret megválasztása egyfajta mûvészet, azonban minden esetben hasznosnak bizonyulhat, ha tudjuk, hogy méretnek mindig érdemes a fizikai avagy központi memória (RAM) méretének két, esetleg háromszorosát választani. Legyen mindig legalább 64 MB-nyi méretû lapozóállományunk, és ha 32 MB RAM-nál kevesebb van a számítógépünkben, akkor is legalább 64 MB-ra állítsuk be. Ha egynél több lemezünk van, mindegyikre rakhatunk lapozóállományt, ezzel a FreeBSD mindegyikõjüket fel tudja használni lapozásra, amivel pedig gyakorlatilag felgyorsítja a folyamatot. Ilyenkor számoljunk úgy, hogy elõször meghatározzuk a teljes lapozóállomány méretét (például 128 MB), majd ezt elosztjuk a rendelkezésünkre álló lemezek számával (például kettõ). Ebbõl kiszámítható az egyes lemezeken elhelyezendõ lapozóállomány mérete, ami most a példánk szerint 64 MB lesz.
e	/var	512 MB-tl 4096 MB-ig	A /var könyvtár foglalja magában az állandó változó naplóállományokat, valamint a többi, adminisztrációhoz használt állományt. Ezek többsége a FreeBSD mindennapos mûködése közben folyamatosan íródnak vagy olvasódnak. Ha ezeket az állományokat egy külön állományrendszerre rakjuk, akkor ezzel segítünk a FreeBSD-nek optimalizálni az ilyen állományok elérését anélkül, hogy ez hatással lenne a többi, más hozzáférési gyakorisággal bíró állományra.
f	/usr	A lemez többi része (legaláb b 8 GB)	Az összes többi állomány többnyire a /usr könyvtárban és annak alkönyvtáraiban helyezkedik el.

Az imént megadott értékeket csak példaként adtuk meg és csak a tapasztalt felhasználók számára ajánljuk. A többi felhasználónak inkább a partíciók automatikus kiosztását javasoljuk a FreeBSD partíciószerkesztõjében található Auto

> `Defaults` opció használatával.

Ha a FreeBSD-t egynél több lemezre telepítjük, akkor a korábban megadott többi slice-ban is létre kell hoznunk partíciókat. Ezt legegyszerûbben úgy tehetjük meg, ha minden lemezen létrehozunk két partíciót: egyet a lapozóállománynak, egyet pedig az állományrendszernek.

Táblázat 3. Több lemez partícióinak kiosztása

Partíció	Állományrendszer	Méret	Leírás
`b`	-	Lásd a leírást	Ahogy már korábban is említettük, szét tudjuk osztani a lapozóállományt a lemezek között. Habár az `a` partíció szabad, a hagyományok mégis azt diktálják, hogy a lapozáshoz használt terület maradjon a `b` partíción.
`e`	/disk*n*	A lemez többi része	A lemez fennmaradó része egyetlen nagy partícióval fedhetõ le. Ez az `e` partíció helyett lehetne minden további nélkül az `a` partíció, azonban a hagyományok szerint az `a` partíciónak a rendszer gyökér állományrendszerét (/) kell tartalmaznia. Nekünk ugyan nem kellene ezt a megszokást követnünk, azonban a sysinstall viszont így tesz, ezért ezzel a választással csak magunkkal teszünk jót. Az állományrendszer bárhová csatlakoztatható - ebben a példában a lemezeket rendre a /diskn könyvtárakhoz csatoltuk, ahol az *n* az adott lemez sorszáma. De itt természetesen más rendszert is követhetünk.

A partíciók elrendezésének kigondolása után most már létre is hozathatjuk ezeket a sysinstall segítségével. Ekkor a következõ üzenetet fogjuk látni:

```
                                Message
Now, you need to create BSD partitions inside of the fdisk
partition(s) just created. If you have a reasonable amount of disk
space (1GMB or more) and don't have any special requirements, simply
use the (A)uto command to allocate space automatically. If you have
more specific needs or just don't care for the layout chosen by
(A)uto, press F1 for more information on manual layout.

                                [  OK  ]
                          [ Press enter or space ]
```

Az üzenet fordítása:

```
                                Üzenet
Most létre kell hoznunk az fdiskkel nemrég elkészített partíciókban a
BSD-s partíciókat. Ha van hozzá elegendő helyünk (1G vagy több) és
nincs semmilyen különleges elvárásunk, akkor egyszerűen csak osszuk
fel automatikusan az (A)uto paranccsal. Amennyiben azonban ennél
```

```
többre lenne szükségünk, vagy csak nincs szükségünk az (A)uto által
felkínált sémára, az F1 lenyomására bővebb információkat is kaphatunk
a kézi kiosztás lehetőségeiről.

                                [  OK  ]
            [ Nyomja le az Enter vagy a Szóköz billentyűt ]
```

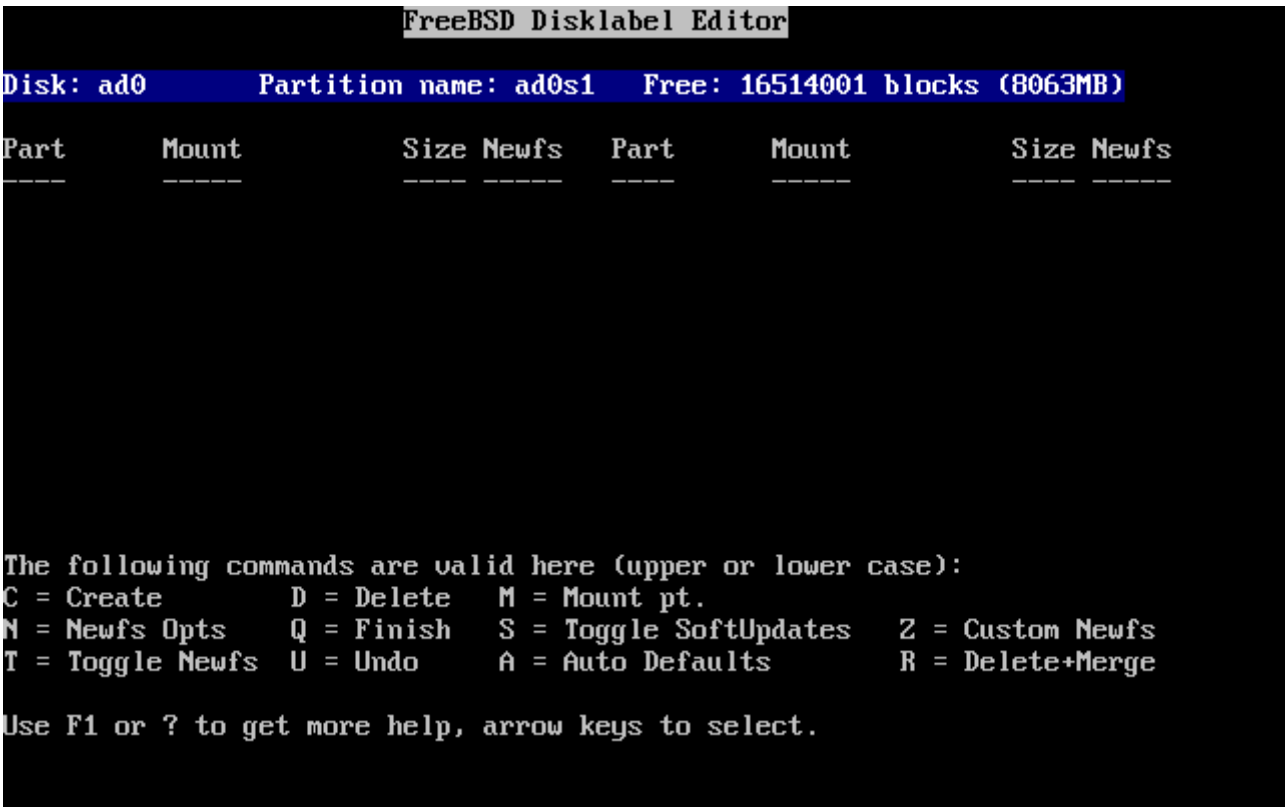
Nyomjuk le a `Enter` billentyût a FreeBSD partíciószerkesztõjének, avagy a Disklabel elindításához.

A A sysinstall Disklabel partíciószerkesztõje mutatja a Disklabel elsõ elindulásakor megjelenõ képet. A képernyõ három részre tagolható.

A felsõ pár sorban a jelenleg használt lemez nevét láthatjuk, valamint azt a slice-ot, ami az általunk létrehozott partíciókat tartalmazza (itt a Disklabel a `Partition name` megnevezéssel hivatkozik a slice-ra). A képernyõn továbbá láthatjuk a slice-ban levõ szabad helyet is, vagyis azt a helyet, amely ugyan a slice-hoz tartozik, viszont még nem rendeltünk hozzá partíciót.

A képernyõ közepén találhatóak az eddig már létrehozott partíciók, az általuk tartalmazott állományrendszerek, azok mérete és az állományrendszerek létrehozására vonatkozó különbözõ beállítások.

A képernyõ alsó harmadában a Disklabel programban használható billentyûk felsorolása szerepel.

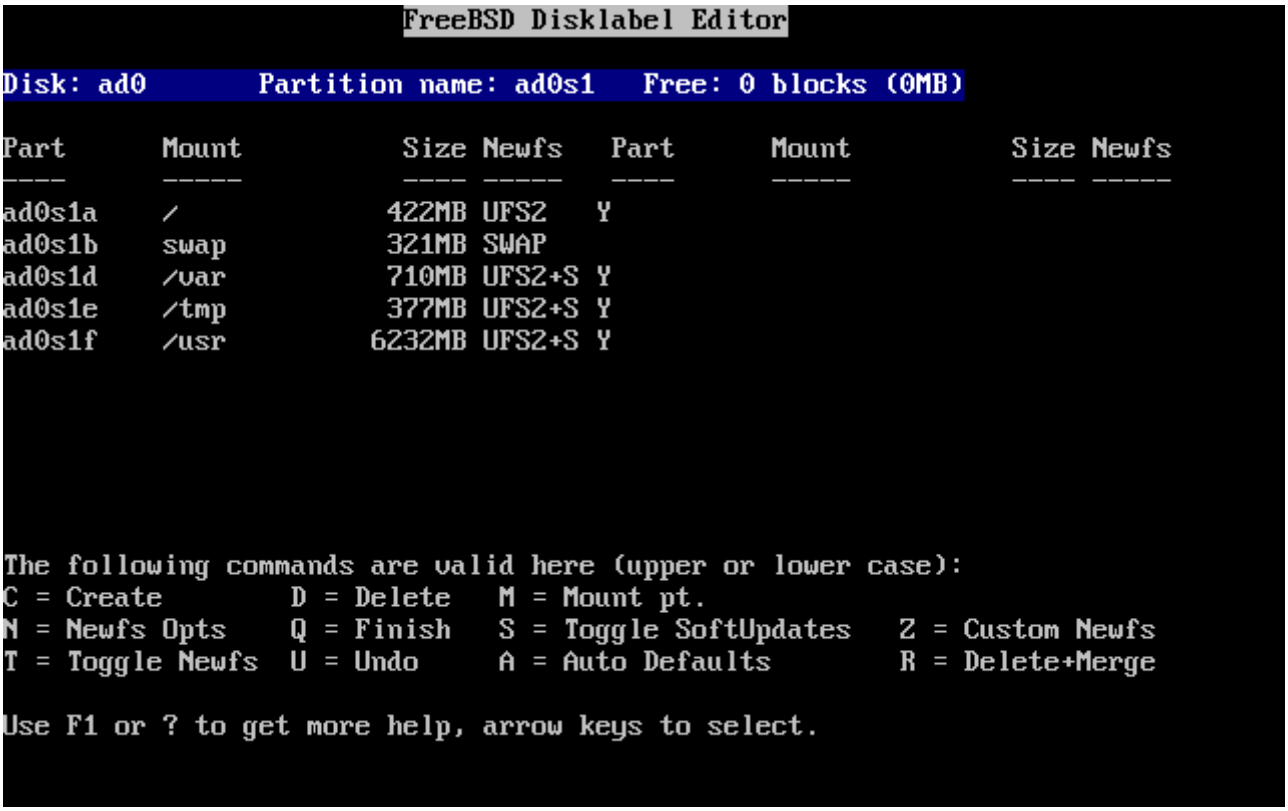


Ábra 18. A sysinstall Disklabel partíciószerkesztõje

A Disklabel képes magától partíciókat készíteni a nekik megfelelõ alapértelmezett méretekkel. A partíciók automatikus méretét egy belsõ partícióméretezõ algoritmus számítja ki a lemez összmérete alapján. Próbáljuk most mi is ezt ki, és nyomjuk le az `A` billentyût. Ekkor a A sysinstall Disklabel partíciószerkesztõje, alapértelmezett értékekkel szerint illusztráltaknak megfelelõ képernyõt tapasztalhatunk. A használt lemez méretétõl függõen az alapértelmezett értékek

megfelelõek lesznek vagy sem. Ez igazából nem számít, hiszen nem kell feltétlenül elfogadnunk az alapértelmezetten megállapított értékeket.

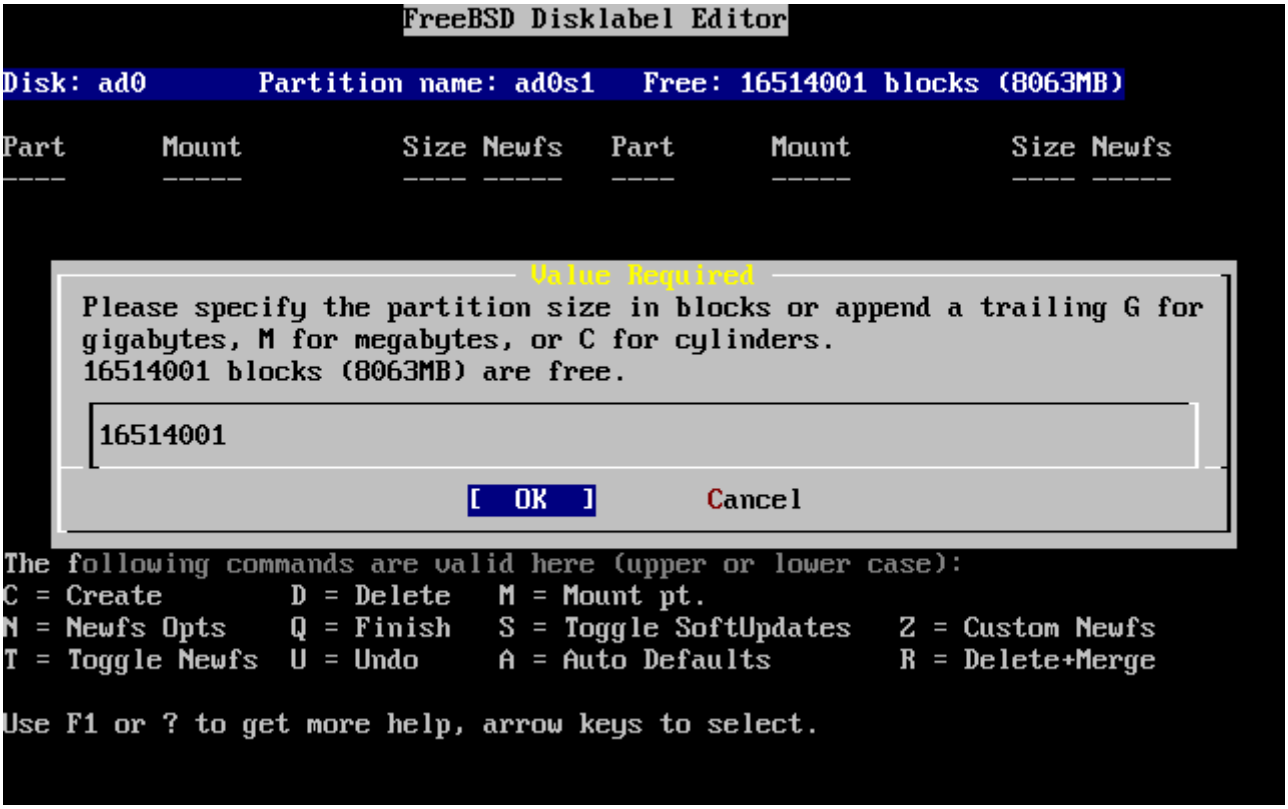
Az alapértelmezett partícionálási sémában a /tmp könyvtár nem a / könyvtár része lesz, hanem saját partíciót kapott. Ezzel igyekszünk elkerülni, hogy a / partíció átmenetileg tárolt állományokkal teljen be.



Ábra 19. A sysinstall Disklabel partíciószerkesztõje, alapértelmezett értékekkel

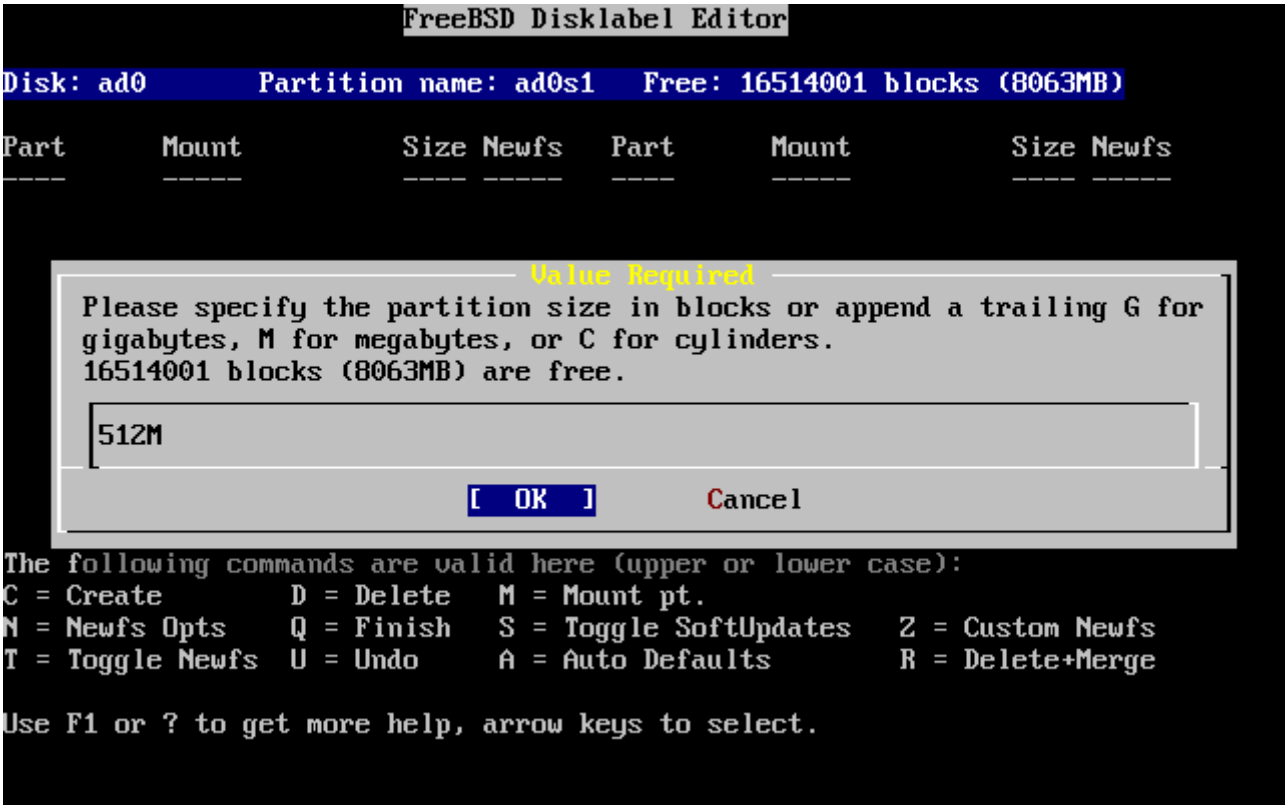
Ha nem az alapértelmezett partíciókat szeretnénk használni, és le akarjuk váltani ezeket a saját magunk által megadottakra, akkor a nyílbillentyûkkel válasszuk ki az elsõ partíciót és a törléséhez nyomjuk meg a `D` billentyût. Hasonlóan járjunk el az összes többi javasolt partíció törléséhez.

Az elsõ (`a`, vagyis a / könyvtárként, azaz a gyökérként csatolt) partíció elkészítéséhez elõször gyõzõdjünk arról, hogy a felsõ sorban a megfelelõ slice van kiválasztva, majd nyomjuk meg a `C` billentyût. Ekkor az új partíció méretét kérdezõ párbeszédablak jelenik meg (lásd: Szabad hely a gyökérpartíción). Itt a méret a lemez blokkjainak számában adható meg, amit viszont `M`-mel lezárva megabyte-ban, `G`-vel gigabyte-ban vagy `C`-vel cilinderben is kifejezhetünk.



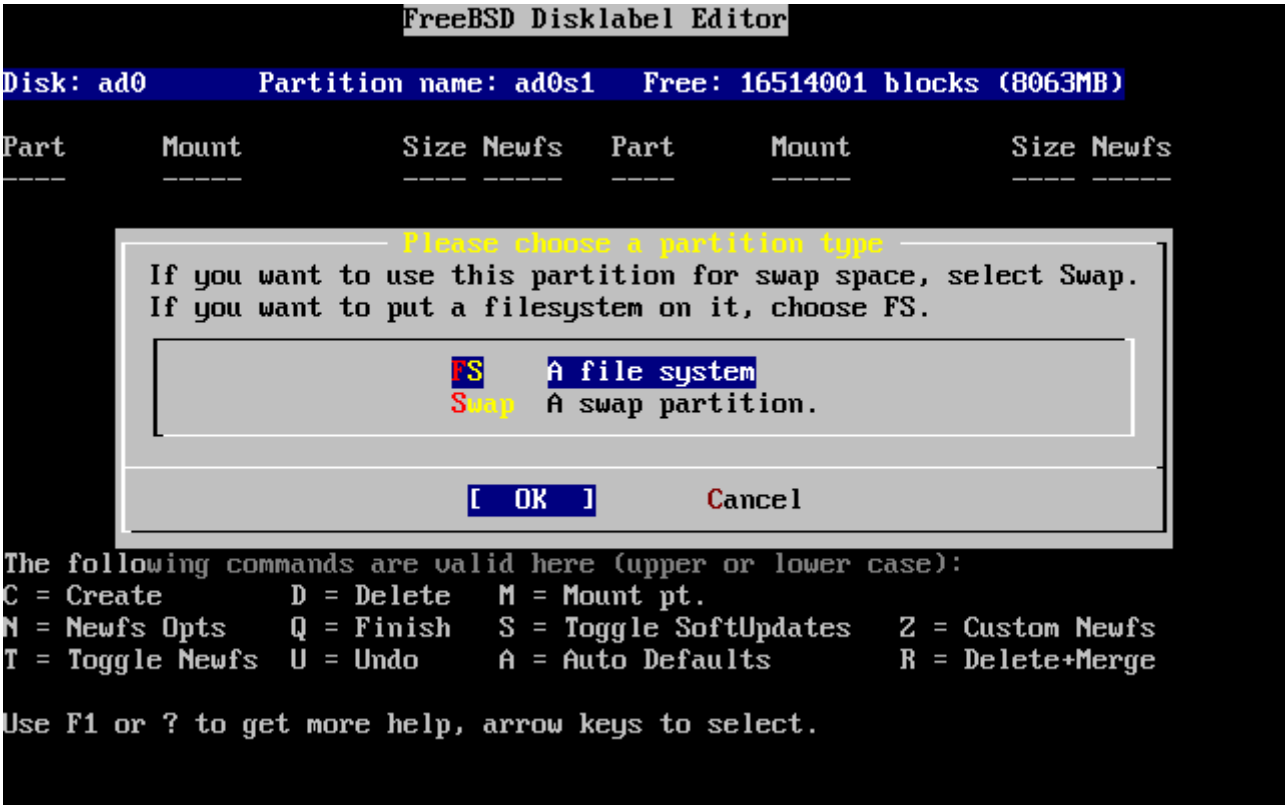
Ábra 20. Szabad hely a gyökérpartíción

Az alapértelmezés szerint felkínált méret az egész slice-ot lefoglaló partíciót hoz létre. Amennyiben a korábbi példában tárgyalt partícióméreteket kívánjuk használni, akkor a `Backspace` billentyû használatával töröljük ki az így megadott értéket, és helyette gépeljük be, hogy `512M`, ahogy ez a A gyökérpartíció méretének szerkesztése segítségével is látható. A bevitelt zárjuk a **[OK]** gomb lenyomásával.



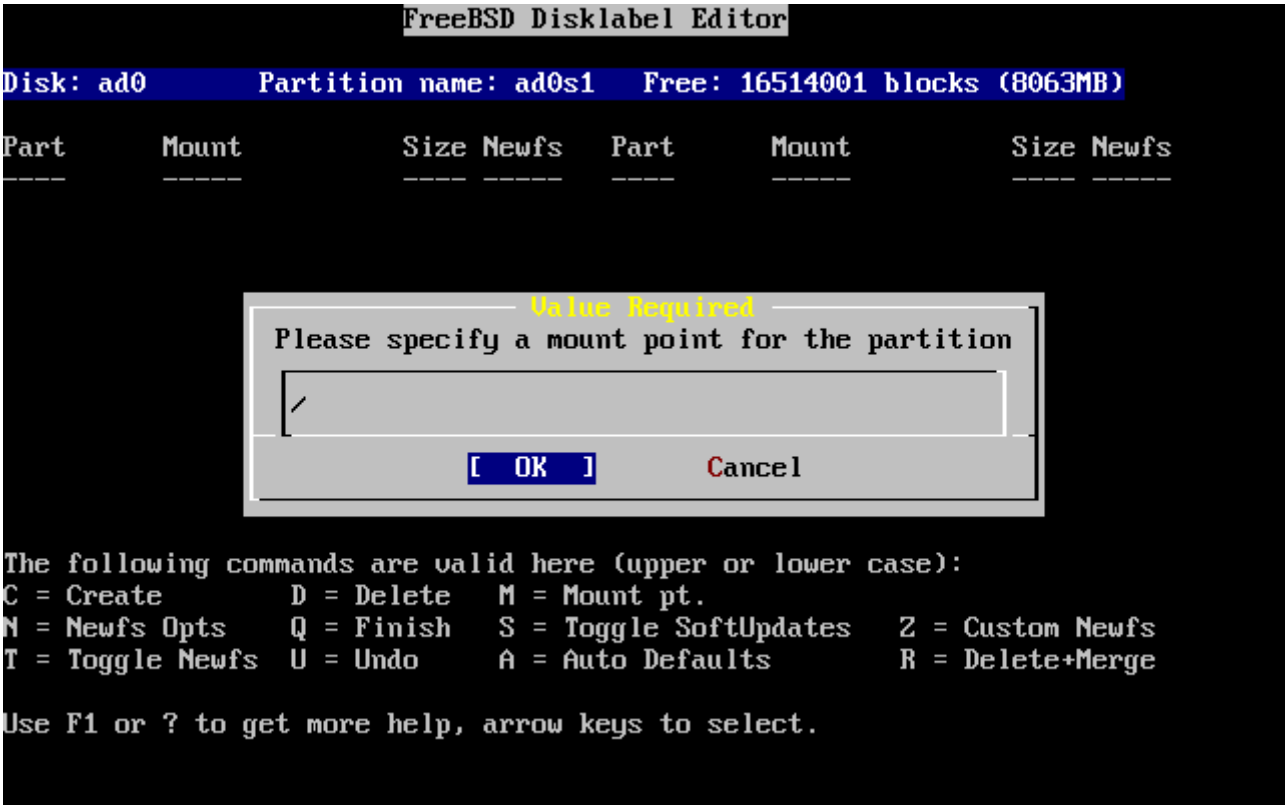
Ábra 21. A gyökérpartíció méretének szerkesztése

Miután meghatároztuk a partíció méretét, a telepítõ megkérdezi, hogy a létrehozandó partícióban állományrendszer vagy lapozóállomány foglaljon-e helyet. Ennek a párbeszédablakját a A gyökérpartíció típusának kiválasztása mutatja. Mivel az elsõ partíciónk állományrendszert fog tartalmazni, ezért mindenképpen az FS paramétert válasszuk ki, majd nyomjuk meg az `Enter` billentyût.



Ábra 22. A gyökérpartíció típusának kiválasztása

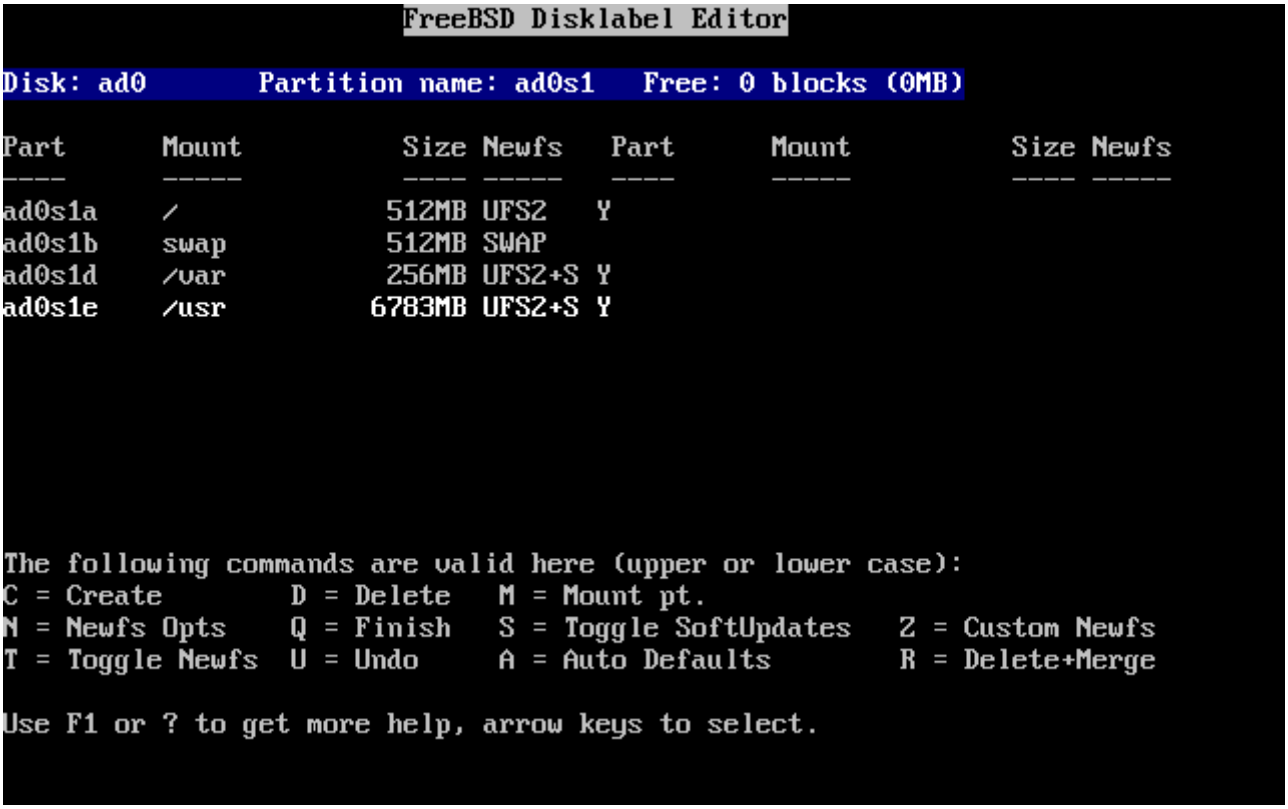
Végezetül, mivel egy állományrendszert hoztunk létre, meg kell mondanunk a Disklabelnek, hova csatlakoztassa. A hozzá tartozó párbeszédablak a A gyökér csatlakozási pontjának megadásan látható. A gyökér állományrendszer csatlakozási pontja a /, ezért itt csak annyit adjunk meg, hogy / és zárjuk az `Enter` billentyû lenyomásával.



Ábra 23. A gyökér csatlakozási pontjának megadása

A képernyõn látható lista ezután az újonnan létrehozott partíciónak megfelelõen frissül. A többi partícióra ugyanígy meg kell ismételnünk ezt a mûveletsort. Arra azonban figyeljünk, hogy a lapozásra használt partíciót létrehozásánál a szerkesztõ nem fogja megkérdezni a csatlakozási pontot, hiszen az ilyen típusú partíciókat sosem csatlakoztatjuk. A /usr, vagyis az utolsó partíció készítése során a slice fennmaradó részének lefoglalásához már nyugodtan meghagyhatjuk a felajánlott értéket.

A FreeBSD partíciószerkesztõjének utolsó képernyõje a A Disklabel partíciószerkesztõn hasonlóhoz, habár az általunk választott értékek minden bizonnyal eltérnek. A mûvelet befejezéséhez nyomjuk le a `Q` billentyût.



Ábra 24. A Disklabel partíciószerkesztõ

2.7. A telepítendõ összetevõk kiválasztása

2.7.1. A terjesztések típusának kiválasztása

A telepítendõ terjesztések típusa nagyban függ attól, hogy a rendszerünket mire szándékozzuk majd használni és mennyi szabad hely áll rendelkezésünkre. Az elõre megadott beállítások a lehetõ legkisebb konfiguráció telepítésétõl egészen a komplett rendszer telepítéséig terjednek. A UNIX® és/vagy FreeBSD világában még az új felhasználók számára szinte tökéletesen megfelelõnek bizonyulhat az egyik ilyen elõkészített beállítás kiválasztása. A terjesztések kiválogatása pedig általában a tapasztaltabb felhasználók számára lehet hasznos.

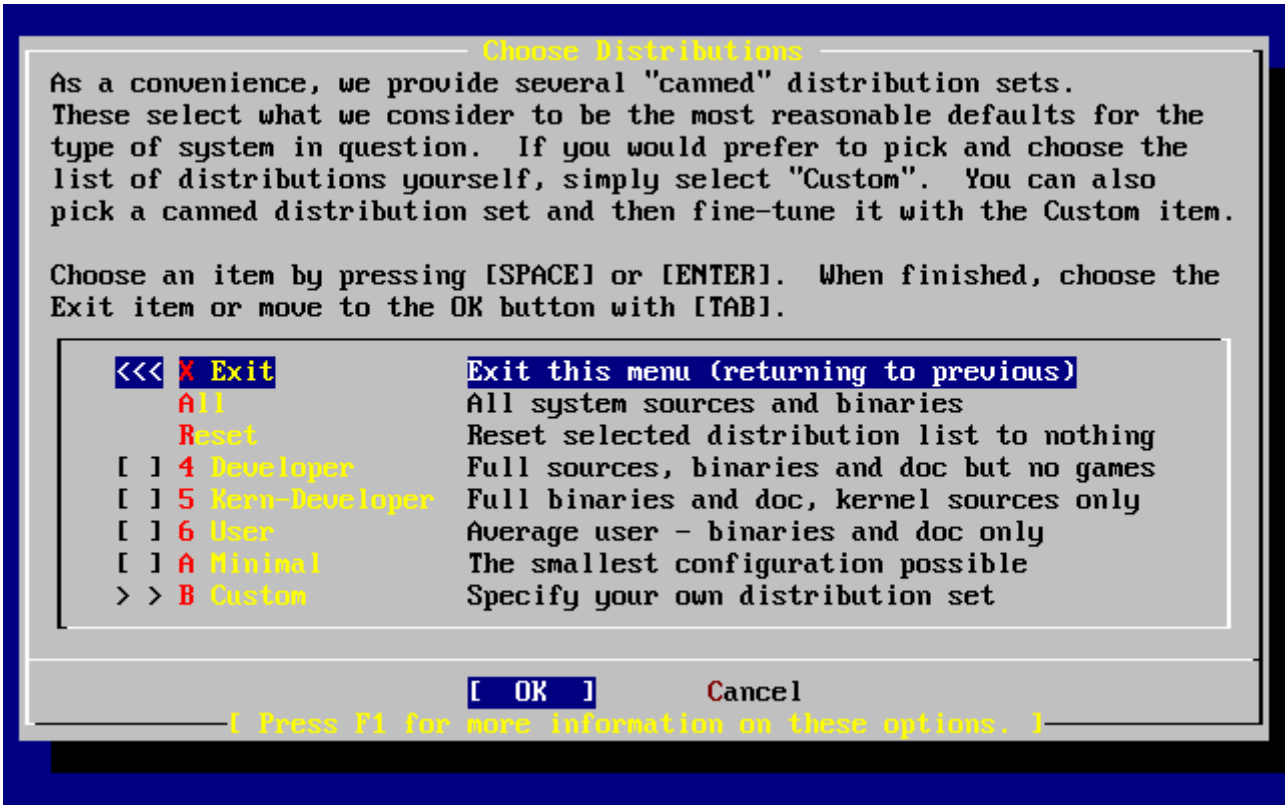
Az `F1` billentyûvel többet is megtudhatunk a terjesztések különbözõ típusairól és bennük található összetevõkrõl. Miután befejeztük a súgó áttanulmányozását, nyomjuk le az `Enter` billentyût, és ezzel visszatérünk a terjesztések kiválasztását tartalmazó menübe.

Ha grafikus felületet szeretnénk használni, akkor az X szerver beállítását az alapértelmezett munkakörnyezet beállítását a FreeBSD telepítése után kell megtenni. Az X szerver beállításáról részletesebben a Az X Window Systemban olvashatunk.

Ha egy saját rendszermag építését is fontolgatjuk, akkor olyan terjesztést válasszuk, amiben a forráskód (kernel source) is megtalálható. A saját rendszermag építésének hátterérõl és mikéntjérõl lásd a A FreeBSD rendszermag testreszabásaet.

Értelemszerûen a legsokoldalúbb rendszer az, amiben minden megtalálató. Így aztán, ha a lemezünk is megengedi, a nyilak és az `Enter` használatával válasszuk a All (Minden) opciót, ahogy azt az A terjesztések kiválasztása is mutatja. Ha viszont úgy érezzük, hogy ehhez nem eléggé nagy a lemezünk, akkor válasszuk az igényeinkhez jobban illeszkedõ típust. Sokat azonban ne üljünk a

tökéletes megoldás kiötlésén, hiszen ezek a terjesztések még a telepítés befejezése után is hozzáadhatóak a rendszerünkhöz.



Ábra 25. A terjesztések kiválasztása

2.7.2. A Portgyûjtemény telepítése

Miután kiválasztottuk a nekünk megfelelõ terjesztést, a telepítõprogram felajánlja a FreeBSD Portgyûjteményének (Ports Collection) telepítésének lehetõségét. A portok gyûjteménye a szoftverek telepítésének egyszerû és kényelmes módja. A Portgyûjtemény önmaga nem tartalmazza a szoftverek lefordításához szükséges forráskódot, hanem helyette csupán azokat az állományokat, amelyek a különbözõ külsõs programok letöltéséhez, fordításához és telepítéséhez kellenek. A Alkalmazások telepítése. csomagok és portokben megtalálhatjuk, miként is kell használni ezt a gyûjteményt.

A telepítõprogram nem fogja ellenõrizni a kibontásához szükséges helyet, ezért csak abban az esetben válasszuk ezt a lehetõséget, ha mindenképpen elfér a merevlemezünkön. A FreeBSD jelenlegi, 12.0 változatában a Portgyûjtemény nagyjából 3 GB helyet foglal el a lemezen. A FreeBSD frissebb verzióiban nyugodtan feltételezhetünk ennél valamivel nagyobb értéket is.

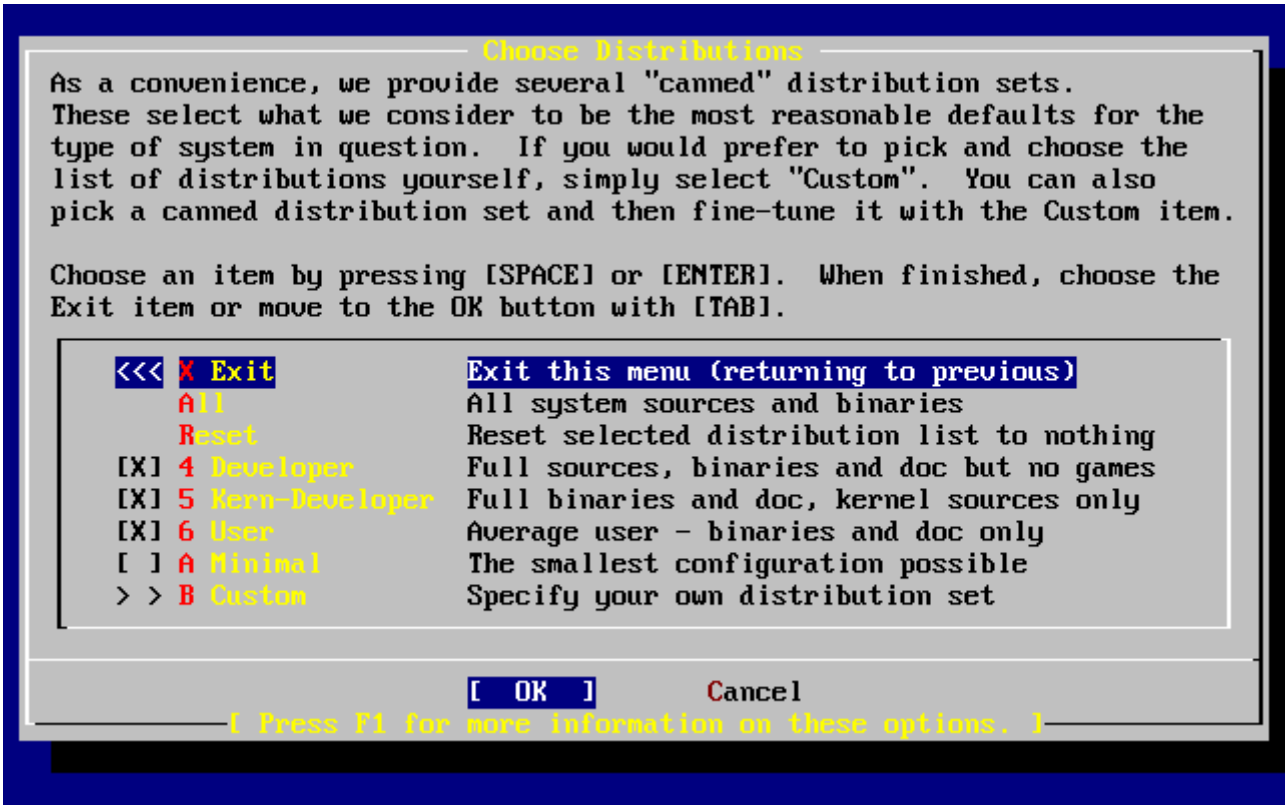
```
                        User Confirmation Requested
Would you like to install the FreeBSD ports collection?

This will give you ready access to over 20 000 ported software packages,
at a cost of around 417 MB of disk space when "clean" and possibly much
more than that if a lot of the distribution tarballs are loaded
(unless you have the extra CDs from a FreeBSD CD/DVD distribution
available and can mount it on /cdrom, in which case this is far less
of a problem).
```

```
The Ports Collection is a very valuable resource and well worth having
on your /usr partition, so it is advisable to say Yes to this option.

For more information on the Ports Collection & the latest ports,
visit:
    http://www.FreeBSD.org/ports

                            [ Yes ]     No
```

Az üzenet fordítása:

```
                     Felhasználói megerősítés szükséges
Szeretné telepíteni a FreeBSD portjainak gyűjteményét?

Ezen keresztül közel 20 000 portolt szoftvercsomaghoz tudunk
könnyedén hozzáférni, amelyek "tiszta" állapotukban nagyjából
417 MB lemezterületünkbe kerülnek, ami a későbbiekben
valószínűleg majd növekedni fog, ahogy letöltjük a különböző
szoftverekhez tartozó állományokat (hacsak nincs meg a FreeBSD
valamelyik CD- vagy DVD alapú terjesztésének az összes lemeze,
amelyeket a /cdrom könyvtárba csatlakoztatva el tudjuk ezeket érni,
mert ekkor kevesebb gondunk lesz vele).

A Portgyűjtemény egy nagyon értékes erőforrás, amelynek megéri helyet
szentelni a /usr partíciónkon, ezért javasoljuk, hogy válassza az
"Igen" opciót. A Portgyűjteményről és annak legújabb portjairól a
http://www.FreeBSD.org/ports oldalon olvashat részletesebben.

                            [ Igen ]     Nem
```

A Portgyûjtemény telepítéséhez a **[yes]** gombot, ennek kihagyásához pedig a **[no]** gombot válasszuk ki a nyilakkal, majd az `Enter` lenyomásával mehetünk tovább. Ekkor a kiválasztott terjesztések menüje fog újra megjelenni.



Ábra 26. A terjesztések telepítésének megerõsítése

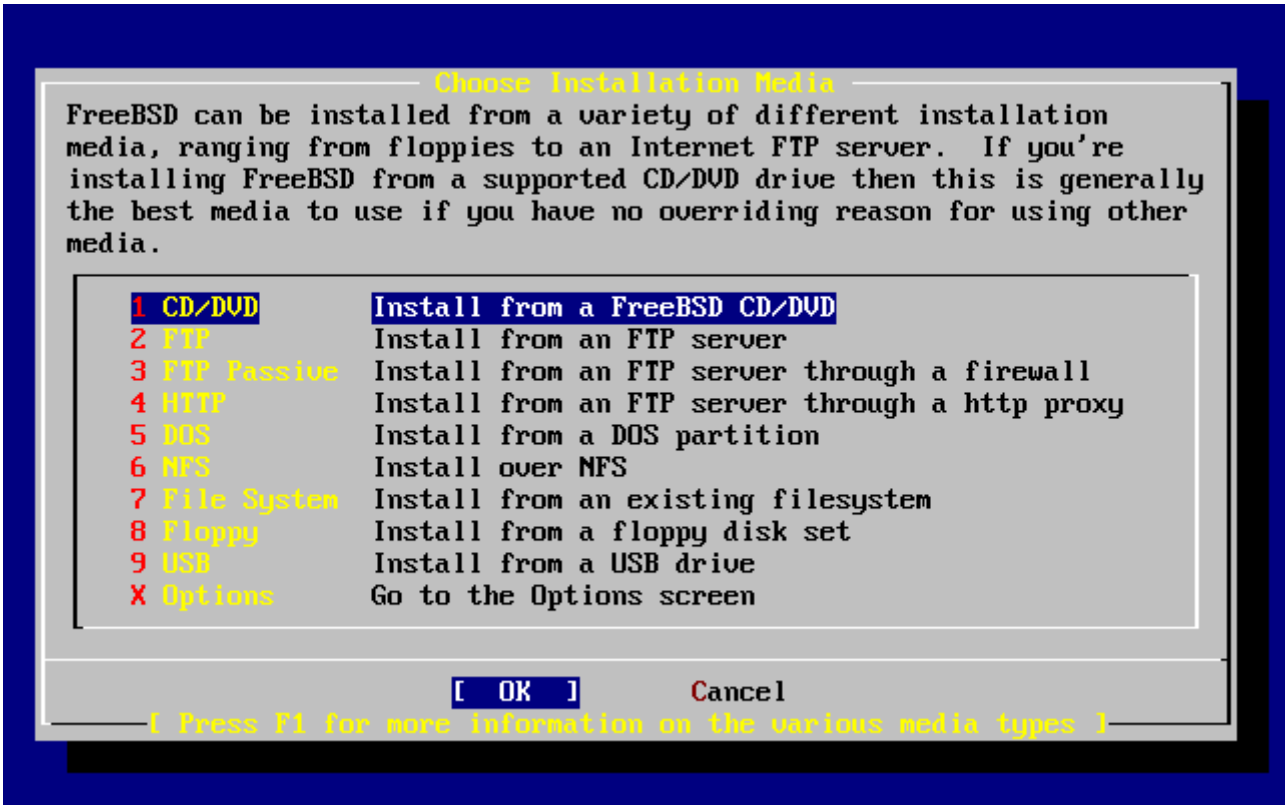
Ha elégedettek vagyunk a beállításokkal, válasszuk ki a nyilakkal az Exit menüpontot, gyõzõdjünk meg róla, hogy a **[OK]** gombon állunk, majd nyomjuk le az `Enter` billentyût a folytatáshoz.

2.8. A telepítés eszközének kiválasztása

Ha CD-rõl vagy DVD-rõl telepítünk, akkor a következõ képernyõn a nyílbillentyûkkel válasszuk ki a Install from a CDROM or DVD (Telepítés CD-rõl vagy DVD-rõl) menüpontot. Ügyeljünk a **[OK]** gomb kiválasztására is, majd a telepítés megkezdéséhez nyomjuk meg az `Enter` billenyût.

A telepítés másfajta módszereinek alkalmazásához válasszuk ki a menüpontok közül a nekünk megfelelõt és kövessük a megjelenõ utasításokat.

Az `F1` billentyû lenyomására megjelenik az adott telepítõeszközhöz tartozó súgó. Innen az `Enter` lenyomása után térhetünk vissza a menühöz.



Ábra 27. A telepítési eszköz kiválasztása

elepítés FTP szerverrõl

Három FTP-s telepítési mód közül választhatunk: aktív, passzív vagy HTTP proxyn keresztül.

Aktív FTP: Install from an FTP server (Telepítés FTP szerverrõl)

Ezzel a beállítással az összes FTP-n keresztüli átvitel "aktív" módban történik. Ez tûzfalak esetén nem mûködik, de gyakran alkalmazható olyan régebbi FTP szerverek esetén, amelyek nem ismerik az passzív adatátvitelt. Ha (az alapértelmezett) passzív módban megakadna a kapcsolat, próbáljunk meg helyette az aktívat.

Passzív FTP: Install from an FTP server through a firewall (Telepítés tûzfalon keresztül FTP szerverrõl)

Ezzel a beállítással a sysinstall programot az FTP mûvelet végrehajtásakor a "passzív" mód használatára utasítjuk. Így át tudunk menni olyan tûzfalakon is, amelyek nem engedik a véletlenszerû TCP portokon érkezõ kapcsolatokat.

FTP HTTP proxyn keresztül: Install from an FTP server through a http proxy (Telepítés HTTP proxyn keresztül FTP szerverrõl)

Ezzel a beállítással megmondhatjuk a sysinstall programnak, hogy (egy böngészõhöz hasonlóan) a HTTP protokollon keresztül használja az FTP mûveletek elvégzéséhez használt proxyt. Ennek a proxynak lesz a feladata az átadott kérések lefordítása és elküldése az FTP szervernek. Ennek köszönhetõen át tudunk menni olyan tûzfalakon is, amelyek egyáltalán nem engednek semmilyen FTP mûveletet, azonban tartozik hozzájuk egy HTTP proxy. Ilyenkor az FTP szerver beállításai mellett meg kell adnunk ezt a HTTP proxyt is.

> Az FTP szervert proxyn keresztül általában úgy érjük el, hogy a felhasználói név részeként egy "@" jellel elválasztva megadjuk a ténylegesen elérni kívánt szerver nevét. A proxy szerver ezután "helyettesíti" a valódi szervert. Például tegyük fel, hogy a `ftp.FreeBSD.org` szerverrõl akarunk telepíteni az 1234 porton várakozó `ize.minta.com` proxy használatával.
>
> Ehhez lépjünk be a beállításokat tartalmazó menübe, állítsuk az FTP kapcsolathoz használt felhasználói nevet az `ftp@ftp.FreeBSD.org` értékre, majd jelszónak adjuk meg az e-mail címünket. Telepítési eszközként adjuk meg az FTP-t (vagy a passzív FTP-t, amennyiben a proxy ismeri) és a `ftp://ize.minta.com:1234/pub/FreeBSD` címet.
>
> Mivel az `ftp.FreeBSD.org` címrõl származó /pub/FreeBSD könyvtár a `ize.minta.com` szerveren keresztül érhetõ el számunkra, ezért lényegében *arról* a géprõl fogunk telepíteni (amely pedig a telepítõ kéréseire elhozza a `ftp.FreeBSD.org` szervertõl az állományokat).

2.9. A telepítés véglegesítése

Ezután ha óhajtjuk, megkezdhetjük a telepítést. Ez egyben az utolsó lehetõségünk a telepítés megszakítására és merevlemezünket érintõ változtatások érvénytelenítésére.

```
                      User Confirmation Requested
Last Chance! Are you SURE you want to continue the installation?

If you're running this on a disk with data you wish to save then WE
STRONGLY ENCOURAGE YOU TO MAKE PROPER BACKUPS before proceeding!

We can take no responsibility for lost disk contents!

                           [ Yes ]    No
```

Az üzenet fordítása:

```
                   Felhasználói megerősítés szükséges
Utolsó esély: BIZTOSAN folytatni kívánja a telepítést?

Ha olyan lemezre szeretne telepíteni, amelyen fontos adatok
találhatóak, HATÁROZOTTAN JAVASOLJUK, hogy a továbblépés előtt
KÉSZÍTSEN RÓLUK MEGBÍZHATÓ BIZTONSÁGI MÁSOLATOT!

Nem vállalunk semmilyen felelősséget az elvesztett adatokért!

                          [ Igen ]    Nem
```

A továbblépéshez válasszuk a **[yes]** gombot és nyomjuk meg az `Enter` billentyût.

A telepítés idõtartama a kiválasztott terjesztéstõl, a telepítésre használt eszköztõl és számítógépünk sebességétõl függ. A folyamat elõrehaladásáról üzenetek sorozata tájékoztat minket.

A telepítés befejezése után a következõ üzenet jelenik meg:

```
                              Message

Congratulations! You now have FreeBSD installed on your system.

We will now move on to the final configuration questions.
For any option you do not wish to configure, simply select No.

If you wish to re-enter this utility after the system is up, you may
do so by typing: /usr/sbin/sysinstall.

                                [ OK ]

                      [  Press enter or space  ]
```

A szöveg fordítása:

```
                              Üzenet

Gratulálunk, sikeresen telepítette a FreeBSD rendszert a számítógépére!

Most rátérünk az utolsó néhány kérdésre. A "Nem" választásával
egyszerűen átugorhatjuk mindazt, amit nem szeretnénk beállítani. Ezt a
segédprogramot a rendszer újbóli elindítása után a "/usr/sbin/sysinstall"
parancs begépelésével tudjuk elérni.

                                [ OK ]

            [ Nyomja le az Enter vagy a Szóköz billentyűt ]
```

Az `Enter` billentyû lenyomásával megkezdhetjük a telepítés utáni beállításokat.

A **[no]** gomb kiválasztásával és az `Enter` lenyomásával megszakíthatjuk a telepítést, így a rendszerünkön semmilyen változtatás nem történik. Ilyenkor a következõ üzenet jelenik meg:

```
                              Message
Installation complete with some errors.  You may wish to scroll
through the debugging messages on VTY1 with the scroll-lock feature.
You can also choose "No" at the next prompt and go back into the
installation menus to retry whichever operations have failed.

                                [ OK ]
```

Az üzenet fordítása:

```
                                Üzenet
A telepítés során hiba történt. A Scroll Lock használatával érdemes
átnézni a VTY1 terminál megjelenő üzeneteket. A következő ablakban a
"Nem" választásával vissza tudunk menni a telepítőmenühöz és
megpróbálkozhatunk ismét a sikertelen műveletek végrehajtásával.

                                [ OK ]
```

Ez az üzenet azért jelent meg, mert semmit sem sikerült telepíteni. Innen az `Enter` megnyomásával térhetünk vissza a fõmenübe, majd onnan tudunk kilépni a telepítõbõl.

2.10. A telepítés után

A sikeres telepítést különféle beállítások követik. Közülük az új FreeBSD rendszer indítása elõtt bármelyik megismételhetõ a beállítások opcióit tartalmazó menü újbóli használatával, vagy pedig a telepítés után a `sysinstall` parancs kiadásával, majd a Configure (Beállítások) menüpont kiválasztásával.

2.10.1. A hálózati eszközök beállítása

A következõ képernyõ már nem jelenik meg, ha az FTP szerveren keresztüli telepítéshez korábban már beállítottuk a PPP kapcsolatot. Ez a korábbiakban említettek szerint állítható be.

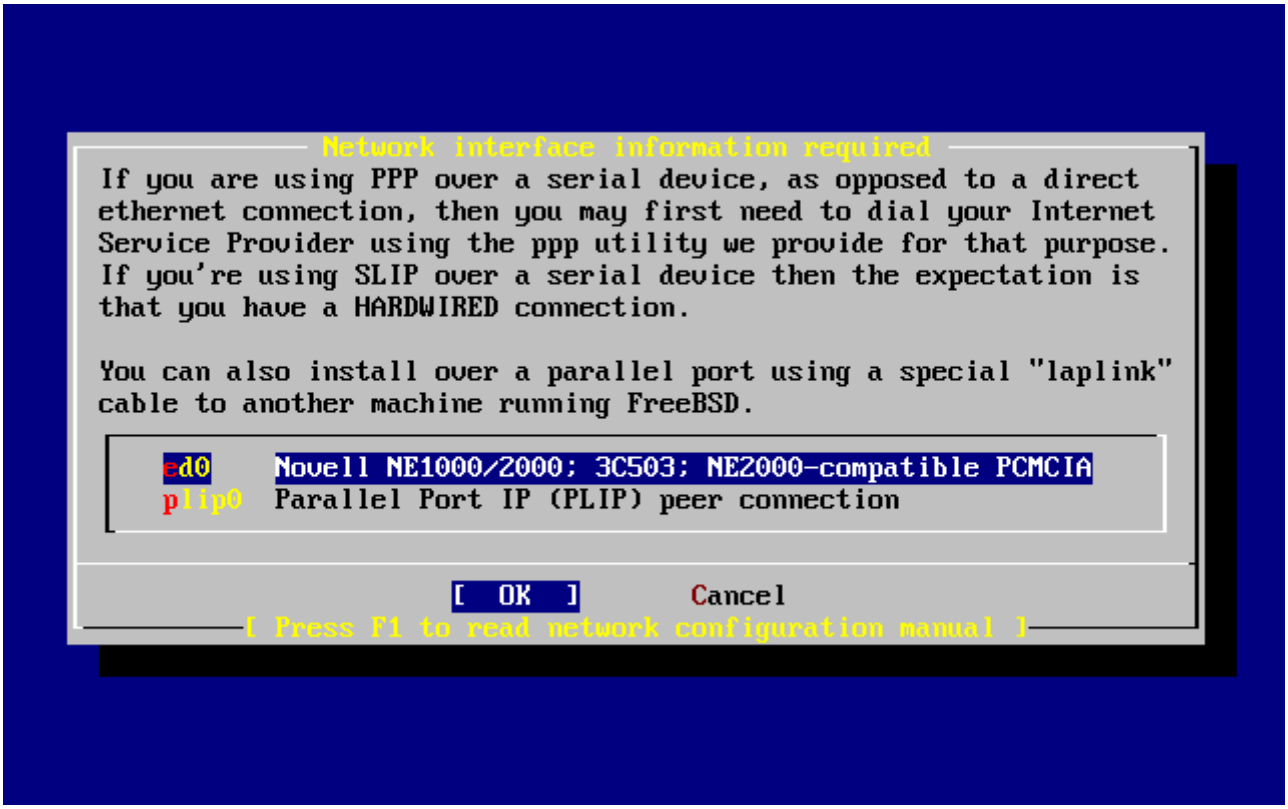
Ha többet szeretnénk megtudni a helyi hálózatokról (LAN), vagy a FreeBSD-t átjáróként, illetve útválasztóként kívánjuk beállítani, olvassuk el az Egyéb haladó hálózati témák címû fejezetet.

```
                    User Confirmation Requested
  Would you like to configure any Ethernet or PPP network devices?

                            [ Yes ]   No
```

Fordítása:

```
                  Felhasználói megerősítés szükséges
  Szeretnénk beállítani valamilyen Ethernet- vagy PPP hálózati eszközt?

                           [ Igen ]   Nem
```

A hálózati eszközeink beállításához válasszuk a **[yes]** gombot, majd nyomjuk meg az `Enter` billentyût. Ellenkezõ esetben a **[no]** gombbal mehetünk tovább.



Ábra 28. Az Ethernet-eszköz kiválasztása

A beállítandó csatoló kiválasztásához használjuk a nyílbillentyûket és utána nyomjuk meg az `Enter` billentyût.

```
                  User Confirmation Requested
      Do you want to try IPv6 configuration of the interface?

                          Yes   [ No ]
```

Fordítás:

```
              Felhasználói megerősítés szükséges
        Megpróbálkozik az IPv6 beállításával a csatolón?

                          Igen   [ Nem ]
```

A példánkban szereplõ helyi hálózatban az aktuális internetes protokoll (IPv4) egyelõre megfelelõ, ezért válasszuk a **[no]** gombot és nyomjuk meg az `Enter` billentyût.

Amennyiben RA-szerveren keresztül egy már létezõ IPv6 hálózathoz csatlakozunk, akkor válasszuk a **[yes]** gombot és nyomjuk meg az `Enter` billentyût. Ezt követõen az RA-szerverek felderítése kezdõdik meg, ami néhány másodpercig eltarthat.

```
                          User Confirmation Requested
      Do you want to try DHCP configuration of the interface?
```

```
                          Yes   [ No ]
```

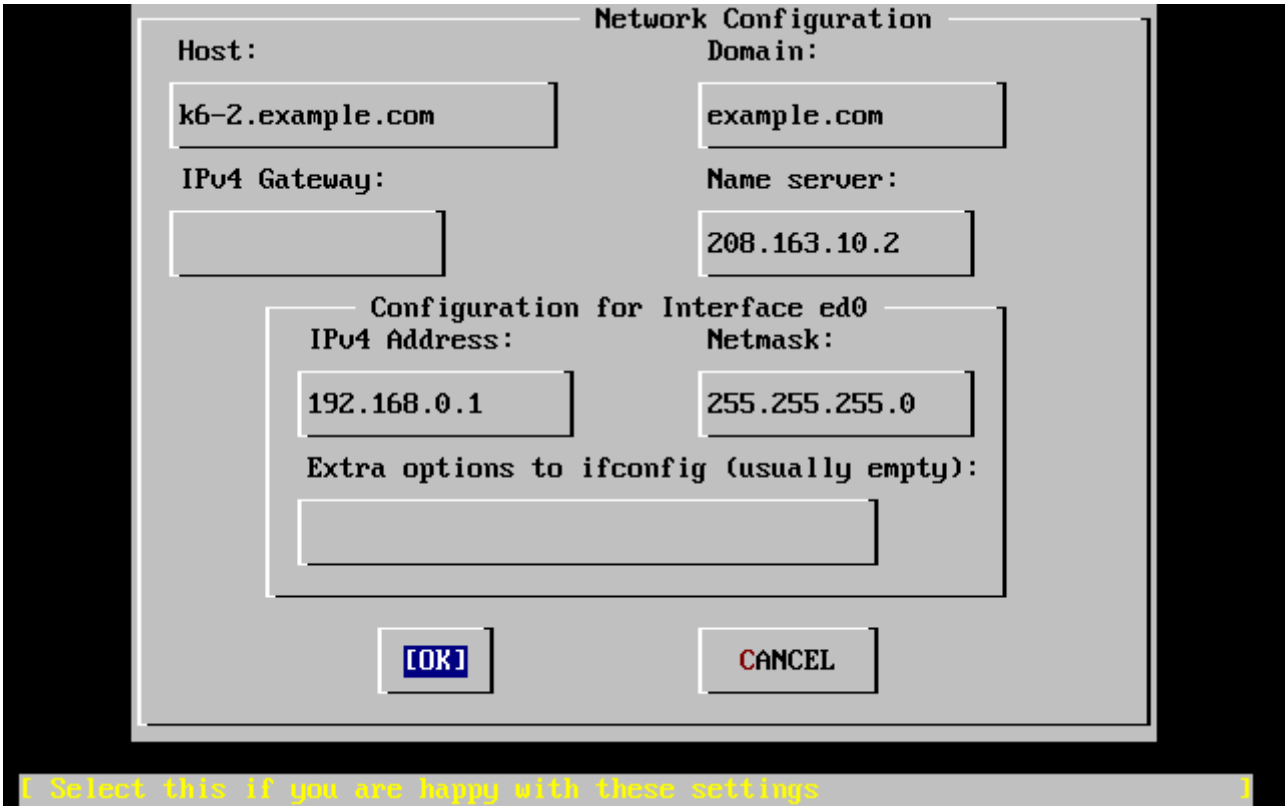
Az üzenet fordítása:

```
                      Felhasználói megerősítés szükséges
          Megpróbálkozik a DHCP használatával a csatolón?

                          Igen   [ Nem ]
```

Ha nincs szükségünk a DHCP (Dynamic Host Configuration Protocol, azaz a Dinamikus állomáskonfigurációs protokoll) használatára, akkor a **[no]** gomb kiválasztásával majd az `Enter` lenyomásával továbbléphetünk.

A **[yes]** gomb kiválasztására elindul a dhclient nevû program, és amennyiben sikerrel jár, magától kitölti a hálózati beállításokra vonatkozó adatokat. Ennek részleteit a A hálózat automatikus beállítása (DHCP)ben találhatjuk meg.

Az alábbi hálózati beállító képernyõ mutatja a helyi hálózat átjárójaként használni kívánt Ethernet-eszköz konfigurációját.



Ábra 29. Az ed0 hálózati beállítása

A `Tab` billentyûvel tudunk navigálni az adatlap mezõi között és kitölteni ezeket a megfelelõ információkkal:

Host (Számítógépnév)

A számítógépünk teljes neve, amely a példában most `k6-2.example.com`.

Domain (Tartomány)

Annak a tartománynak a neve, amelyben a számítógépünk a található. Ez itt konkrétan a `example.com`.

IPv4 Gateway (IPv4-átjáró)

A helyben nem elérhetõ célok megközelítésére használt gép IP-címe. Ezt a mezõt mindenképpen töltsük ki akkor, ha a számítógépünk valamilyen hálózatba van kötve. Azonban *hagyjuk üresen*, ha a számítógép a hálózat átjárója az internet felé. Az IPv4 átjárót más néven "default gateway"-nek (alapértelmezett átjárónak) vagy "default route"-nak (alapértelmezett útvonalnak) is nevezik.

Name server (Névszerver)

A helyi DNS (névfeloldó) szerverünk IP-címe. Ha nem található ilyen a helyi hálózatunkon, akkor az internet-szolgáltató DNS szerverének címét (a példában ez a `208.163.10.2`) adjuk meg.

IPv4 address (IPv4-cím)

A csatoló IP-címe, amely az ábrán a `192.168.0.1`.

Netmask (Hálózati maszk)

A helyi hálózatban használt címtartomány a `192.168.0.0` - `192.168.0.255`, amihez a `255.255.255.0` hálózati maszk tartozik.

Extra options to ifconfig (Az ifconfig további beállításai)

Az `ifconfig` parancs adott csatolóra vonatkozó egyéb beállításai. Jelen esetünkben itt semmi sem szerepel.

Miután végeztünk, a `Tab` billentyû lenyomásával válasszuk ki a **[OK]** gombot és nyomjuk le az `Enter` billentyût.

```
                  User Confirmation Requested
       Would you like to bring the ed0 interface up right now?

                          [ Yes ]   No
```

A fordítás:

```
                 Felhasználói megerősítés szükséges
                   Aktiválja most az ed0 csatolót?

                         [ Igen ]   Nem
```

A **[yes]** gomb kiválasztásával, majd az `Enter` lenyomásával csatlakoztatjuk a számítógépet a hálózathoz, ami ezután használhatóvá válik. Ez azonban a telepítés számára nem jelent túlságosan sokat, hiszen ettõl függetlenül a számítógépet egyébként is újra kell majd indítanunk.

2.10.2. Az átjáró beállítása

```
                    User Confirmation Requested
       Do you want this machine to function as a network gateway?

                              [ Yes ]    No
```

A fordítás:

```
                  Felhasználói megerősítés szükséges
       Ezt a számítógépet hálózati átjáróként is használni akarja?

                             [ Igen ]    Nem
```

Ha a számítógépet a helyi hálózat átjárójaként használni akarjuk gépek közti csomagok továbbítására, akkor válasszuk a **[yes]** gombot és nyomjuk meg hozzá az `Enter` billentyût. Ha viszont ez a gép csupán a hálózat egy tagja, akkor válasszuk a **[no]** gombot és a folytatáshoz nyomjuk meg az `Enter` billentyût.

2.10.3. A hálózati szolgáltatások beállítása

```
                      User Confirmation Requested
Do you want to configure inetd and the network services that it provides?

                               Yes   [ No ]
```

Fordítás:

```
                   Felhasználói megerősítés szükséges
Beállítja az inetd démont és az általa felkínált hálózati szolgáltatásokat?

                              Igen   [ Nem ]
```

Ha itt a **[no]** gombot választjuk, akkor ezzel kikapcsoljuk a különbözõ szolgáltatásokat, például a telnetd démont. Ez azt jelenti, hogy a távoli felhasználók nem lesznek képesek a telnet program használatával belépni erre a számítógépre. A helyi felhasználók viszont továbbra is képesek lesznek távoli számítógépeket elérni a telnet segítségével.

Az /etc/inetd.conf átírásával azonban ezek a szolgáltatások késõbb természetesen engedélyezhetõek. A Áttekintés foglalkozik a téma részleteivel.

A **[yes]** gomb választásával már a telepítés során beállíthatjuk a szolgáltatásokat. Ekkor egy további párbeszédablak is felbukkan:

```
                      User Confirmation Requested
```

```
The Internet Super Server (inetd) allows a number of simple Internet
services to be enabled, including finger, ftp and telnetd.  Enabling
these services may increase risk of security problems by increasing
the exposure of your system.

With this in mind, do you wish to enable inetd?

                            [ Yes ]   No
```

Fordítása:

```
                   Felhasználói megerősítés szükséges
A fő internetes kiszolgáló (az inetd) számos egyszerű internetes
szolgáltatás, többek közt a finger, ftp és telnet elérését teszi
lehetővé. Ezen szolgáltatások engedélyezése azonban a felmerülő
biztonsági problémák kockázatát, mivel ezzel rendszerünket jobban
kitesszük támadásoknak.

Mindezek tudatában használni kívánja az inetd démont?

                           [ Igen ]   Nem
```

A folytatáshoz válasszuk a **[yes]** gombot.

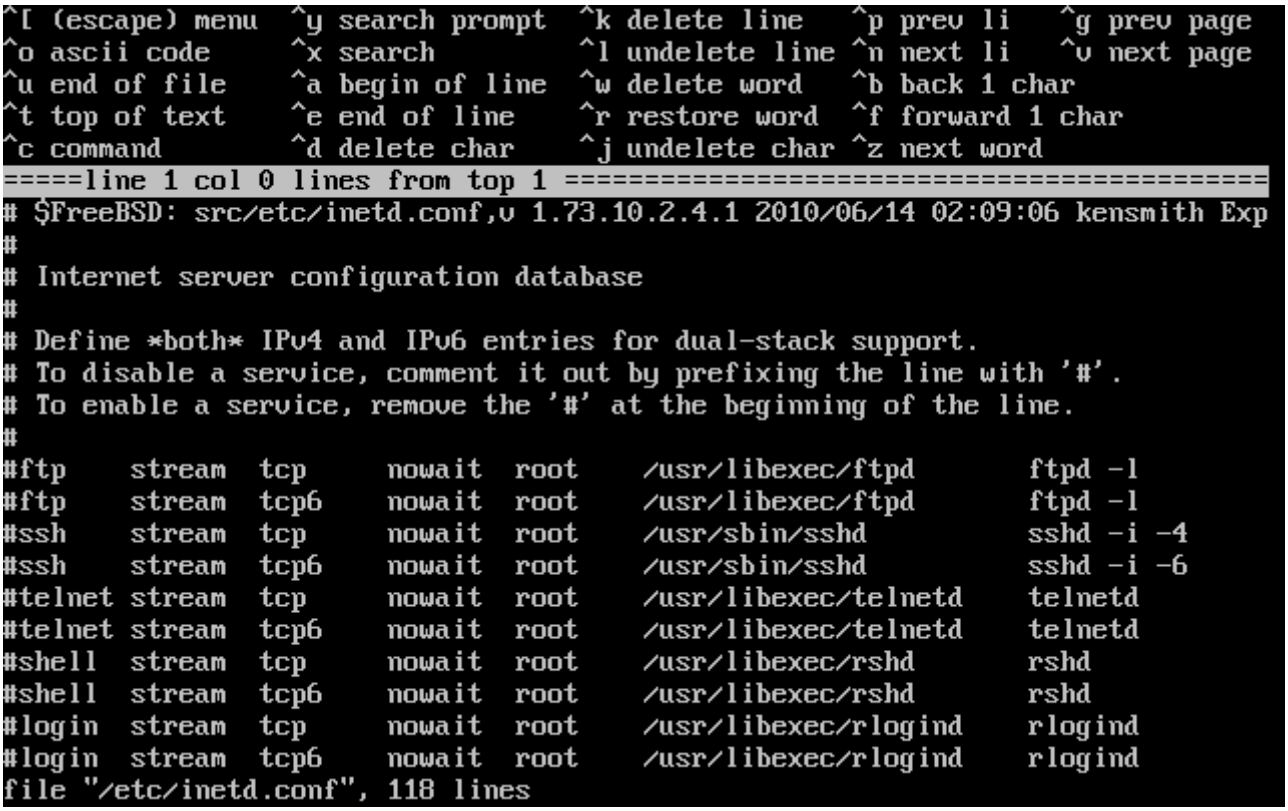
```
                      User Confirmation Requested
inetd(8) relies on its configuration file, /etc/inetd.conf, to determine
which of its Internet services will be available.  The default FreeBSD
inetd.conf(5) leaves all services disabled by default, so they must be
specifically enabled in the configuration file before they will
function, even once inetd(8) is enabled.  Note that services for
IPv6 must be separately enabled from IPv4 services.

Select [Yes] now to invoke an editor on /etc/inetd.conf, or [No] to
use the current settings.

                            [ Yes ]   No
```

Fordítás:

```
                   Felhasználói megerősítés szükséges
Az inetd(8) démonnak az elérhető internetes szolgáltatások
megállapításához szüksége van a beállításait tartalmazó
/etc/inetd.conf állományra. A FreeBSD-hez tartozó inetd.conf(5)
állomány alapértelmezés szerint az összes szolgáltatást letiltja,
ezért a működéséhez minden egyes szolgáltatást külön kell engedélyezni
az említett állományban, még abban az esetben is, ha az inetd(8)
démont korábban már engedélyeztük. Az IPv6 szolgáltatások az IPv4
```

```
szolgáltatásoktól külön engedélyezendőek.

Az [ Igen ] választásával behívjuk az /etc/inetd.conf szerkesztését,
míg a [ Nem ] választásával pedig az imént felvázolt beállításokat
fogadjuk el.

                          [ Igen ]   Nem
```

A **[yes]** gomb kiválasztásával lehetõségünk nyílik szolgáltatásokat engedélyezni a sorok elején található # jel törlésével.



Ábra 30. Az inetd.conf módosítása

Miután felvettük az összes használni kívánt szolgáltatást, az `Esc` billentyû lenyomásával elõhozhatjuk azt a menüt, ahol elmenthetjük a módosításainkat és kiléphetünk.

2.10.4. Az SSH-n keresztüli bejelentkezés engedélyezése

```
                User Confirmation Requested
             Would you like to enable SSH login?
                     Yes       [  No  ]
```

Fordítás:

```
                Felhasználói megerősítés szükséges
             Engedélyezi az SSH-n keresztüli bejelentkezést?
                      Igen       [  Nem  ]
```

A **[yes]** gomb kiválasztása engedélyezi az OpenSSH-hoz tartozó sshd(8) démont, aminek segítségével a számítógépünkre biztonságosan be tudunk jelentkezni távolról. Az OpenSSH részleteirõl lásd a OpenSSHt.

2.10.5. Anonim FTP

```
                    User Confirmation Requested
Do you want to have anonymous FTP access to this machine?

                             Yes    [ No ]
```

Fordítás:

```
                 Felhasználói megerősítés szükséges
Hozzáférhető legyen ez a számítógép anonim FTP használatán keresztül?

                            Igen    [ Nem ]
```

Az anonim FTP tiltása

Az alapértelmezett **[no]** gomb kiválasztásával és az `Enter` billentyû lenyomásával a jelszóval védett FTP hozzáféréssel rendelkezõ felhasználók továbbra is elérhetik a számítógépünket.

Az anonim FTP engedélyezése

Ha ezt választjuk, akkor anonim FTP kapcsolaton keresztül bárki hozzáférhet a számítógépünkhöz. Ebben az esetben azonban alaposan meg kell fontolnunk néhány biztonsági következményt. A beállítással járó kockázatokról az Biztonságben olvashatunk többet.

Az anonim FTP bekapcsolásához a nyílbillentyûkkel válasszuk ki a **[yes]** feliratú gombot és nyomjuk meg az `Enter` billentyût. Ekkor egy további párbeszédablak is megjelenik:

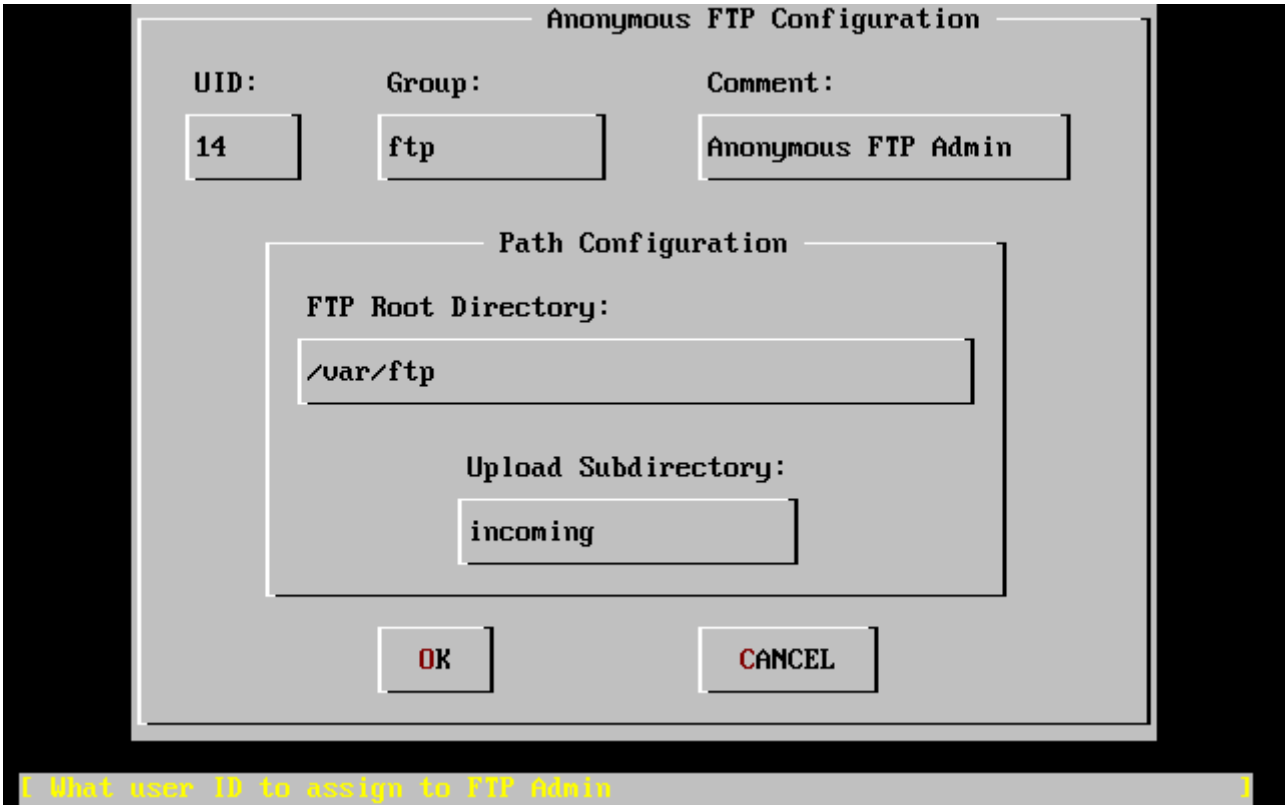
```
                    User Confirmation Requested
Anonymous FTP permits un-authenticated users to connect to the system
FTP server, if FTP service is enabled.  Anonymous users are
restricted to a specific subset of the file system, and the default
configuration provides a drop-box incoming directory to which uploads
are permitted.  You must separately enable both inetd(8), and enable
ftpd(8) in inetd.conf(5) for FTP services to be available.  If you
did not do so earlier, you will have the opportunity to enable inetd(8)
again later.

If you want the server to be read-only you should leave the upload
directory option empty and add the -r command-line option to ftpd(8)
in inetd.conf(5)

Do you wish to continue configuring anonymous FTP?
```

```
                        [ Yes ]         No
```

Az üzenet fordítása:

```
                    Felhasználói megerősítés szükséges

Az anonim FTP használatával a rendszer FTP szolgáltatásához
hitelesítetlen felhasználók is hozzáférhetnek, amennyiben az aktív. A
névtelen felhasználók az állományrendszernek csak egy részét érhetik
el, valamint az alapbeállítások szerint a feltöltést egy külön erre
a célra fenntartott könyvtárba végezhetik el. Az FTP szolgáltatás
használatát külön engedélyeznünk kell az inetd(8) démon részéről és
az inetd.conf(5) állományban található ftpd(8) démon aktiválásával.
Ha eddig még nem tettük volna meg, akkor az inetd(8) használatát
később még újra engedélyezhetjük.

Ha csak letöltést kívánunk engedni, akkor hagyjuk a feltöltési
könyvtárra vonatkozó paramétert üresen és az inetd.conf(5)
állományban az ftpd(8) parancssorához adjuk hozzá az -r kapcsolót.

Folytatja az anonim FTP beállítását?

                        [ Igen ]        Nem
```

Az üzenet értesít minket arról, hogy az anonim FTP kapcsolatok engedélyezéséhez az FTP szolgáltatást az /etc/inetd.conf állományban is be kell majd kapcsolni, lásd A hálózati szolgáltatások beállítása. Válasszuk a **[yes]** gombot és a folytatáshoz nyomjuk meg az `Enter` billentyût. Ekkor a következõ képernyõ jön elõ:



Ábra 31. Az anonim FTP alapbeállításai

A beállítások kitöltése során a `Tab` billentyûvel mozoghatunk az adatmezõk között:

UID (felhasználói azonosító)

A névtelen FTP felhasználókhoz társított felhasználói azonosító. A feltöltött állomány tulajdonosa ez az azonosító lesz.

Group (csoport)

A névtelen FTP felhasználók csoportja.

Comment (megjegyzés)

Ez a szöveg szerepel a felhasználónál az /etc/passwd állományban.

FTP Root Directory (az FTP gyökere)

Itt találhatóak az anonim FTP-n keresztül elérhetõ állományok.

Upload Subdirectory (feltöltési könyvtár)

A névtelen FTP felhasználók által feltöltött állományok ide kerülnek.

Az FTP gyökere alapból a /var könyvtár lesz. Ha a becsült FTP-forgalom lebonyolításához itt nem rendelkezünk elegendõ hellyel, akkor az /usr könyvtárban található /usr/ftp alkönyvtár is beállítható az FTP gyökerének.

Ha elfogadhatónak találjuk az értékeket, nyomjuk le az `Enter` billentyût a folytatáshoz.

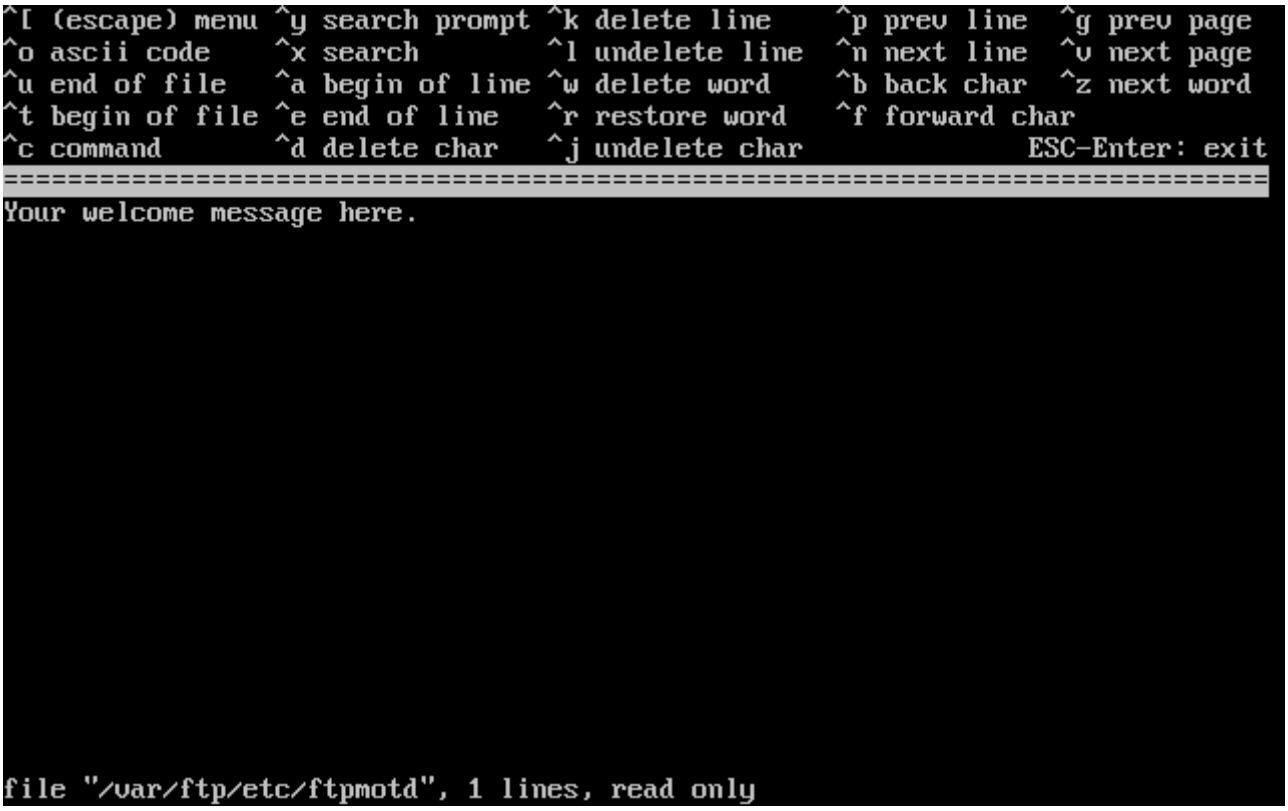
```
                    User Confirmation Requested
        Create a welcome message file for anonymous FTP users?
```

```
                              [ Yes ]    No
```

Fordítás:

```
                     Felhasználói megerősítés szükséges
        Létre kíván hozni egy köszöntő üzenetet tartalmazó állományt
        az anonim FTP felhasználók számára?

                              [ Igen ]    Nem
```

A **[yes]** választásával és az `Enter` megnyomásával az üzenet szerkesztéséhez egy szövegszerkesztõ fog elindulni.



Ábra 32. Az FTP köszöntõ üzenetének szerkesztése

Ez az ee szövegszerkesztõ. Az üzenet átírásához használjuk a megadott utasításokat, de akár késõbb is módosíthatjuk ezt a kedvenc szövegszerkesztõnkkel. Ehhez a módosítandó állomány neve és helye a szerkesztõ képernyõjének alján olvasható.

A kilépéshez az `Esc` lenyomására felbukkanó menüben alapból az a) leave editor (kilépés a szerkesztõbõl) menüpont érhetõ el, ezért itt az `Enter` lenyomásával léphetünk tovább. Az `Enter` ismételt lenyomásával elmenthetjük a módosításainkat.

2.10.6. A hálózati állományrendszer beállítása

A hálózati állományrendszer (Network File System, NFS) állományok közzétételét teszi lehetõvé hálózaton keresztül. Használata során egy számítógép beállítható szervernek, kliensnek vagy akár mindkettõnek. Ezzel kapcsolatban a A hálózati állományrendszer (NFS) ajánlott elolvasásra.

Az NFS szerver

```
                      User Confirmation Requested
  Do you want to configure this machine as an NFS server?

                                Yes    [ No ]
```

A fordítása:

```
                    Felhasználói megerősítés szükséges
    Be akarja állítani NFS szervernek ezt a számítógépet?

                               Igen    [ Nem ]
```

Ha nincs szükségünk a hálózati állományrendszer szerver részére, akkor válasszuk a **[no]** gombot és nyomjuk le az `Enter` billentyût.

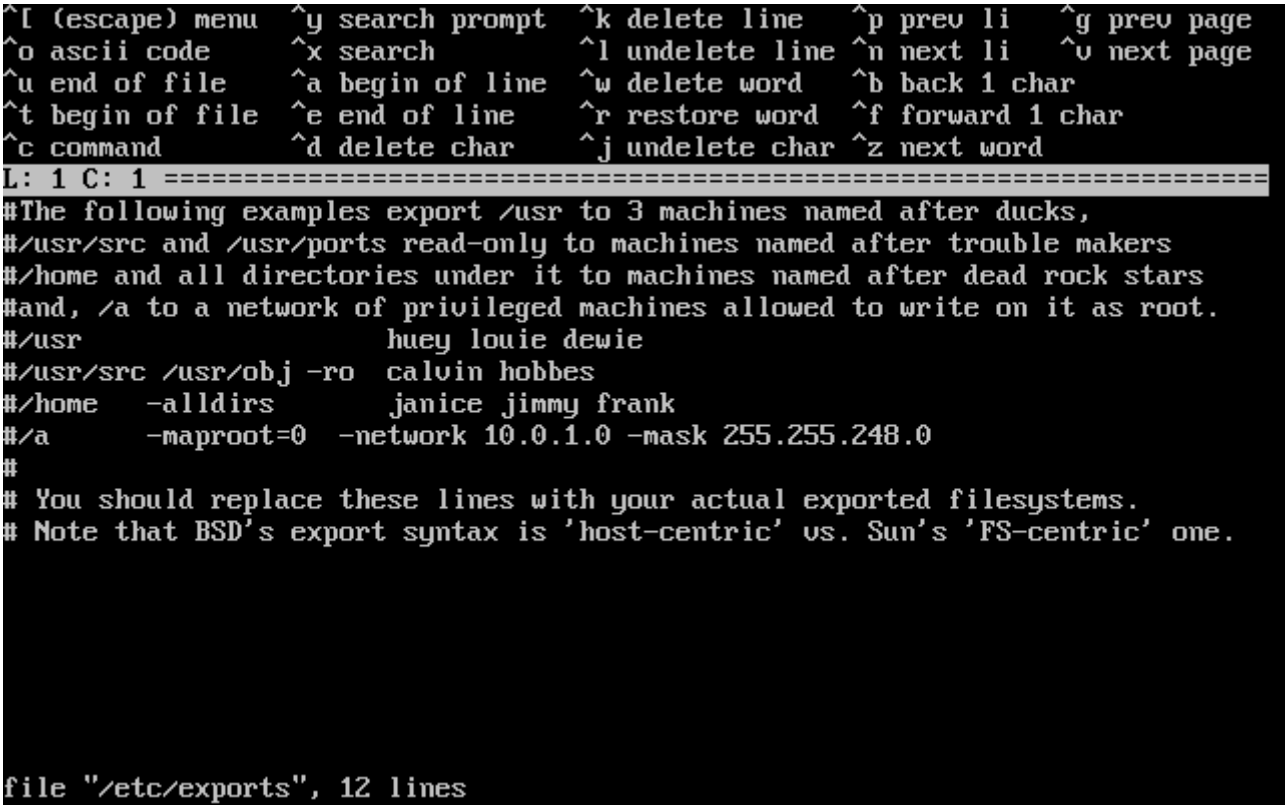
Amennyiben a **[yes]** gombot választjuk, egy üzenet fogja közölni velünk, hogy létre kell hoznunk az exports állományt.

```
                                Message
Operating as an NFS server means that you must first configure an
/etc/exports file to indicate which hosts are allowed certain kinds of
access to your local filesystems.
Press [Enter] now to invoke an editor on /etc/exports
                                [ OK ]
```

Az üzenet fordítása:

```
                                Üzenet
Az NFS szerver működtetéséhez először az /etc/exports állomány
összeállításán keresztül meg kell adnunk, hogy milyen gépek milyen
típusú hozzáféréssel rendelkezzenek a helyi állományrendszereinken.
Az [Enter] lenyomására megkezdődik az /etc/exports állomány
szerkesztése.
                               [ OK ]
```

Az `Enter` billentyû lenyomásával továbbléphetünk. Ekkor az exports állomány létrehozására és szerkesztésére egy szövegszerkesztõ indul el.



Ábra 33. Az exports szerkesztése

A exportálni kívánt állományrendszerek felsorolásához használjuk képernyõn a megadott utasításokat, vagy tegyük meg ezt késõbb az általunk választott szövegszerkesztõ segítségével. Ilyenkor ne felejtsük el megjegyezni az állomány képernyõ alján látható nevét és helyét.

Amikor végeztünk, az `Esc` billentyûvel felhozható menüben alapból az a) leave editor (kilépés a szövegszerkesztõbõl) menüpont aktív, ezért itt a folytatáshoz egyszerûen nyomjuk le az `Enter` billentyût.

Az NFS kliens

Az NFS kliens beállításával NFS szerverekhez tudunk hozzáférni.

```
                      User Confirmation Requested
  Do you want to configure this machine as an NFS client?

                                Yes   [ No ]
```

Fordítás:

```
                    Felhasználói megerősítés szükséges
       Beállítja NFS kliensnek ezt a számítógépet?

                               Igen   [ Nem ]
```

A nyílbillentyûkkel igényeinknek megfelelõen válasszuk a **[yes]** vagy **[no]** gombokat és utána nyomjuk meg az `Enter` billentyût.

2.10.7. A rendszerkonzol beállításai

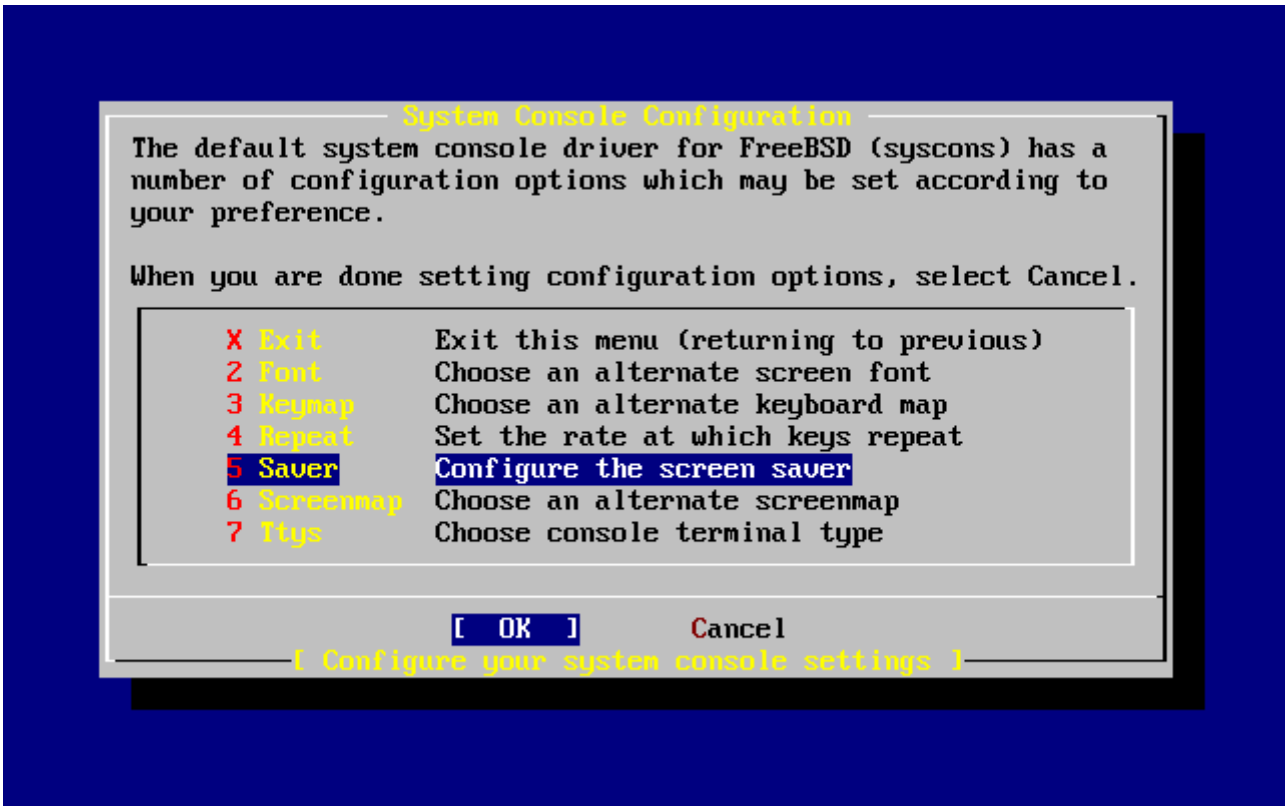
Számos beállítás kapcsolódik a rendszerben található konzolok testreszabásához.

```
                    User Confirmation Requested
        Would you like to customize your system console settings?

                                [ Yes ]  No
```

Fordítás:

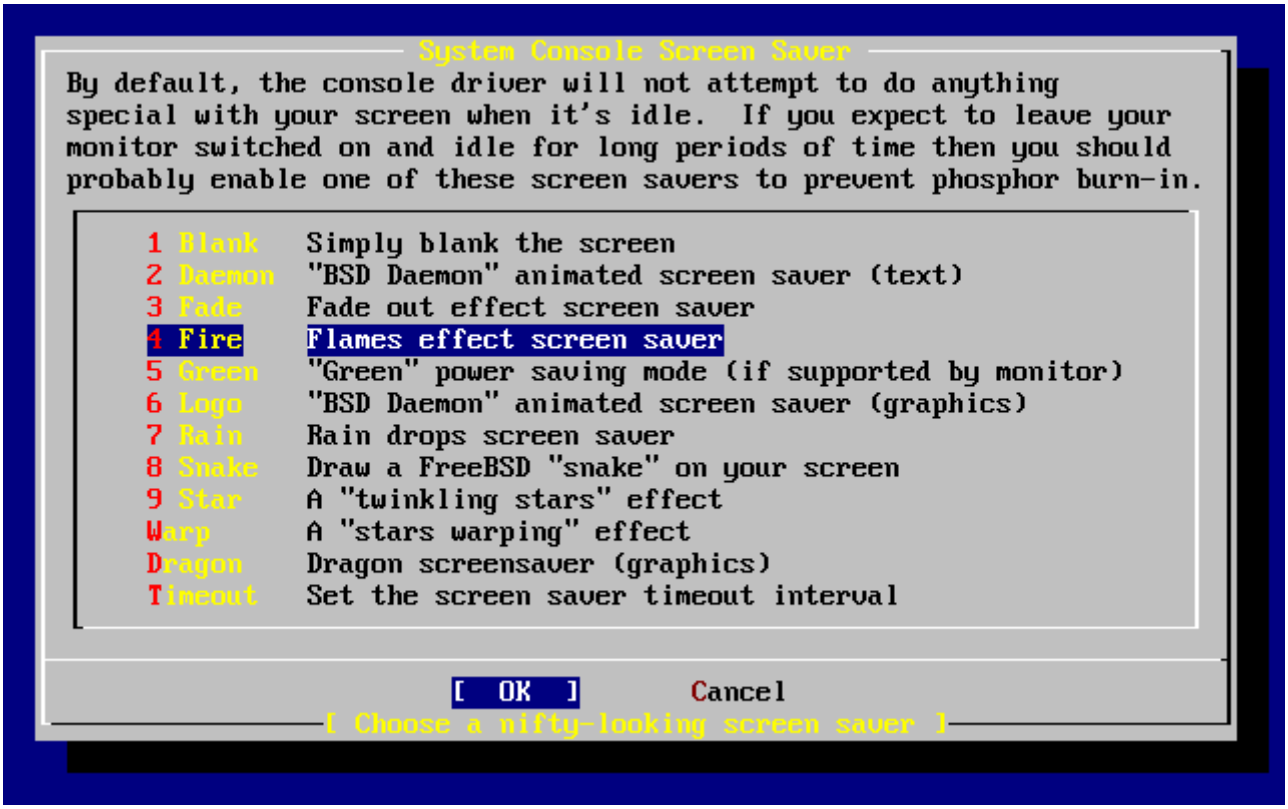
```
                      Felhasználói megerősítés szükséges
                 Testreszabja a rendszerkonzol beállításait?

                                [ Igen ]  Nem
```

A beállítások megtekintéséhez és megváltoztatásához válasszuk a **[yes]** gombot és nyomjuk le az `Enter` billentyût.



Ábra 34. A rendszerkonzol beállításai

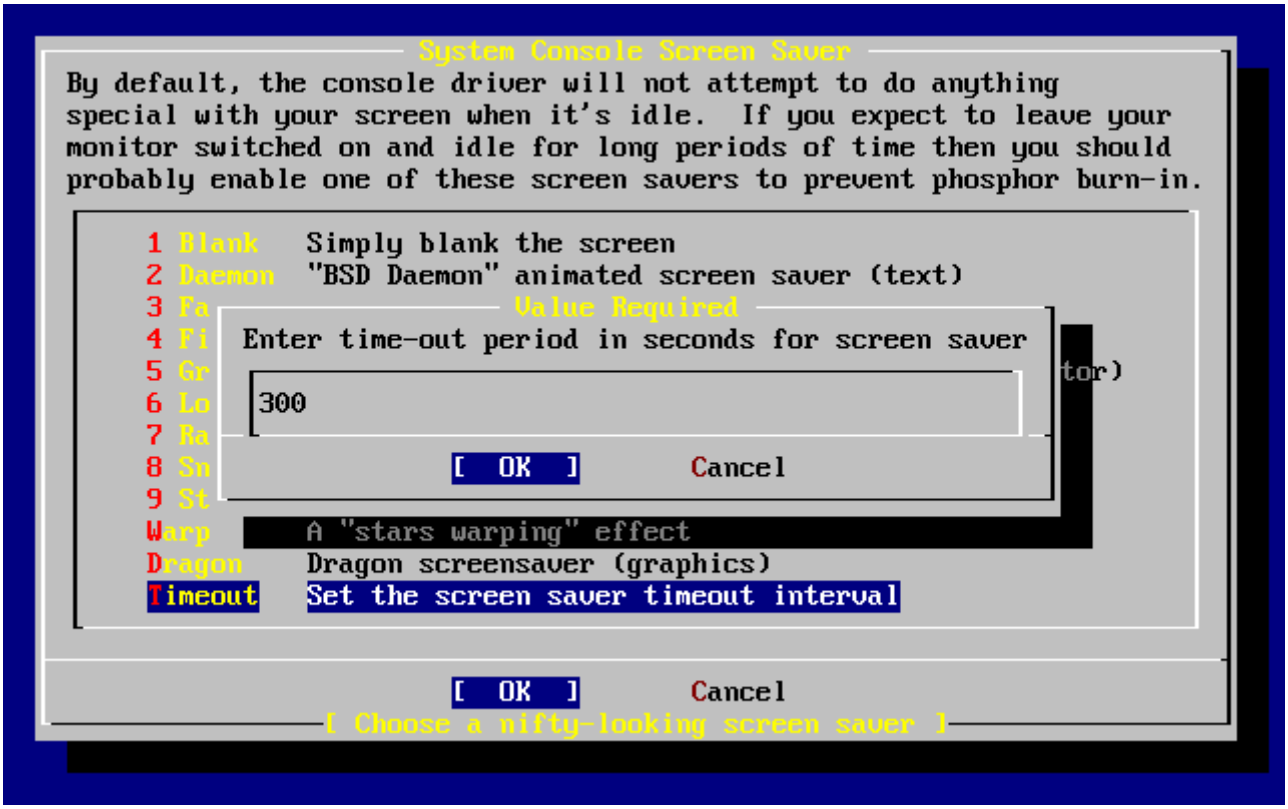
A képernyõkímélõ beállítása egy gyakori opció. A nyilak használatával álljunk a Saver menüpontra, majd nyomjuk le az `Enter` billentyût.



Ábra 35. A képernyõkímélõ beállításai

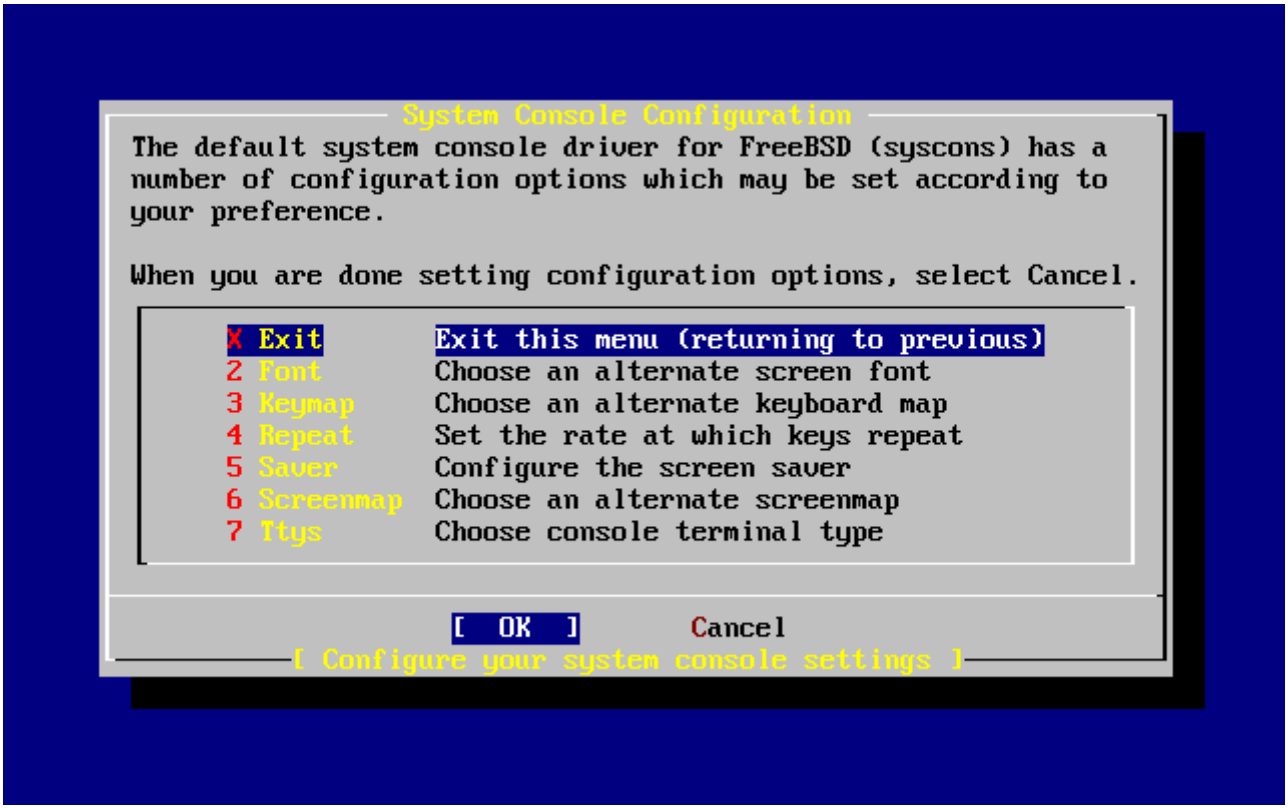
A nyilakkal válasszuk ki a használni kívánt képernyõkímélõt és nyomjuk meg hozzá az `Enter` billentyût. Ekkor a rendszerkonzol beállításait tartalmazó menü jelenik meg ismét.

Az aktivizálódás ideje alapbeállítás szerint 300 másodperc. Ennek megváltoztatásához válasszuk ismét a Saver menüpontot. A képernyõkímélõ beállításait tartalmazó menüben a nyílbillentyûkkel válasszuk a Timeout (Idõkorlát) menüpontot és nyomjuk meg az `Enter` billentyût. Ekkor egy párbeszédablak jelenik meg:



Ábra 36. A képernyõkímélõhöz tartozó idõkorlát beállítása

Miután megváltoztattuk az értéket, a rendszerkonzol beállításához a **[OK]** gomb kiválasztásával, majd az `Enter` billentyû lenyomásával térhetünk vissza.



Ábra 37. Kilépés a rendszerkonzol beállító menüjébõl

A Exit (Kilépés) választásával és az `Enter` lenyomásával folytathatjuk tovább a telepítés utólagos beállításait.

2.10.8. Az idõzóna beállítása

Ha kiválasztjuk számítógépünk számára a megfelelõ idõzónát, akkor lehetõvé tesszük, hogy magától elvégezze a helyi idõhöz kapcsolódó összes szükséges korrekciót és helyesen kezelje az idõzónákhoz kapcsolódó többi funkciót.

A példában az Egyesült Államok keleti idõzónájában elhelyezkedõ számítógépet láthatunk. A mi beállításaink természetesen a saját földrajzi helyzetünktõl függenek.

```
                    User Confirmation Requested
          Would you like to set this machine's time zone now?

                              [ Yes ]   No
```

Fordítás:

```
                      Felhasználói megerősítés szükséges
                 Beállítja most a számítógép időzónáját?

                           [ Igen ]   Nem
```

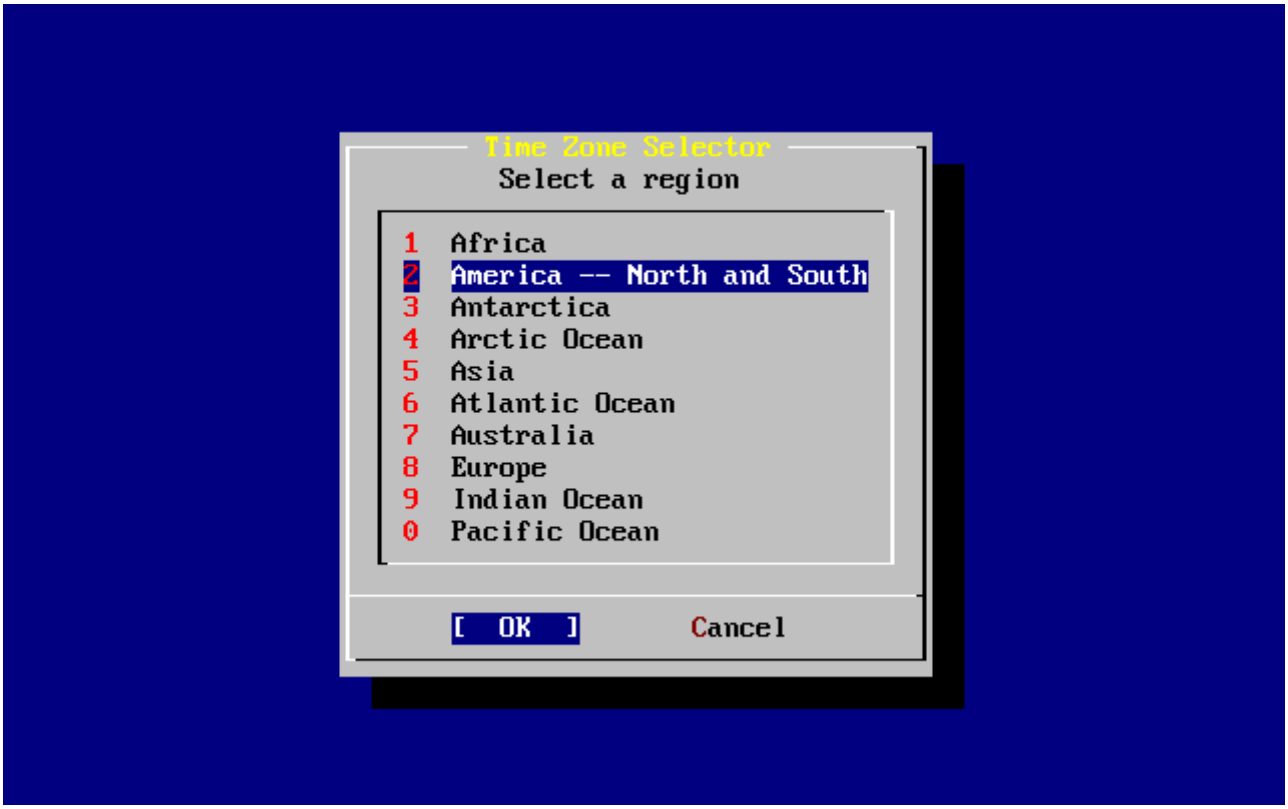
A **[yes]** gomb és az `Enter` billentyû segítségével kiválaszthatjuk az idõzóna beállítását.

```
                       User Confirmation Requested
Is this machine's CMOS clock set to UTC? If it is set to local time
or you don't know, please choose NO here!

                                Yes   [ No ]
```

Fordítás:

```
                     Felhasználói megerősítés szükséges
A számítógép órája az egységes világidőhöz (UTC) van beállítva? Ha a
helyi időhöz vagy nem tudjuk, akkor itt válasszuk a NEM gombot!

                               Igen   [ Nem ]
```

A számítógépünk órájának beállításának megfelelõen válasszuk a **[yes]** vagy **[no]** gombot, és nyomjuk meg az `Enter` billentyût.

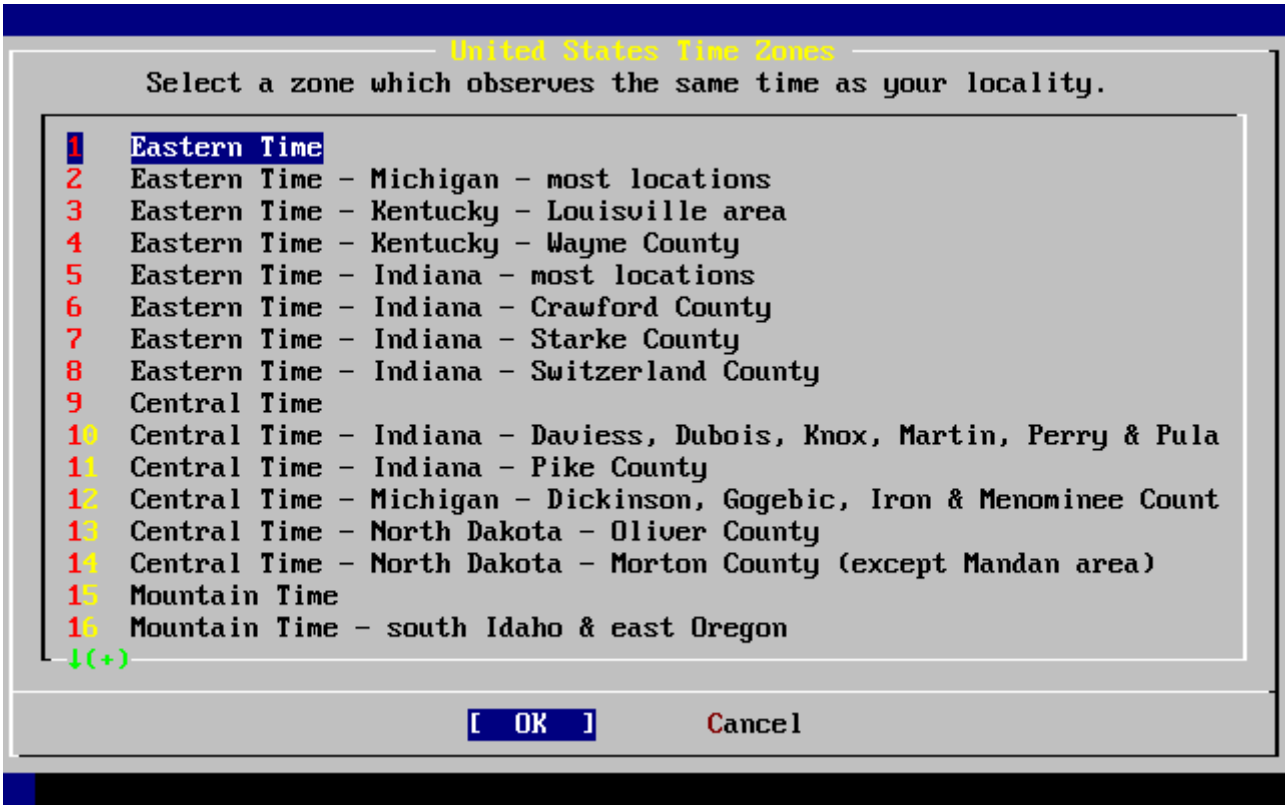


Ábra 38. A térség kiválasztása

A nyilakkal kiválasztható a megfelelõ térség, amit aztán az `Enter` billentyûvel tudunk lezárni.

Ábra 39. Az ország kiválasztása

A megfelelõ ország a nyílbillentyûkkel, valamint az `Enter` billentyûvel választható ki.



Ábra 40. Az idõzóna kiválasztása

A nekünk megfelelõ idõzóna a nyilakkal választható meg, amit ezután az `Enter` billentyûvel tudunk jóváhagyni.

```
                        Confirmation
          Does the abbreviation 'EDT' look reasonable?

                        [ Yes ]   No
```

Az üzenet fordítása:

```
                        Megerősítés
                Ezek szerint az 'EDT' elfogadható?

                        [ Igen ]   Nem
```

Erõsítsük meg, hogy az idõzóna helyes-e. Ha rendbenlevõnek látszik, nyomjuk meg az `Enter` billentyût a folytatáshoz.

2.10.9. Linux binárisok használata

Ez a rész csak a FreeBSD 7.*X* telepítésére vonatkozik, FreeBSD 8.*X* esetén ez a képernyõ nem jelenik meg.

```
                    User Confirmation Requested
          Would you like to enable Linux binary compatibility?
```

```
                          [ Yes ]   No
```

A fordítás:

```
                    Felhasználói megerősítés szükséges
               Engedélyezi a Linux binárisok futtatását?

                          [ Igen ]   Nem
```

A **[yes]** gomb kiválasztásával és az `Enter` lenyomásával megengedjük, hogy a Linuxra készült szoftvereket futtassunk FreeBSD-n. A telepítõ ennek biztosításához még további csomagokat is fel fog rakni.

Ha FTP-n keresztül telepítünk, akkor a számítógépnek csatlakoznia kell az internetre. Ilyenkor elõfordulhat, hogy az FTP szerveren nem találhatóak meg a Linux® kompatibilitással kapcsolatos csomagok. Ezeket azonban késõbb is telepíthetjük.

2.10.10. Az egér beállításai

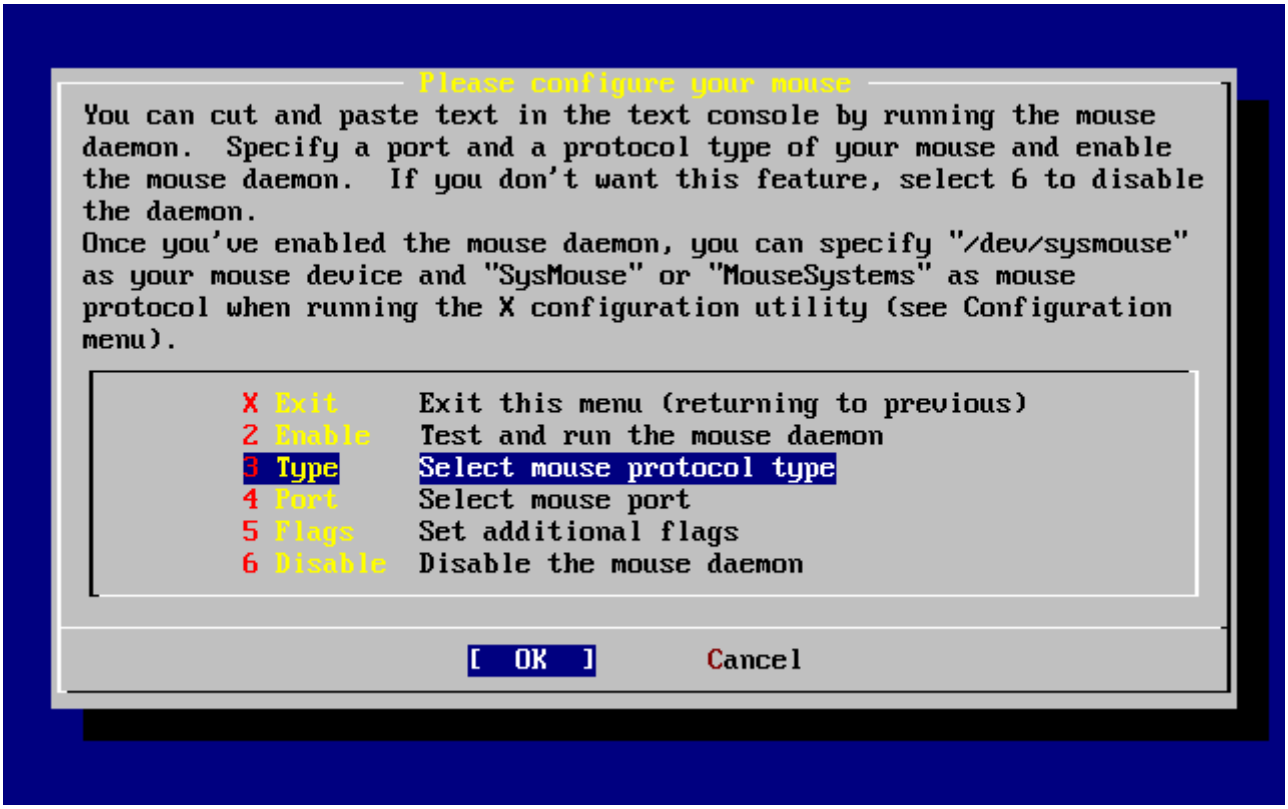
Ezen beállítás használatával egy háromgombos egérrel lehetõségünk adódik a konzol és a felhasználói programok között kivágni és bemásolni szövegeket. Kétgombos egér használata esetén nézzük meg a moused(8) man oldalán, miként tudjuk emulálni a háromgombos mûködést. A következõ példa egy nem USB-s (tehát PS/2-es vagy soros portra csatlakozó) egér beállítását illusztrálja:

```
                     User Confirmation Requested
          Does this system have a PS/2, serial, or bus mouse?

                          [ Yes ]    No
```

Fordítás:

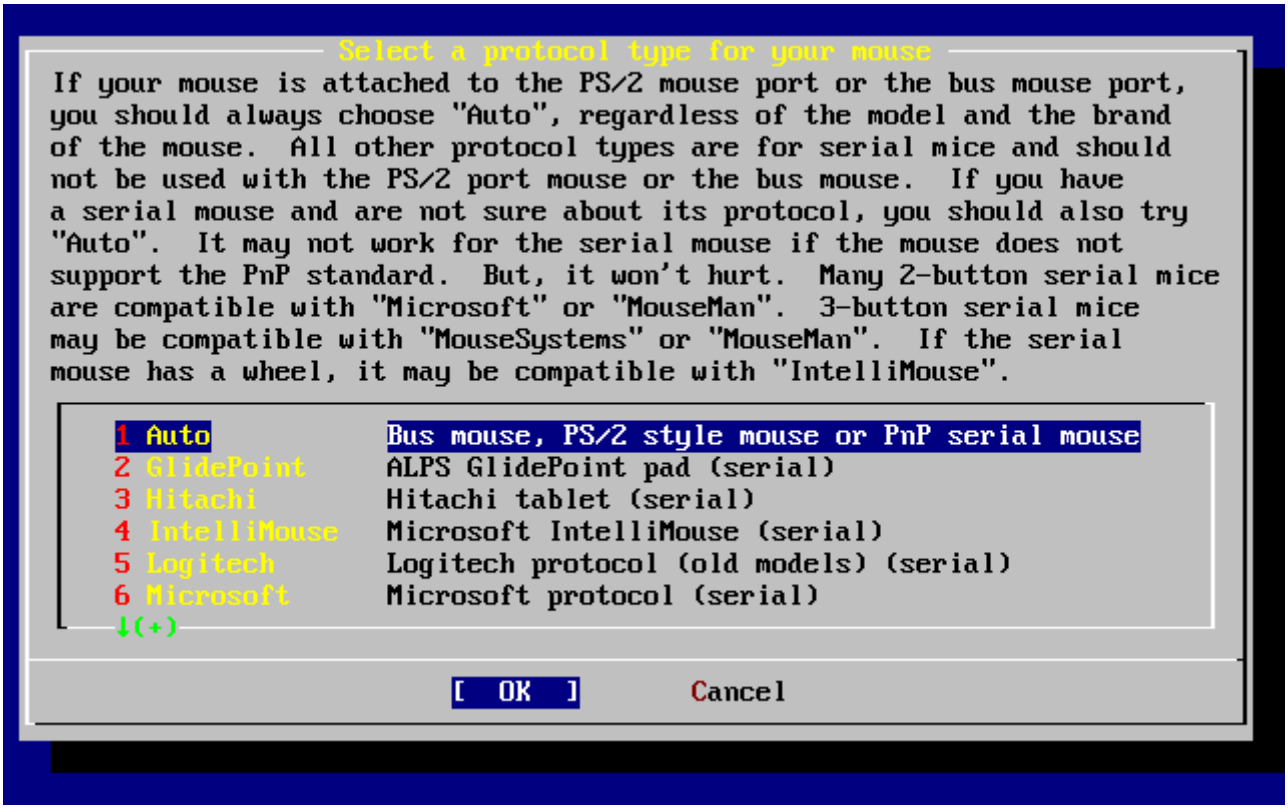
```
                   Felhasználói megerősítés szükséges
        Csatlakozik a rendszeréhez PS/2-es, soros vagy buszos egér?

                          [ Igen ]    Nem
```

A PS/2, soros vagy buszos egér használatához válasszuk a **[yes]** gombot, illetve az USB-s egérhez pedig a **[no]** gombot, majd nyomjuk meg az `Enter` billentyût.



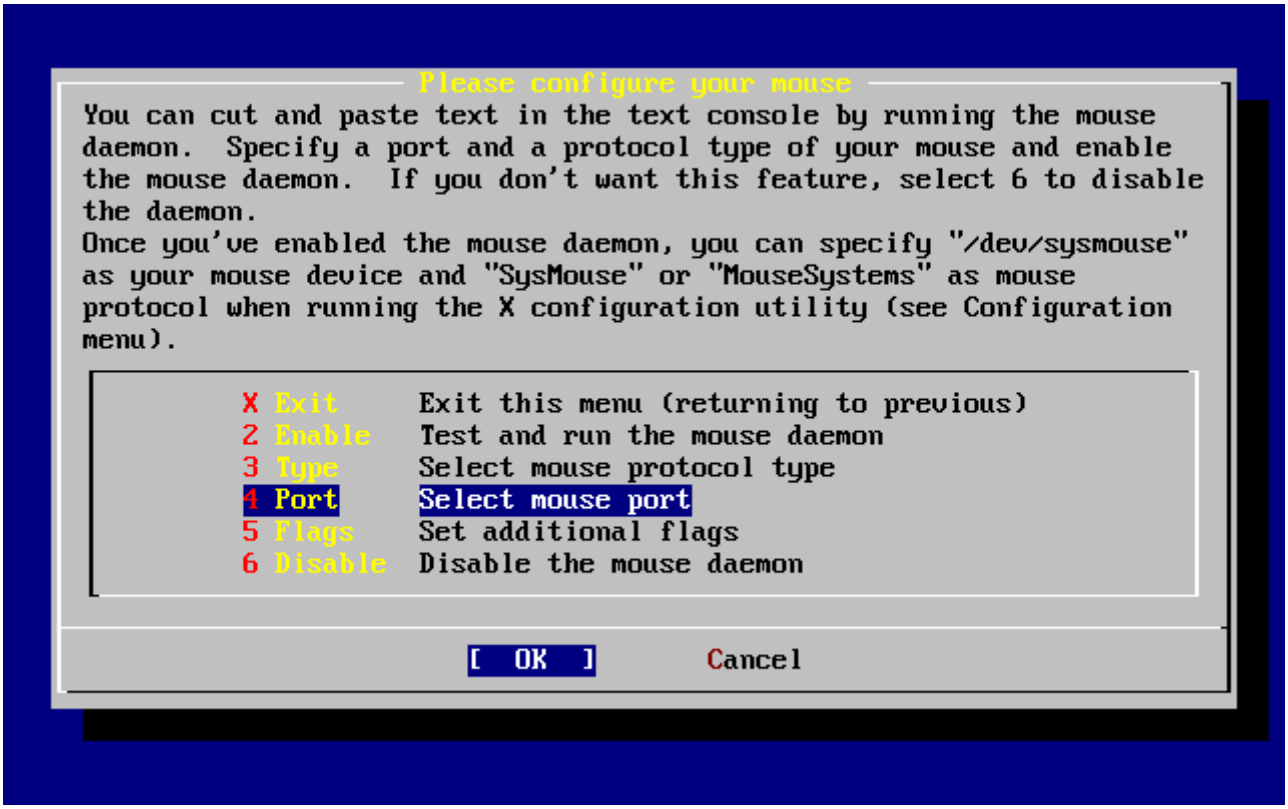
Ábra 41. Az egér által használt protokoll típusának beállítása

A nyílbillentyûk használatával keressük ki a Type (Típus) menüpontot és nyomjuk le az `Enter` billentyût.



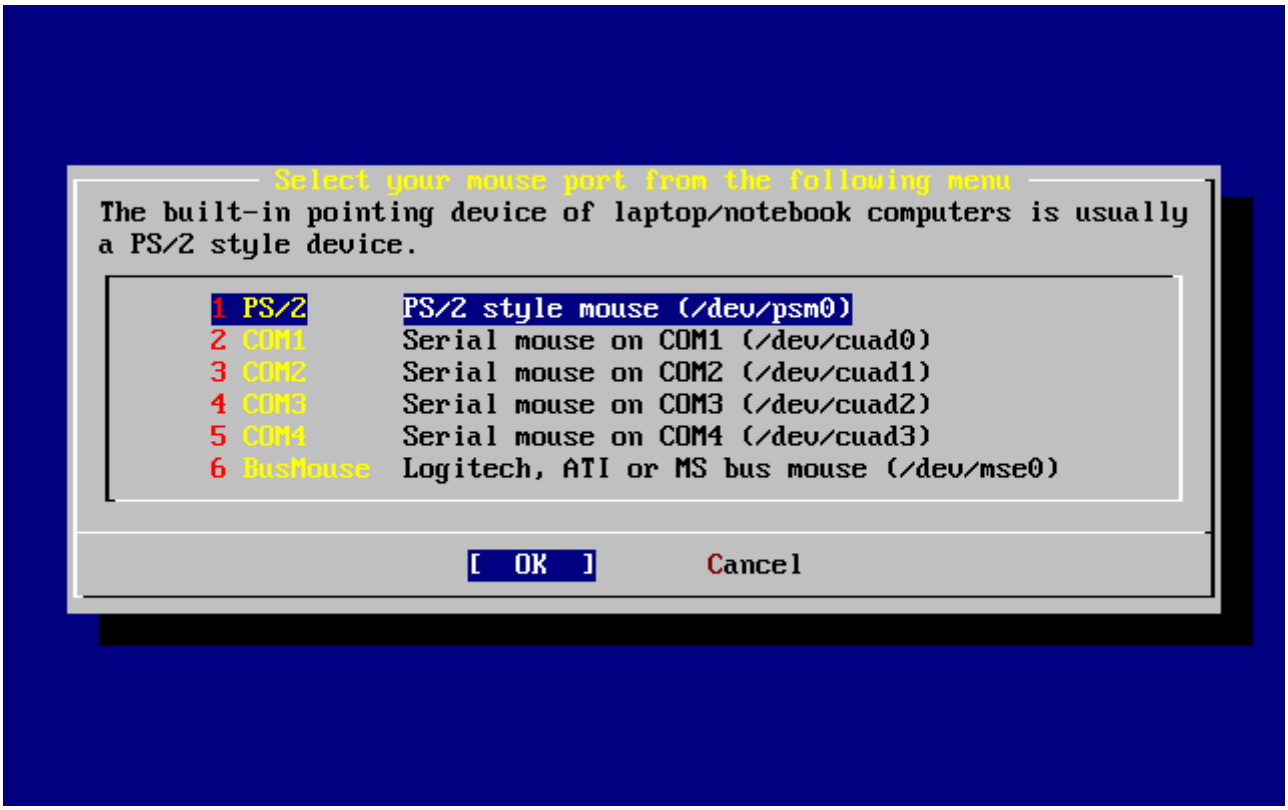
Ábra 42. Az egér protokolljának beállítása

A példában használt egér típusa PS/2, ezért itt a alapértelmezés szerint felkínált Auto megfelelõ. A protokoll megváltoztatásához a nyilakkal válasszunk ki egy másikat. Ezután gondoskodjunk róla, hogy az **[OK]** gombot választottuk ki és a kilépéshez nyomjuk meg az `Enter` billentyût.



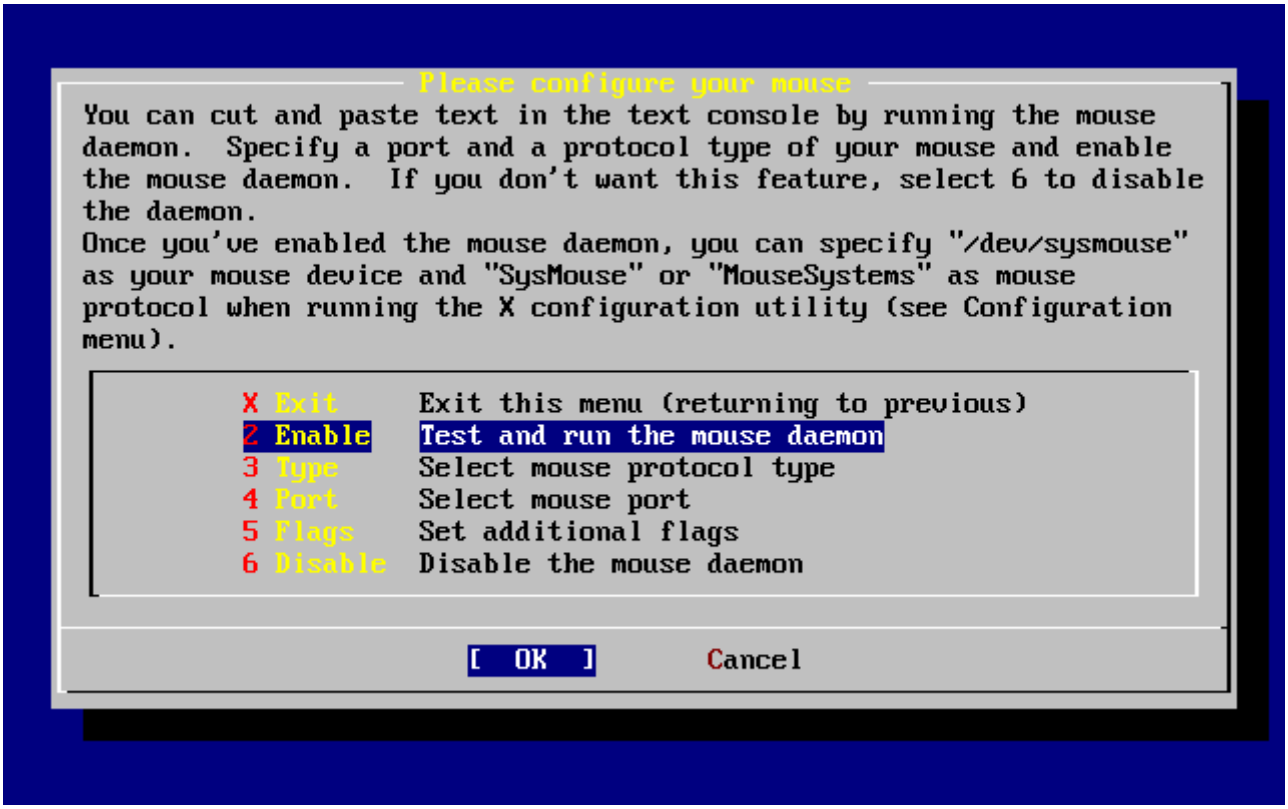
Ábra 43. Az egér portjának beállítása

A nyílbillentyûkkel válasszuk ki a Port menüpontot és nyomjuk meg az `Enter` billentyût.



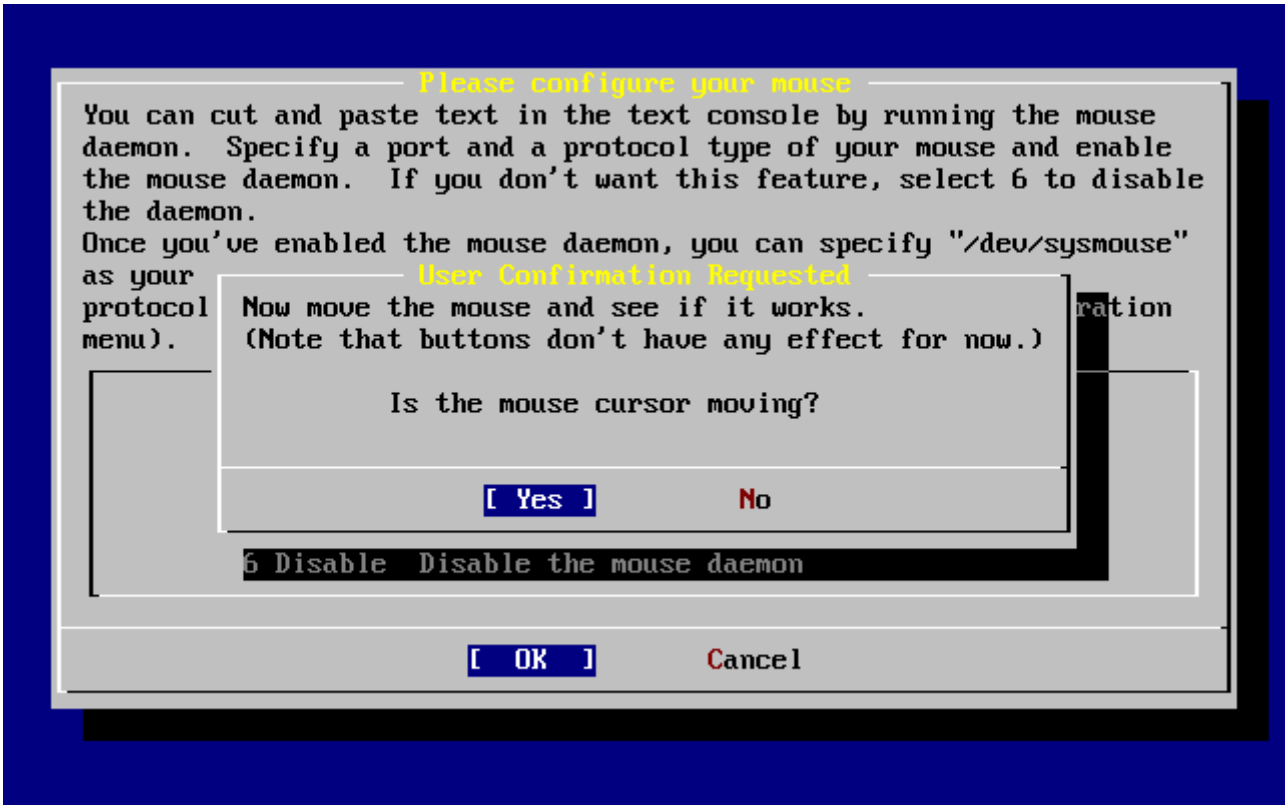
Ábra 44. Az egér portjának kiválasztása

Mivel a példában szereplõ rendszerhez egy PS/2 egér csatlakozik, ezért az alapértelmezett PS/2 menüpont megfelelõnek tûnik. A port megváltoztatásához használjuk a nyilakat, majd nyomjuk le az `Enter` billentyût.



Ábra 45. Az egérdémon engedélyezése

Befejezésül a egérhez tartozó démon aktiválásához és kipróbálásához válasszuk ki a nyilakkal az Enable (Engedélyezés) menüpontot.



Ábra 46. Az egérdémon kipróbálása

Próbáljuk mozgatni a képernyõn megjelenõ egérkurzort, és ellenõrizzük, hogy a kurzor a mozdulatainknak megfelelõen reagál-e. Ha mindent rendben találunk, akkor válasszuk a **[yes]** gombot és nyomjuk le az `Enter` billentyût. Ellenkezõ esetben az egeret nem jól állítottuk be -

válasszuk a **[no]** gombot és kísérletezzünk tovább más beállításokkal.

Az utólagos beállítások folytatásához válasszuk elõször az Exit (Kilépés) menüpontot, majd nyomjuk meg az `Enter` billentyût.

2.10.11. Csomagok telepítése

A csomagok elõre lefordított binárisokat tartalmaznak, és használatukkal igen kényelmesen tudunk szoftvereket telepíteni.

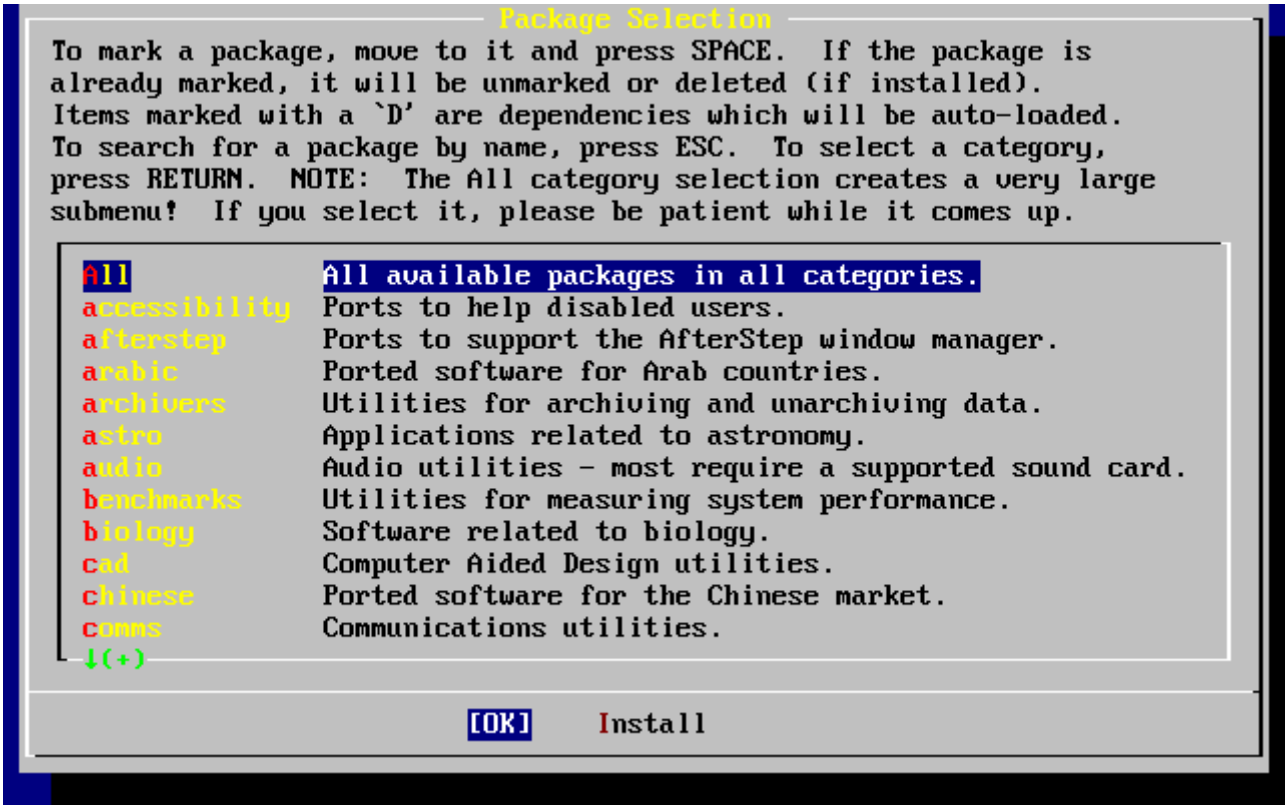
Szemléltetés céljából most bemutatjuk az egyik ilyen csomag telepítését. Természetesen igény szerint más csomagokat is hozzávehetünk. A telepítés után a `sysinstall` parancs használható további csomagok telepítésére.

```
                    User Confirmation Requested
The FreeBSD package collection is a collection of hundreds of
ready-to-run applications, from text editors to games to WEB servers
and more. Would you like to browse the collection now?

                           [ Yes ]   No
```

Az üzenet fordítása:

```
                Felhasználói megerősítés szükséges
A FreeBSD csomaggyűjteménye többezernyi azonnal használható
alkalmazást tartalmaz, a szövegszerkesztőktől a játékokon keresztül a
WEBszervereken át szinte mindent. Át kívánja lapozni most ezt a
gyűjteményt?

                          [ Igen ]   Nem
```

A **[yes]** kiválasztása és az `Enter` lenyomása után a csomagválasztó képernyõ következik:

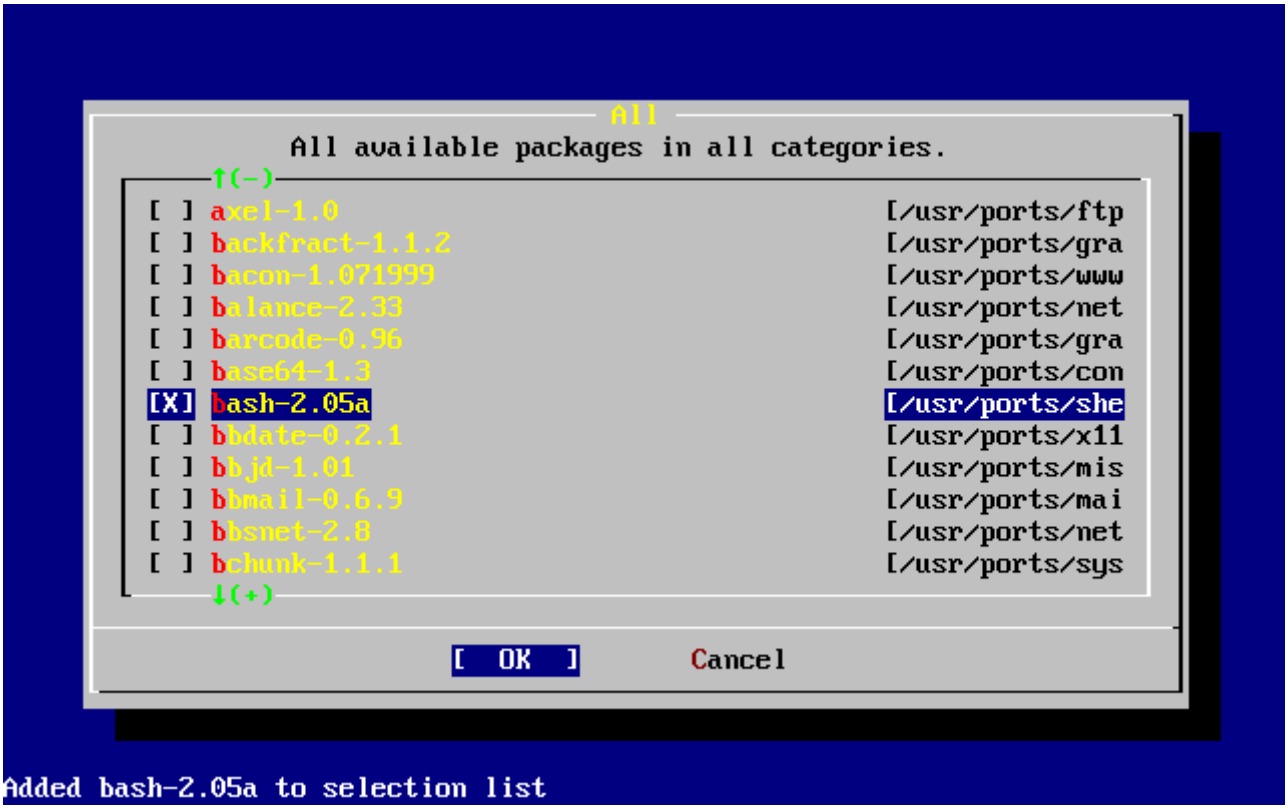


Ábra 47. A csomagok kategóriájának kiválasztása

Ekkor csak az adott telepítõeszközön elérhetõ csomagok fognak megjelenni.

Az összes csomagot az All (Mind) menüpont kiválasztásával láthatjuk, vagy leszûkíthetjük ezt egy adott kategóriára is. Álljunk a kiválasztott kategóriához tartozó menüpontra és nyomjuk meg az `Enter` billentyût.

Ezután egy menü fogja felsorolni az adott kategórián belül telepíthetõ csomagokat:



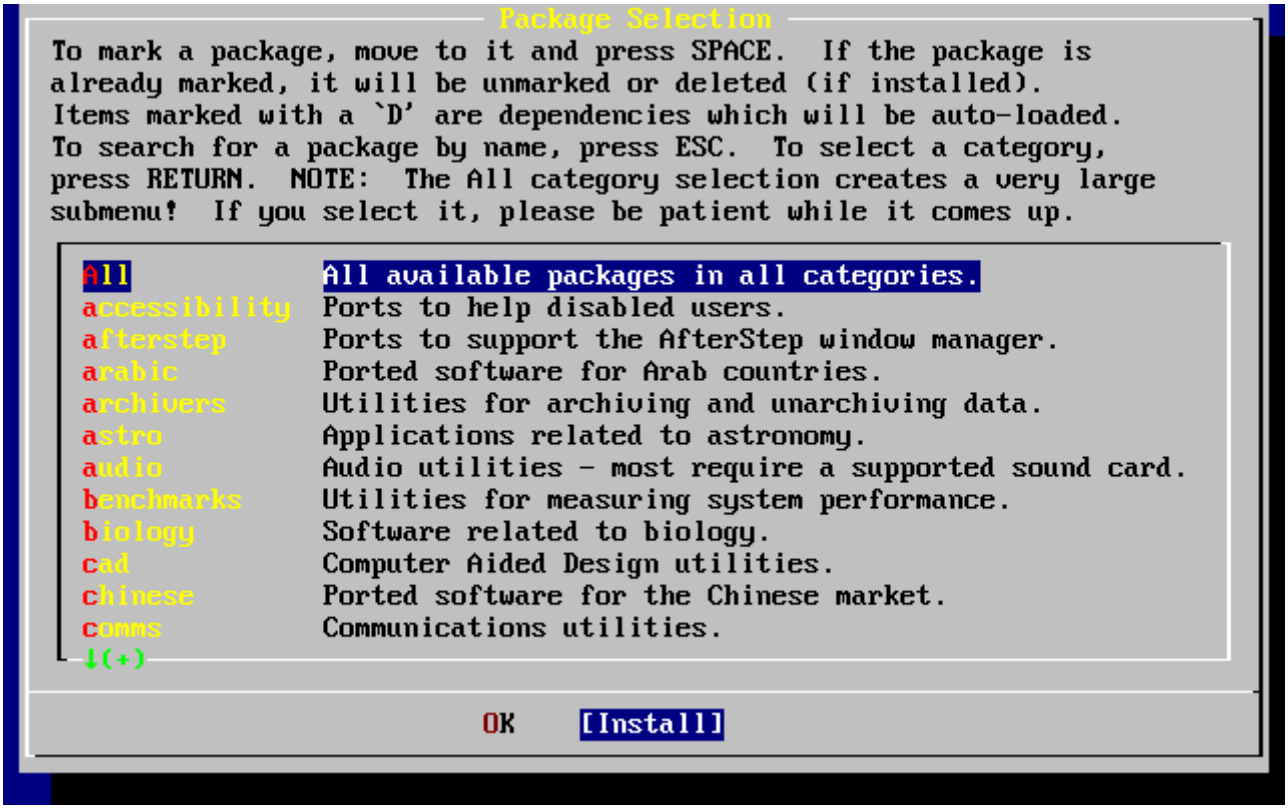
Ábra 48. Csomag kiválasztása

A példában a bash parancsértelmezõt választottuk ki. Válogassunk kedvünkre a csomagok között, és álljunk a telepíteni kívántakra, majd a `Szóköz` billentyû lenyomásával jelöljük be ezeket. Minden egyes csomag rövid leírása a képernyõ bal alsó sarkában olvasható.

A `Tab` billentyû segítségével mozoghatunk az utoljára kiválasztott csomag, az **[OK]** és **[Cancel]** gombok között.

Miután bejelöltük az összes telepítésre szánt csomagot, a csomagválasztó menübe úgy tudunk visszatérni, ha a `Tab` billentyûvel átváltunk az **[OK]** gombra és nyomjuk meg az `Enter` billentyût.

Ezeken felül a bal és jobb nyilak használhatóak az **[OK]** és **[Cancel]** gombok közti váltásra. Ugyanezzel a módszerrel választható ki az **[OK]** gomb is, ami után az `Enter` billentyû megnyomásával visszajutunk a csomagválasztó menübe.



Ábra 49. Csomagok telepítése

A nyilakkal és a `Tab` billentyûvel válasszuk ki az **[Install]** (Telepítés) gombot és nyomjuk meg az `Enter` billentyût. Ekkor meg kell erõsítenünk a csomagok telepítését:

Ábra 50. Csomagok telepítésének megerõsítése

Az **[OK]** kiválasztása majd az `Enter` billentyû lenyomása indítja el a csomagok telepítését. A telepítés befejezéséig különbözõ üzenetek fognak megjelenni. Figyeljünk az ilyenkor felbukkanó hibaüzenetekre!

A beállítások véglegesítése a csomagok telepítése után folytatódik. Amennyiben egyetlen csomagot sem választottunk és szeretnénk továbblépni, akkor is az **[Install]** (Telepítés) gombot válasszuk.

2.10.12. Felhasználók és csoportok felvétele

A telepítés során legalább egy felhasználót érdemes hozzáadnunk a rendszerhez, mivel a rendszer használatához így nem kell `root` felhasználóként bejelentkezni. Általánosságban véve ahhoz egyébként is kicsi a gyökérpartíció, hogy `root` felhasználóként (rendszeradminisztrátorként) futtassunk rajta programokat, és gyorsan be is telik. A nagyobb veszélyt azonban itt olvashatjuk:

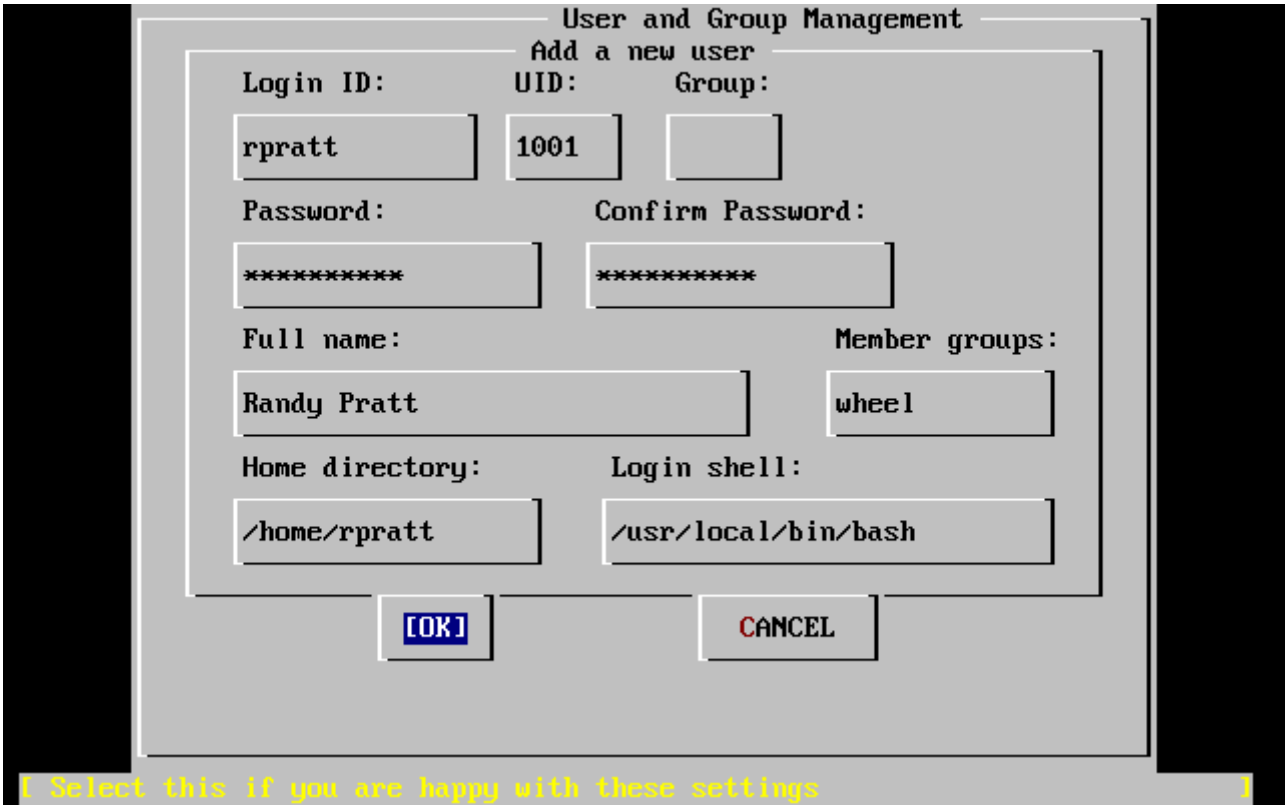
```
                    User Confirmation Requested
Would you like to add any initial user accounts to the system? Adding
at least one account for yourself at this stage is suggested since
working as the "root" user is dangerous (it is easy to do things which
adversely affect the entire system).

                           [ Yes ]   No
```

```
                 Felhasználói megerősítés szükséges
Szeretnénk mosta rendszerbe felvenni felhasználói fiókokat? Ebben a
lépésben legalább egy felhasználó felvétele javasolt, hiszen "root"
felhasználóként veszélyes dolgozni (mivel így könnyen tehetünk olyan
dolgokat, amelyek káros hatással lehetnek rendszerünkre).

                          [ Igen ]   Nem
```

Ezért válasszuk a **[yes]** gombot és az `Enter` billentyû lenyomásával lépjünk tovább a felhasználók felvételéhez.

Ábra 51. Felhasználók kiválasztása

A nyílbillentyûkkel válasszuk ki a User (Felhasználó) menüpontot és nyomjuk meg az `Enter` billentyût.



Ábra 52. A felhasználó adatainak megadása

Amikor a `Tab` billentyûvel lépkedünk a kitöltendõ mezõk között, a képernyõ alsó részén az alábbi leírások magyarázzák az egyes mezõk tartalmát:

Login ID (Bejelentkezési azonosító)

Az új felhasználó bejelentkezési neve (kötelezõ).

UID (Felhasználói azonosító)

A felhasználó számszerû azonosítója (automatikusan létrejön, ha üresen hagyjuk).

Group (Csoport)

A felhasználó bejelentkezési csoportjának neve (automatikusan létrejön, ha üresen hagyjuk).

Password (Jelszó)

A felhasználó jelszava (óvatosan bánjunk ezzel a mezõvel!)

Full name (Teljes név)

A felhasználó teljes neve (megjegyzés).

Member groups (További csoportok)

A felhasználó ezen csoportoknak is tagja (tehát rendelkezik az engedélyeikkel).

Home directory (Felhasználói könyvtár)

A felhasználó saját könyvtára (ha üresen hagyjuk, az alapértelmezés szerint töltõdik ki).

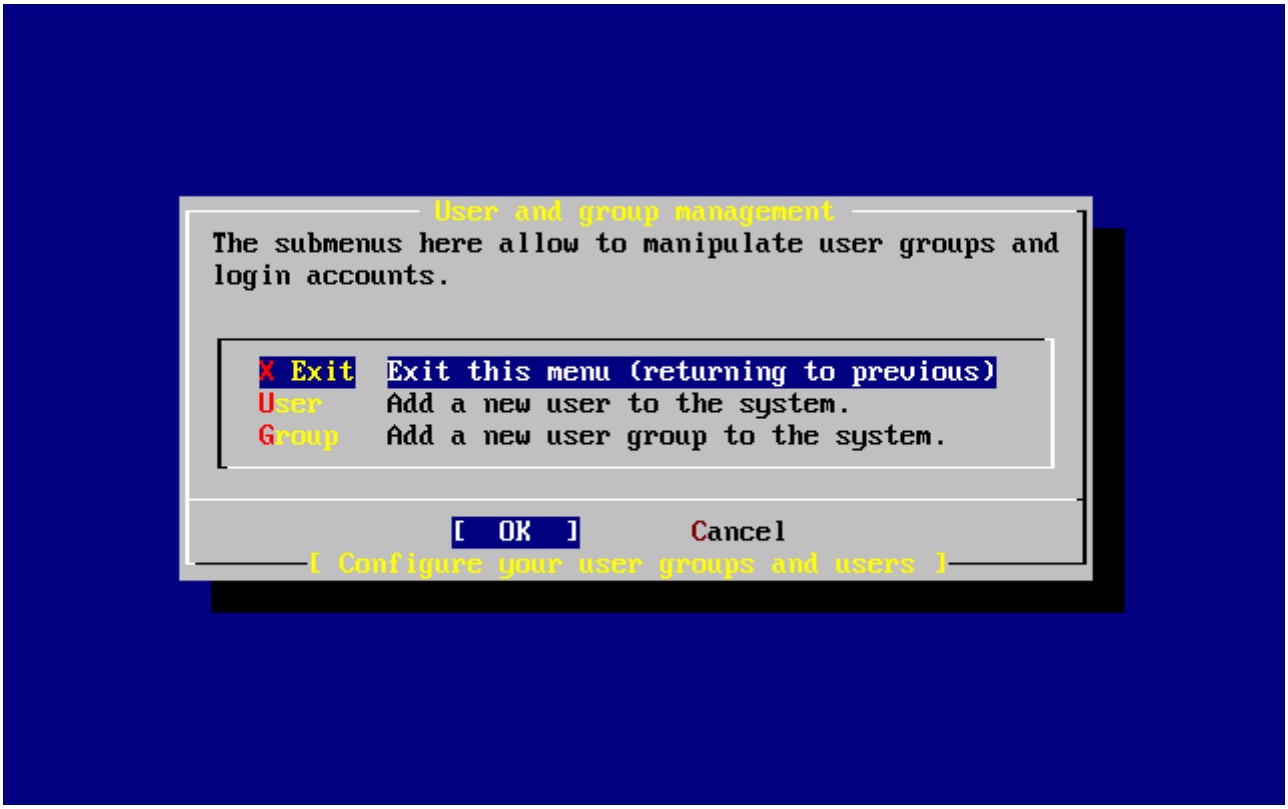
Login shell (Parancsértelmezõl)

A felhasználó által használt parancsértelmezõ (ha üresen hagyjuk, az alapértelmezés szerint töltõdik, mint például /bin/sh).

Az ábrán a bejelentkezés után használt parancsértelmezõt a /bin/sh parancsértelmezõrõl a /usr/local/bin/bash parancsértelmezõre változtattuk, így most a korábban telepített bash parancsértelmezõt fogjuk használni. Itt ne is próbáljunk nem létezõ parancsértelmezõt kiválasztani, hiszen ekkor nem tudunk majd bejelentkezni. A BSD világban egyébként a C shell a leggyakrabban használt, amelyet a /bin/tcsh megadásával választhatjuk ki.

Az ábrán szereplõ felhasználót ezenkívül még a `wheel` csoportba is felvettük, aminek köszönhetõen képes lesz a rendszerünkben a `root` felhasználói jogaival rendelkezõ rendszeradminisztrátorrá válni.

Amikor mindent megfelelõnek találunk, nyomjunk az **[OK]** gombra és ekkor ismét a felhasználók és csoportok karbantartását tartalmazó menü jelenik meg:



Ábra 53. Kilépés a felhasználók és csoportok menüjébõl

Csoportokat is létre tudunk hozni, amennyiben erre szükségünk lenne. Ez a rész a telepítés befejezése után továbbra is elérhetõ a `sysinstall` parancs segítségével.

Amikor befejeztük a felhasználók hozzáadását, a nyilakkal válasszuk ki az Exit (Kilépés) menüpontot és a telepítés folytatásához nyomjuk meg az `Enter` billentyût.

2.10.13. A `root` felhasználó jelszavának megadása

```
                      Message
Now you must set the system manager's password.
This is the password you'll use to log in as "root".

                       [ OK ]

              [ Press enter or space ]
```

Fordítása:

```
                      Üzenet
Most meg kell adnia a rendszergazda jelszavát. Ezt a jelszót
kell a "root" felhasználó bejelentkezésekor használni.

                       [ OK ]

    [ Nyomja le az Enter vagy a Szóköz billentyűt ]
```

A `root` felhasználó jelszavának beállításához nyomjuk meg az `Enter` billentyût.

A jelszót kétszer kell megadnunk. Felesleges megemlíteni, hogy gondoskodjunk arról az esetrõl is, ha véletlenül elfelejtenénk ezt a jelszót. Megemlítjük, hogy az itt begépelt jelszó nem lesz látható és a betûk helyett sem jelennek meg csillagok.

```
New password:
Retype new password :
```

A jelszó sikeres megadása után a telepítés folytatódik.

2.10.14. Kilépés a telepítõbõl

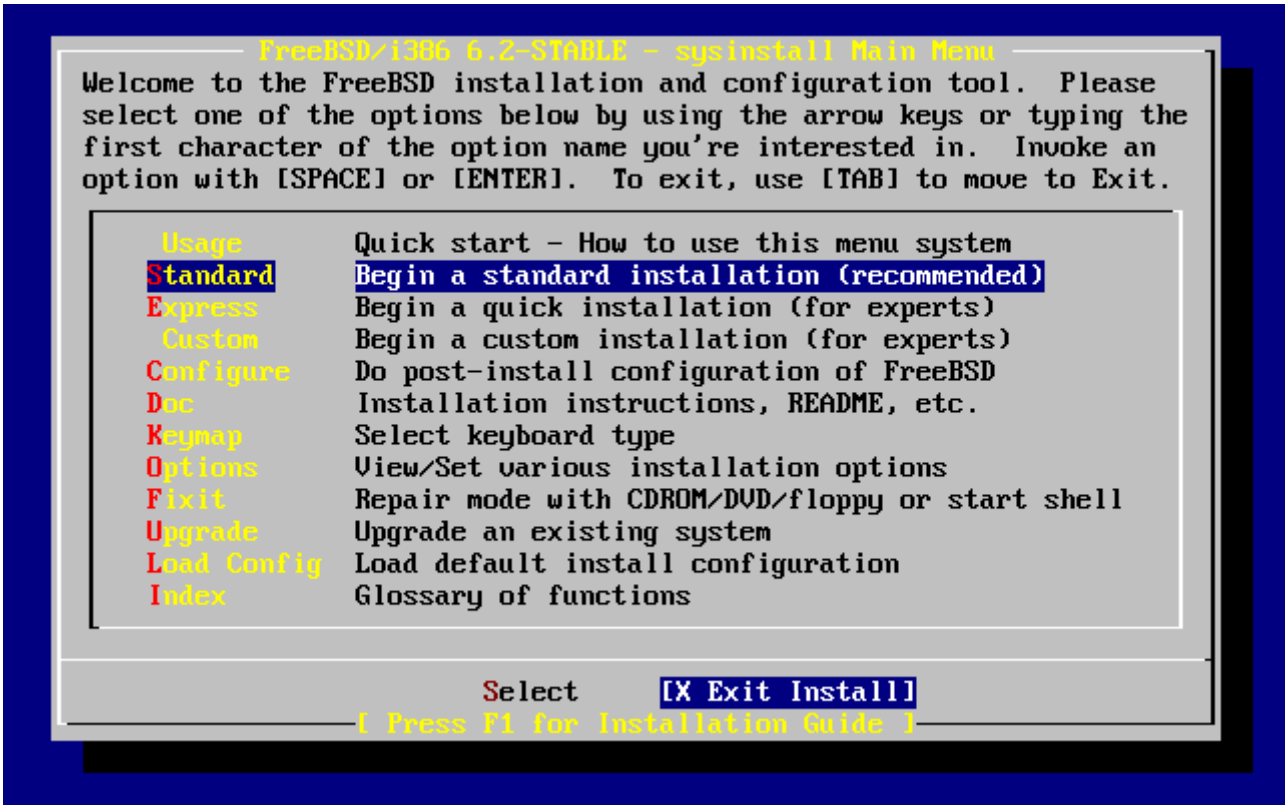
Ha be szeretnénk még állítani egyéb hálózati szolgáltatást vagy valamilyen más konfigurációs lépést kívánunk még elvégezni, ezen a ponton megtehetjük vagy a telepítés után a `sysinstall` parancs kiadásával.

```
                    User Confirmation Requested
Visit the general configuration menu for a chance to set any last
options?

                              Yes   [ No ]
```

Fordítás:

```
                 Felhasználói megerősítés szükséges
Végignézi még utoljára a beállításokat arra az esetre, ha véletlenül
kihagytunk volna valamit?

                             Igen   [ Nem ]
```

Ha a nyilakkal a **[no]** gombot választjuk, majd megnyomjuk rajta az `Enter` billentyût, akkor visszatérünk a telepítõ fõmenüjébe.



Ábra 54. Kilépés a telepítõbõl

Válasszuk ki a nyílbillentyûkkel a **[X Exit Install]** (Kilépés a telepítõbõl) gombot és nyomjuk meg az `Enter` billentyût. Ezután meg kell erõsítenünk kilépési szándékunkat:

```
                    User Confirmation Requested
Are you sure you wish to exit? The system will reboot.

                           [ Yes ]   No
```

Fordítás:

```
                Felhasználói megerősítés szükséges
          Valóban ki akar lépni? A rendszer ezt követően újra fog
          indulni!

                          [ Igen ]    Nem
```

Válasszuk a **[yes]** gombot. Ha CD-meghajtóról indítottuk a telepítést, akkor a következõ üzenet fog figyelmeztetni minket a lemez kivételére:

```
                      Message
Be sure to remove the media from the drive.

                     [ OK ]
            [ Press enter or space ]
```

Fordítás:

```
                     Üzenet
Ne felejtsük el kivenni a CD-lemezt a meghajtóból.

                    [ OK ]
        [ Nyomjunk Entert vagy szóközt ]
```

A CD-meghajtó egészen az újraindítás megkezdéséig zárolt lesz, ezért csak ekkor tudjuk (gyorsan) kivenni a meghajtóból a lemezt. Nyomjuk meg az **[OK]** gombot az újraindításhoz.

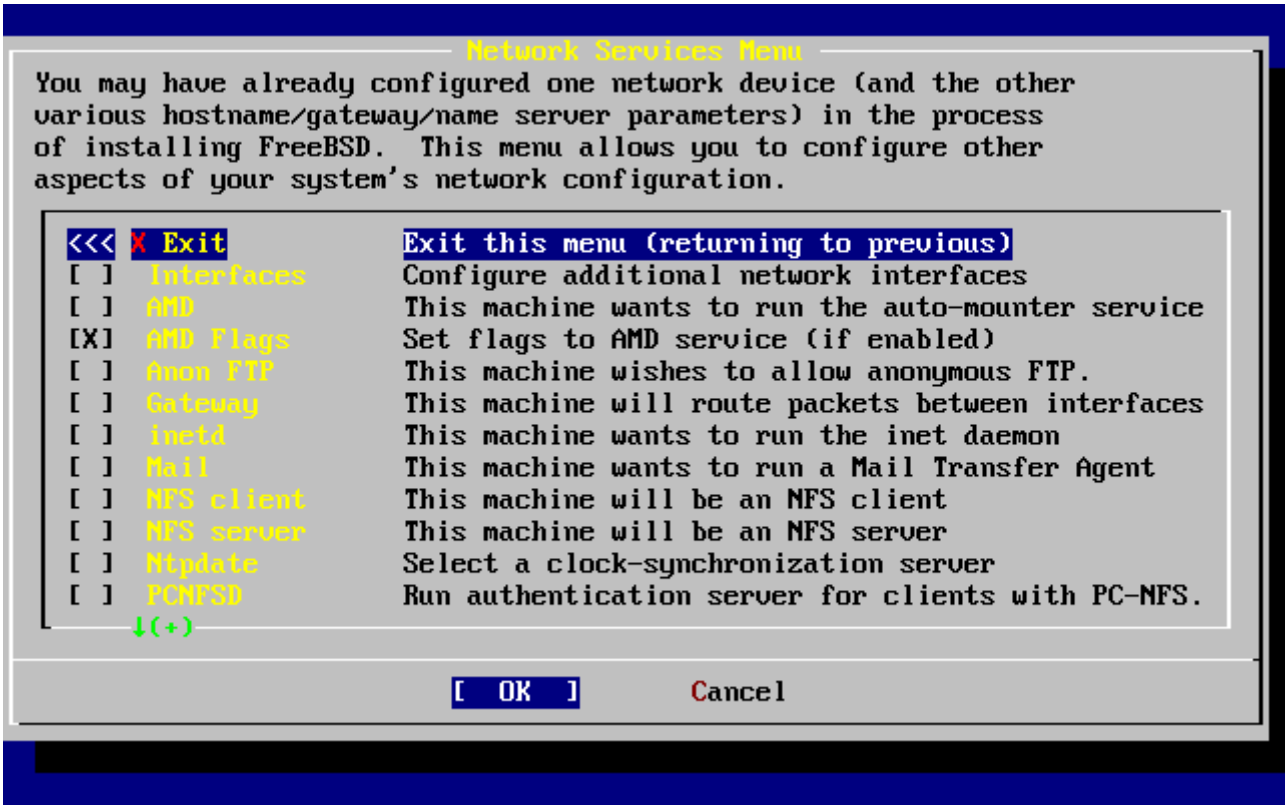
A rendszer újraindul, legyünk résen és figyeljük a megjelenõ hibaüzeneteket, errõl bõvebben lásd a A FreeBSD indulásaban.

2.10.15. További hálózati szolgálatások beállítása

A hálózati szolgáltatások terén csekély tapasztalattal rendelkezõ kezdõ felhasználók számára ijesztõ lehet ezek beállítása. A hálózatok és többek közt az internet kezelése napjaink modern operációs rendszereink, így a FreeBSD-nek is az egyik fontos területe. Ezért nagyon hasznos ismernünk valamennyire a FreeBSD által felkínált hálózati lehetõségeket. A telepítés közben ezért a felhasználónak tisztában kell lennie a rendelkezésére álló szolgáltatásokkal.

A hálózati szolgáltatások olyan programok, amelyek a hálózat minden részérõl fogadnak adatokat. Mindent el kell követnünk annak érdekében, hogy ezek a programok ne tehessenek semmilyen "kárt". Sajnos a programozók sem tökéletesek, és az idõk során már elõfordult párszor, hogy a hálózati szolgáltatásokban maradtak hibák, amelyek kihasználásával a támadók rossz dolgokat tudtak csinálni. Ezért fontos, hogy csak is azokat a szolgáltatásokat engedélyezzük, amelyekre ténylegesen szükségünk van. Ha nem tudjuk eldönteni, akkor az a legjobb, ha egészen addig egyiket sem engedélyezzük, amíg valóban szükségünk nem lesz rájuk. A sysinstall újbóli elindításával vagy az /etc/rc.conf megfelelõ beállításával mindig tudunk új szolgáltatásokat aktiválni.

A Networking (Hálózatok) menüpont kiválasztása után valami ilyesmit láthatunk:



Ábra 55. A hálózati beállítások menüjének felsõ szintje

Ezek közül a Interfaces (Csatolók), vagyis az elsõ menüpontról korábban már szó esett a A hálózati eszközök beállításaban, ezért ez most nyugodtan kihagyható.

Az AMD menüpont kiválasztásával engedélyezzük a BSD automatikus csatlakoztatásokért felelõs segédeszközét (AMD, az AutoMounter Daemon). Ezt általában az NFS protokollal (lásd lentebb) együtt szokás használni a távoli állományrendszerek automatikus csatlakoztatásához. Itt nincs szükség semmilyen különleges beállításra.

A következõ sorban az AMD Flags (Az AMD beállításai) menüpont szerepel. Kiválasztása után az AMD beállításait bekérõ ablak fog felbukkani. Ez már számos alapértelmezett beállítást tartalmaz:

```
-a /.amd_mnt -l syslog /host /etc/amd.map /net /etc/amd.map
```

A `-a` kapcsolóval adjuk meg a csatlakozási pontok alapértelmezett helyét, amely ebben az esetben az /.amd_mnt. A `-l` kapcsolóval adjuk meg az alapértelmezett log (napló) állományt, habár a `syslogd` használata során az összes naplózási tevékenység a rendszer naplózó démonján fut majd keresztül. A /host könyvtárba fognak csatlakozni a távoli gépek exportált állományrendszerei, míg a /net könyvtárba a különbözõ IP-címekrõl exportált állományrendszerek kerülnek csatlakoztatásra. Az /etc/amd.map állomány tartalmazza az AMD exportjainak alapértelmezett beállításait.

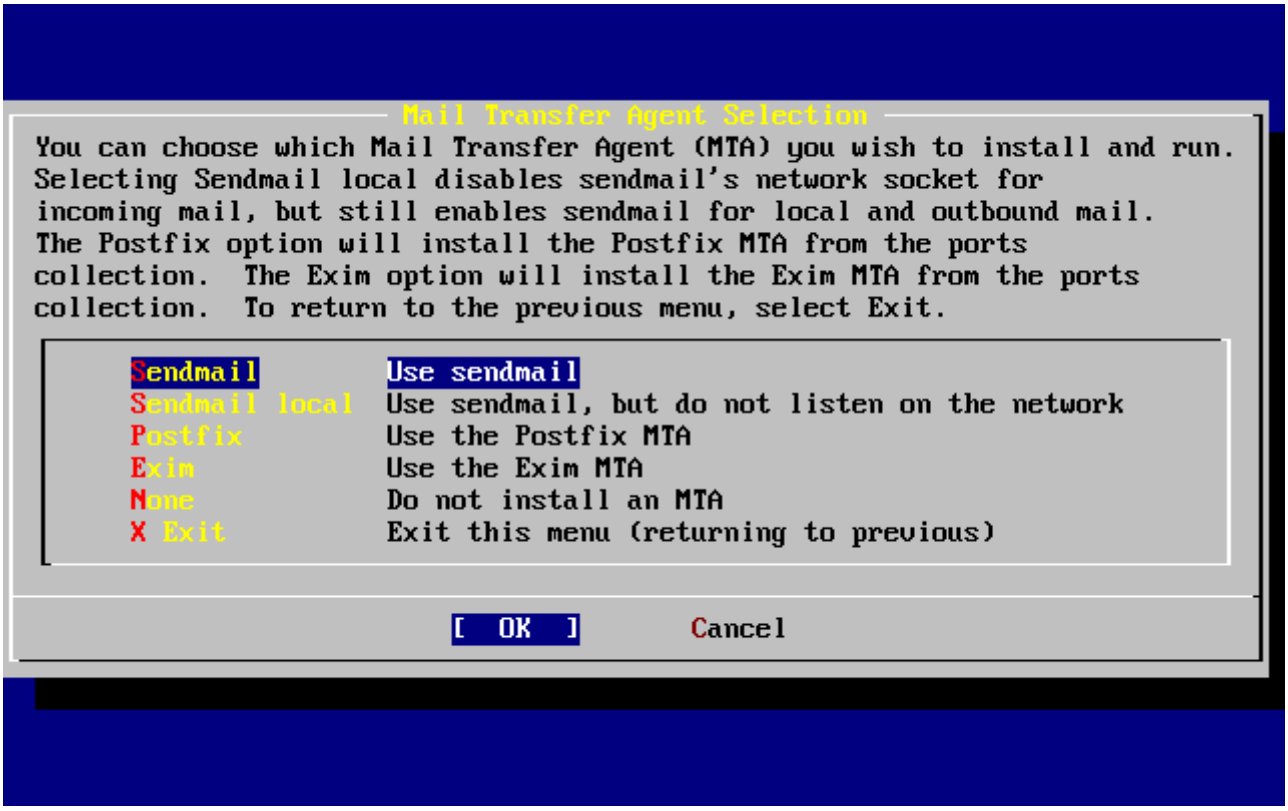
Az Anon FTP menüponton keresztül engedélyezhetjük az anonim FTP kapcsolatokat. A menüpont kiválasztásával számítógépünket egy anonim FTP szerverré tehetjük, azonban legyünk tekintettel a beállításhoz tartozó biztonsági veszélyekre! A kiválasztásakor egy ablak tájékoztat minket a beállítás részleteirõl és felmerülõ biztonsági kockázatokról.

A Gateway (Átjáró) menüpont használatával a korábbiakban tárgyaltak szerint állíthatjuk be számítógépünket hálózati átjárónak. Ugyanekkor a Gateway menüben nyílik lehetõségük

kikapcsolni ezt a beállítást, amennyiben a telepítési folyamat korábbi lépései során véletlenül engedélyeztük volna.

Az Inetd menüpont segítségével beállíthatjuk, vagy akár teljesen ki is kapcsolhatjuk a korábban tárgyalt inetd(8) démont.

A Mail (Levelezés) menüpontban beállíthatjuk a rendszer alapértelmezett MTA avagy levéltovábbító ügynökét (Mail Transfer Agent). Ennek hatására a következõ menü jelenik meg:



Ábra 56. Az alapértelmezett MTA kiválasztása

Itt választhatunk, hogy a különbözõ levélküldõ rendszerek közül melyiket telepítsük alapértelmezettként. Egy ilyen alkalmazás lényegében nem több, mint egy levélküldésre használt szerver, amely továbbítja a rendszerben vagy az interneten található felhasználók számára a leveleket.

A Sendmail választásával a FreeBSD alapból felkínált megoldását, a népszerû sendmail szervert telepíthetjük. A Sendmail local (Helyi Sendmail) menüpont kiválasztásával szintén a sendmail lesz a telepítendõ levélküldõ szerver, azonban nem lesz képes az internetrõl érkezõ leveleket fogadni. Az itt felsorolt többi beállítás, tehát a Postfix és Exim, a Sendmail beállításához hasonlóan zajlik. Mind a kettõ elektronikus levelek kézbesítésére használható, azonban bizonyos felhasználók a sendmail helyett inkább ezek valamelyikét használják.

Valamelyik vagy éppen semelyik levéltovábbító szerver kiválasztása után az NFS client (NFS kliens) beállítására vonatkozó menü jelentkezik.

Az NFS client beállításával a rendszerünk NFS szerverekkel lesz képes kapcsolatba lépni. Egy ilyen NFS szerver az NFS protokoll segítségével a hálózaton keresztül elérhetõvé tesz állományrendszereket. Ha gépünk független, akkor nem fontos kiválasztanunk ezt a menüpontot. A rendszernek késõbb további beállításokra is szüksége lehet, amelyekrõl az A hálózati

állományrendszer (NFS)ban olvashatunk részletesebben.

Az NFS server (NFS szerver) menüpont kiválasztásával hozzájárulunk, hogy rendszerünk NFS szerverként üzemeljen. Ehhez meg kell adnunk az RPC, vagyis a távoli eljáráshívások kiszolgálásának elindításához szükséges adatokat is. Az RPC használatával a különbözõ kiszolgálók és programok között tudjuk vezérelni a kapcsolatot.

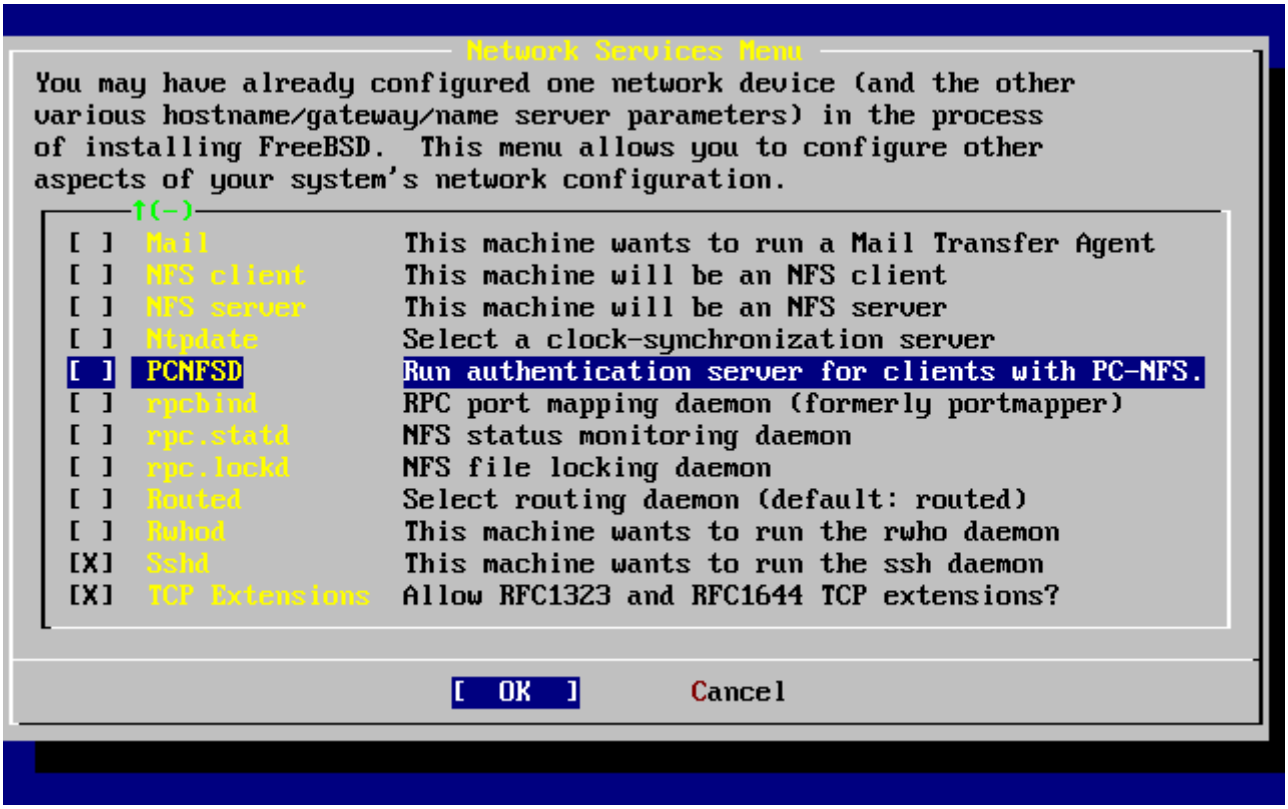
A sorban az Ntpdate beállítása következik, ahol az idõszinkronizációhoz kapcsolódó opciókat találjuk. Kiválasztásakor az ábrán szereplõhöz hasonló menü fog megjelenni:

Ábra 57. Az Ntpdate beállítása

Ebbõl a menübõl válasszuk ki a hozzánk legközelebb levõ szevert. Egy közeli szerver megadásával az idõszinkronizáció sokkalta pontosabbá válik, mivel a tõlünk távolabbi szerverek kapcsolatának késleltetése nagyobb lehet.

A következõ beállítás az PCNFSD. Ennek kiválasztása során a Portgyûjteménybõl telepítésre kerül a net/pcnfsd csomag. Ez lényegében egy hasznos segédprogram, amellyel olyan operációs rendszerek számára tudunk hitelesítést szolgáltatni az NFS használata során, amelyek maguktól erre nem képesek, mint például a Microsoft® MS-DOS® rendszere.

A többi beállítás megtekintéséhez egy kicsit lejjebb kell haladnunk a listában:



Ábra 58. A hálózati beállítások menüjének alsó szintje

Az rpcbind(8) és rpc.statd(8), valamint az rpc.lockd(8) segédprogramok mind a távoli eljáráshívásokhoz (Remote Procedure Call, RPC) használhatóak. Az `rpcbind` segédprogram az NFS szerverei és kliensei között felügyeli a kapcsolatot, ezért a használata az NFS szerverek és kliensek mûködéséhez elengedhetetlen. Az állapot figyeléséhez az rpc.statd démon felveszi a kapcsolatot a többi gépen futó rpc.statd démonokkal. A jelentett állapotok általában a /var/db/statd.status állományban találhatóak. Itt a következõként felsorolt elem az rpc.lockd, amelynek kiválasztásával állományzárolási szolgáltatásokat érhetünk el. Ezt többnyire az rpc.statd démonnal együtt alkalmazzák a zárolásokat kérõ gépek és a kérések gyakoriságának nyilvántartására. Míg ezekkel a beállításokkal gyönyörûen nyomon lehet követni a mûködést, az NFS szerverek és kliensek megfelelõ mûködéséhez nem kötelezõ a használatuk.

Ahogy haladunk tovább a listában, a következõ elem a Routed, vagyis az útválasztásért felelõs démon lesz. A routed(8) segédprogram a hálózati útválasztó táblázatokat tartja karban, felderíti az elérhetõ útválasztókat és kérésre bármelyik hozzá fizikailag csatlakozó gép számára átadja az általa nyilvántartott útválasztási adatokat. Ezt leginkább a helyi hálózat átjárójaként mûködõ számítógépek használják. Kiválasztásakor egy ablak fog rákérdezni a segédprogram helyére. Az itt alapból felkínált érték általában megfelelõ, ezért nyugtázhatjuk az `Enter` billentyû lenyomásával. Ezt követõen egy másik menü jelenik meg, ahol a routed beállításait adhatjuk meg. Itt alapértelmezés szerint a `-q` kapcsoló szerepel.

A következõ sor az Rwhod beállítását, aminek kiválasztásával el tudjuk indíttatni az rwhod(8) démont a rendszer elindítása során. Az `rwhod` segédprogram a rendszerüzeneteket a hálózaton idõközönként szétküldi vagy "figyelõ" (consumer) módban összegyûjti ezeket. Ennek pontosabb részleteit az ruptime(1) és rwho(1) man oldalakon találhatjuk meg.

Az sshd(8) démoné az utolsó elõtti beállítás. Ez az OpenSSH biztonságos shell szervere, melyet a szabványos telnet és FTP szerverek helyett ajánlanak. Az sshd szerver tehát két gép közti

biztonságos, titkosított kapcsolatok létrehozására használható.

A lista végén a TCP Extensions (TCP kiterjesztések) menüpontot találhatjuk. Segítségével a TCPRFC 1323 és RFC 1644 dokumentumokban leírt kiterjesztéseinek használatát engedélyezhetjük. Ezzel egyes gépek esetén felgyorsulhat a kapcsolat, azonban más esetekben pedig eldobódhat. Ez szerverek használatánál nem ajánlott, viszont független gépeknél kifizetõdõ lehet.

Most, miután beállítottuk a hálózati szolgáltatásokat, lépjünk vissza a lista elején található X Exit (Kilépés) menüpontra és folytassuk a beállítást a következõ opcióval, vagy egyszerûen az X Exit kétszeri kiválasztásával, majd a **[X Exit Install]** (Kilépés a telepítõbõl) gomb lenyomásával lépjünk ki a sysinstall programból.

2.10.16. A FreeBSD indulása

A FreeBSD/i386 indulása

Ha minden remekült ment, a képernyõn lentrõl felfelé gördülõ üzeneteket fogunk látni, majd a rendszer várni fog tõlünk egy bejelentkezési nevet. A kiírt üzeneteket között a `Scroll Lock` lenyomása után a `PgUp` és `PgDn` billentyûk használatával tudunk lapozni. A `Scroll Lock` ismételt lenyomásával visszatérünk a bejelentkezéshez.

Nem minden esetben lesz látható az összes üzenet (a puffer végessége miatt), de miután bejelentkeztünk, ezeket a `dmesg` parancs kiadásával is megnézhetjük.

Bejelentkezni a telepítéskor megadott felhasználói név/jelszó párossal tudunk (a példában ez most `rpratt`). Lehetõleg ne jelentkezzünk be `root` felhasználóként!

A rendszer indításakor jellemzõen elõforduló üzenetek (a verzióra vonatkozó adatokat kihagytuk):

```
Copyright (c) 1992-2002 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
        The Regents of the University of California. All rights reserved.

Timecounter "i8254"  frequency 1193182 Hz
CPU: AMD-K6(tm) 3D processor (300.68-MHz 586-class CPU)
  Origin = "AuthenticAMD"  Id = 0x580  Stepping = 0
  Features=0x8001bf<FPU,VME,DE,PSE,TSC,MSR,MCE,CX8,MMX>
  AMD Features=0x80000800<SYSCALL,3DNow!>
real memory  = 268435456 (262144K bytes)
config> di sn0
config> di lnc0
config> di le0
config> di ie0
config> di fe0
config> di cs0
config> di bt0
config> di aic0
config> di aha0
config> di adv0
config> q
```

```
avail memory = 256311296 (250304K bytes)
Preloaded elf kernel "kernel" at 0xc0491000.
Preloaded userconfig_script "/boot/kernel.conf" at 0xc049109c.
md0: Malloc disk
Using $PIR table, 4 entries at 0xc00fde60
npx0: <math processor> on motherboard
npx0: INT 16 interface
pcib0: <Host to PCI bridge> on motherboard
pci0: <PCI bus> on pcib0
pcib1: <VIA 82C598MVP (Apollo MVP3) PCI-PCI (AGP) bridge> at device 1.0 on pci0
pci1: <PCI bus> on pcib1
pci1: <Matrox MGA G200 AGP graphics accelerator> at 0.0 irq 11
isab0: <VIA 82C586 PCI-ISA bridge> at device 7.0 on pci0
isa0: <ISA bus> on isab0
atapci0: <VIA 82C586 ATA33 controller> port 0xe000-0xe00f at device 7.1 on pci0
ata0: at 0x1f0 irq 14 on atapci0
ata1: at 0x170 irq 15 on atapci0
uhci0: <VIA 83C572 USB controller> port 0xe400-0xe41f irq 10 at device 7.2 on pci0
usb0: <VIA 83C572 USB controller> on uhci0
usb0: USB revision 1.0
uhub0: VIA UHCI root hub, class 9/0, rev 1.00/1.00, addr 1
uhub0: 2 ports with 2 removable, self powered
chip1: <VIA 82C586B ACPI interface> at device 7.3 on pci0
ed0: <NE2000 PCI Ethernet (RealTek 8029)> port 0xe800-0xe81f irq 9 at
device 10.0 on pci0
ed0: address 52:54:05:de:73:1b, type NE2000 (16 bit)
isa0: too many dependant configs (8)
isa0: unexpected small tag 14
fdc0: <NEC 72065B or clone> at port 0x3f0-0x3f5,0x3f7 irq 6 drq 2 on isa0
fdc0: FIFO enabled, 8 bytes threshold
fd0: <1440-KB 3.5" drive> on fdc0 drive 0
atkbdc0: <keyboard controller (i8042)> at port 0x60-0x64 on isa0
atkbd0: <AT Keyboard> flags 0x1 irq 1 on atkbdc0
kbd0 at atkbd0
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: model Generic PS/2 mouse, device ID 0
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on isa0
sc0: <System console> at flags 0x1 on isa0
sc0: VGA <16 virtual consoles, flags=0x300>
sio0 at port 0x3f8-0x3ff irq 4 flags 0x10 on isa0
sio0: type 16550A
sio1 at port 0x2f8-0x2ff irq 3 on isa0
sio1: type 16550A
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
ppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode
ppc0: FIFO with 16/16/15 bytes threshold
ppbus0: IEEE1284 device found /NIBBLE
Probing for PnP devices on ppbus0:
plip0: <PLIP network interface> on ppbus0
lpt0: <Printer> on ppbus0
lpt0: Interrupt-driven port
```

```
ppi0: <Parallel I/O> on ppbus0
ad0: 8063MB <IBM-DHEA-38451> [16383/16/63] at ata0-master using UDMA33
ad2: 8063MB <IBM-DHEA-38451> [16383/16/63] at ata1-master using UDMA33
acd0: CDROM <DELTA OTC-H101/ST3 F/W by OIPD> at ata0-slave using PIO4
Mounting root from ufs:/dev/ad0s1a
swapon: adding /dev/ad0s1b as swap device
Automatic boot in progress...
/dev/ad0s1a: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0s1a: clean, 48752 free (552 frags, 6025 blocks, 0.9% fragmentation)
/dev/ad0s1f: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0s1f: clean, 128997 free (21 frags, 16122 blocks, 0.0% fragmentation)
/dev/ad0s1g: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0s1g: clean, 3036299 free (43175 frags, 374073 blocks, 1.3% fragmentation)
/dev/ad0s1e: filesystem CLEAN; SKIPPING CHECKS
/dev/ad0s1e: clean, 128193 free (17 frags, 16022 blocks, 0.0% fragmentation)
Doing initial network setup: hostname.
ed0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet 192.168.0.1 netmask 0xffffff00 broadcast 192.168.0.255
        inet6 fe80::5054::5ff::fede:731b%ed0 prefixlen 64 tentative scopeid 0x1
        ether 52:54:05:de:73:1b
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x8
        inet6 ::1 prefixlen 128
        inet 127.0.0.1 netmask 0xff000000
Additional routing options: IP gateway=YES TCP keepalive=YES
routing daemons:.
additional daemons: syslogd.
Doing additional network setup:.
Starting final network daemons: creating ssh RSA host key
Generating public/private rsa1 key pair.
Your identification has been saved in /etc/ssh/ssh_host_key.
Your public key has been saved in /etc/ssh/ssh_host_key.pub.
The key fingerprint is:
cd:76:89:16:69:0e:d0:6e:f8:66:d0:07:26:3c:7e:2d root@k6-2.example.com
 creating ssh DSA host key
Generating public/private dsa key pair.
Your identification has been saved in /etc/ssh/ssh_host_dsa_key.
Your public key has been saved in /etc/ssh/ssh_host_dsa_key.pub.
The key fingerprint is:
f9:a1:a9:47:c4:ad:f9:8d:52:b8:b8:ff:8c:ad:2d:e6 root@k6-2.example.com.
setting ELF ldconfig path: /usr/lib /usr/lib/compat /usr/X11R6/lib
/usr/local/lib
a.out ldconfig path: /usr/lib/aout /usr/lib/compat/aout /usr/X11R6/lib/aout
starting standard daemons: inetd cron sshd usbd sendmail.
Initial rc.i386 initialization:.
rc.i386 configuring syscons: blank_time screensaver moused.
Additional ABI support: linux.
Local package initialization:.
Additional TCP options:.

FreeBSD/i386 (k6-2.example.com) (ttyv0)
```

```
login: rpratt
Password:
```

Az RSA és DSA kulcsok generálása a lassabb gépeken soká is eltarthat, habár ez mindig csak a friss telepítések utáni elsõ indításkor történik meg. A rendszer késõbbi indulásai ettõl már gyorsabbak lesznek.

Ha X szervert is beállítottunk és választottunk hozzá egy alapértelmezett munkakörnyezetet, akkor ezt a parancssorból a `startx` kiadásával elindíthatjuk el.

2.10.17. A FreeBSD leállítása

Fontos, hogy mindig szabályosan állítsuk le az operációs rendszert, ne kapcsoljuk ki csak úgy egyszerûen a számítógépünket! A leállításhoz elõször a `su` parancs kiadásával, majd itt a `root` jelszavának megadásával vegyük fel az ehhez szükséges rendszeradminisztrátori jogosultságokat. Ez viszont csak abban az esetben fog mûködni, ha a felhasználónk tagja a `wheel` csoportnak. Minden más esetben egyszerûen jelentkezzünk be `root` felhasználóként és használjuk a `shutdown -h now` parancsot.

```
The operating system has halted.
Please press any key to reboot.
```

A fenti üzenet jelzi, hogy a leállító parancs kiadása után már kikapcsolhatjuk a számítógépet, vagy ha ehelyett egy billentyût nyomunk le, akkor a gép újraindul.

A `Ctrl` + `Alt` + `Del` billentyûkombináció használatával is újra tudjuk indítani a rendszert, azonban ez normál mûködés közben nem ajánlott.

2.11. Hibakeresés

A most következõ szakaszban azokra a telepítés során felmerülõ problémákra próbálunk meg megoldásokat adni, amelyeket eddig már sokan jeleztek nekünk. Ezek mellett szerepel néhány kérdés és válasz is a FreeBSD és az MS-DOS® vagy Windows® közös használatáról.

2.11.1. Mit tegyünk ha valami nem mûködik

A PC architektúra különféle korlátozásai miatt szinte lehetetlen 100%-ban megbízhatóvá tenni az eszközök felderítését, azonban ennek hibája kapcsán néhány dolgot még tenni tudunk.

Ellenõrizzük a Hardware Notes (Hardverjegyzék) címû dokumentumban, hogy az adott hardvert a FreeBSD valóban ismeri.

Amennyiben a hardvereszközünket a rendszer ismeri, azonban még mindig jelentkeznek fagyások vagy egyéb gondok, készítenünk kell egy crosssref:kernelconfig[kernelconfig,saját rendszermag]ot. Ezzel olyan eszközök támogatását is beépíthetjük a rendszermagba, amelyek eredetileg nem szerepelnek a GENERIC rendszermagban. A telepítéshez készített rendszerindító lemezeken

található rendszermag a legtöbb eszközt a gyári IRQ, IO-cím és DMA csatorna beállításaik mentén próbálja felkutatni. Ha viszont a hardverünket átállítottuk, ennek megfelelõen módosítanunk kell a rendszermag beállításait és újra kell fordítanunk, hogy a FreeBSD tudja, hol is keresse az eszközt.

Olyan is adódhat, hogy egy nem létezõ eszköz keresése egy utána keresendõ másik, jelenlevõ eszköz felkutatását akadályozza meg. Ilyenkor az ütközõ meghajtókat le kell tiltani.

Egyes problémák elkerülhetõek vagy csillapíthatóak a különbözõ hardverösszetevõk, különösen az alaplapi firmware frissítésével. Az alaplap firmware-jére sokszor csak BIOS-ként hivatkoznak, és a legtöbb alaplap- vagy számítógépgyártó honlapján találhatjuk meg ezeket, valamint a rájuk vonatkozó utasításokat.

A legtöbb gyártó azonban erõsen tiltakozik az alaplapi BIOS-frissítések ellen, és csak indokolt esetekben, például kritikus javításoknál javasolják. A frissítés kimenetele *lehet* rossz is, aminek következménye a BIOS tartós károsodása.

2.11.2. Az MS-DOS® és Windows® állományrendszereinek használata

A FreeBSD jelenleg nem támogatja a Double Space™ alkalmazással tömörített állományrendszereket, ezért a FreeBSD csak úgy tud az adataihoz hozzáférni, ha elõtte kitömörítjük ezeket. Ezt a Start menü Programs (Programok) > System Tools (Rendszereszközök) menüjében található Compression Agent (Lemeztömörítés) elindításával tehetjük meg.

A FreeBSD támogatja az MS-DOS® alapú (gyakran csak FAT típusúnak nevezett) állományrendszereket. A mount_msdosfs(8) parancs segítségével az ilyen rendszerek könnyedén becsatlakoztathatók a már létezõ könyvtárszerkezetbe, amivel így el tudjuk érni a tartalmát. A mount_msdosfs(8) programot általában nem közvetlenül hívjuk meg, hanem az /etc/fstab vagy a mount(8) segédprogram megfelelõ paraméterezésével.

Az /etc/fstab állományban általában így néz ki egy ilyen sor:

```
/dev/ad0sN  /dos    msdosfs rw 0 0
```

A mûvelet végrehajtásához a /dos könyvtárnak már léteznie kell. Az /etc/fstab pontos formátumával kapcsolatban a fstab(5) man oldalt olvassuk el.

Az MS-DOS® állományrendszerek esetében a mount(8) parancsot többnyire így adjuk ki:

```
# mount -t msdosfs /dev/ad0s1 /mnt
```

Ebben a példában a MS-DOS® állományrendszer az elsõdleges merevlemez elsõ partícióján helyezkedik el. A mi helyzetünk ettõl eltérõ lehet, ezért ehhez vizsgáljuk meg a `dmesg` és `mount` parancsok kimeneteit. Segítségükkel elegendõ információt tudunk összeszedni a gépünkön található partíciók kiosztásáról.

Elõfordulhat, hogy a FreeBSD a többi operációs rendszertõl eltérõ módon számozza a slice-okat (vagyis az MS-DOS® partíciókat). Konkrétan: a kiterjesztett MS-DOS® partíciók általában nagyobb sorszámot kapnak, mint az elsõdleges MS-DOS® partíciók. Az fdisk(8) segédprogram segíthet megállapítani, hogy mely slice-ok tartoznak a FreeBSD-hez és melyek más operációs rendszerekhez.

A mount_ntfs(8) parancs használatával az NTFS partíciók hasonló módon csatlakoztathatóak.

2.11.3. Kérdések és válaszok

A rendszerem teljesen leáll amikor az indítás során eszközöket próbál megtalálni, vagy furcsán viselkedik a telepítés során, esetleg a floppy meghajtót nem is keresi.

A FreeBSD az i386, amd64 és ia64 platformokon az indítás közben az eszközök felderítésében erõsen építkeznek a rendszeren elérhetõ ACPI szolgáltatásra. Sajnos még mindig vannak hibák az ACPI meghajtóban, az alaplapokban és a BIOS-okban. A rendszerbetöltõ harmadik fokozatában viszont az `hint.acpi.0.disabled` megadásával kikapcsolható az ACPI használata:

```
set hint.acpi.0.disabled="1"
```

Ez a beállítás a rendszer minden egyes indításakor törlõdik, ezért a `hint.acpi.0.disabled="1"` bejegyzést fel kell vennünk a /boot/loader.conf állományba. A rendszerbetöltõ mûködésérõl részletesebben a Áttekintésban olvashatunk.

A FreeBSD telepítése után elõször indítom el a merevlemezrõl a rendszert, a rendszermag betöltõdik és nekilát felkutatni a hardvereszközöket, azonban megáll a következõ üzenettel:

Ez egy régóta fennálló probléma olyan rendszerek esetén, ahol a rendszerindításhoz használt lemez nem az elsõ. A BIOS a FreeBSD-tõl eltérõ sorszámozást használ, és az általa alkalmazott megfeleltetések megfejtése nehézkes.

Amikor a rendszer indítására használt lemez nem az elsõ lemez a rendszerünkben, segítenünk kell a FreeBSD-nek a megtalálásában. Két gyakori helyzet alakulhat ki, és mind a kettõben el kell árulnunk a FreeBSD-nek, hogy hol található a rendszer indításához használható gyökér állományrendszer. Ezt a lemez BIOS-ban nyilvántartott sorszámának, típusának és a neki megfelelõ FreeBSD szerinti lemezszám megadásával tehetjük meg.

Az elsõ szituációban két IDE-lemezünk van, mind a kettõt masterként állítottuk be a hozzájuk tartozó IDE-buszokon, és a közülük a másodikról akarjuk indítani a FreeBSD-t. A BIOS ezeket 0. és 1. lemezként látja, miközben a FreeBSD pedig ad0 és ad2 eszközként.

A FreeBSD 1. BIOS-számozású lemezen van, amelynek a típusa `ad` és a FreeBSD szerinti a 2 sorszámot viseli. Ezért ezt kell használnunk:

```
1:ad(2,a)kernel
```

Ha az elsõdleges buszon van egy slave meghajtónk, akkor mindez nem szükséges (és valószínûleg

rossz is).

A második szituációban egy SCSI-lemezrõl akarjuk indítani a rendszert, miközben egy vagy több IDE-lemez is található a gépünkben. Ebben az esetben a FreeBSD szerinti sorszám kisebb lesz, mint a BIOS szerinti. Ha tehát a két IDE-lemezünk mellett van még egy SCSI-lemez is, akkor annak a BIOS szerinti sorszáma 2, a típusa `da` és a FreeBSD szerinti sorszáma pedig 0. Ennek megfelelõen a

```
2:da(0,a)kernel
```

sorral tudjuk elárulni a FreeBSD-nek, hogy a BIOS szerint 2. lemezrõl akarjuk indítani, amely a rendszerben található elsõ SCSI-lemeznek felel meg. Ha csak egy IDE-lemezünk van, akkor a sort kezdjük az `1:` beírásával.

Miután megtaláltuk a megfelelõ értékeket, a hozzá tartozó sort egy szövegszerkesztõ segítségével tegyük közvetlenül a /boot.config állományba. A FreeBSD ezen állomány tartalmát fogja alapból felhasználni a `boot:` bekérésénél, hacsak másképpen nem utasítjuk.

A telepítés után elõször próbálom meg elindítani a merevlemezrõl a FreeBSD-t, azonban a rendszerválasztó mindig csak F? opciókat kínál fel, és a rendszer indítása sem halad tovább.

A FreeBSD telepítése során rosszul adtunk meg a partíciószerkesztõben a merevlemezhez tartozó geometriát. Menjünk vissza a partíciószerkesztõhöz és adjuk meg újra a merevlemezünk helyes geometriáját. Ennek használatához pedig a FreeBSD-t is újra kell telepítenünk.

Ha egyáltalán képtelenek vagyunk megállapítani a merevlemezhez tartozó geometriát, akkor próbáljuk meg ezt: a lemez elején hozzunk létre egy kis méretû DOS partíciót és rakjuk utána a FreeBSD-t. Amikor a telepítõprogram észreveszi a DOS partíciót, megpróbálja magától kikövetkeztetni belõle a helyes geometriát, ami általában mûködik is.

Ez a tanács ugyan már nem érvényes, de álljon itt felvilágosításként:

Ha teljesen egy FreeBSD alapú szerver vagy munkaállomás kialakítására szánjuk a számítógépünket, és nem törõdünk a DOS-szal, Linuxszal és a többi operációs rendszerrel történõ (jövõbeli) kompatibilitással, használhatjuk akár az egész lemezt is (a partíciószerkesztõben ez az A opció). Ezzel egy olyan nem szabványos beállítást engedélyezünk, amivel a FreeBSD elfoglalja a lemezt annak legelsõ szektorától a legutolsó szektoráig. Ilyenkor ugyan el tudunk tekinteni a geometriával kapcsolatos beállításoktól, azonban így a FreeBSD-n kívül semmilyen más operációs rendszert nem tudunk majd futtatni a gépen.

A rendszer megtalálja a ed(4) hálózati kártyámat, azonban folyamatosan hibát ad idõtúllépésre hivatkozva.

Az említett kártya valószínûleg a /boot/device.hints állományban beállítottaktól eltérõ IRQ-t használ. A ed(4) meghajtó alapértelmezés szerint nem használ "szoftveres" beállításokat (amiket DOS-ban az EZSETUP használatával adunk meg), viszont engedélyezhetjük, ha a kártyánál megadjuk az `-l` beállítást.

Hardveresen ezt a kártyán levõ jumperek segítségével állíthatjuk be (ehhez változtassuk meg a rendszermag beállításait is, amennyiben szükséges), vagy a `-l` kapcsolón keresztül a

`hint.ed.0.irq="-l"` megadásával utasíthatjuk a rendszermagot az IRQ szoftveres beállítására.

Másik lehetõség, amikor a kártyánk a 9-es IRQ-t használja, amelyet általában megosztanak a 2-es IRQ-val, ami gyakori problémák forrása (különösen abban az esetben, amikor a VGA kártya a 2-es IRQ-t használja!) lehet. Lehetõleg ne használjuk a 2-es és 9-es IRQ-kat.

Amikor a sysinstall programot egy X11 terminálban futtatom, a sárga színû betûket viszonylag nehéz olvasni a világosszürke háttérrel. Esetleg lehet valahogy növelni a kontrasztotszínek kontraszt az alkalmazás használatakor?

Ha az X11 telepítése után a sysinstall által választott színekkel nem olvasható a szöveg xterm(1) vagy rxvt(1) terminálokban, akkor vegyük fel a következõ sort a felhasználói könyvtárunkban levõ .Xdefaults konfigurációs állományunkba: `XTerm*color7:#c0c0c0`. Ezzel majd egy sötétebb szürke hátteret kapunk.

2.12. Telepítési útmutató haladóknak

Ebben a szakaszban megtudhatjuk, hogyan telepítsük a FreeBSD-t speciális esetekben.

2.12.1. A FreeBSD telepítése billentyûzet vagy monitor nélkül

A telepítés ezen fajtáját "fej nélküli telepítésnek" (headless install) hívják, mivel a gép, amire a FreeBSD-t telepíteni akarjuk, nem rendelkezik monitorral vagy éppen még VGA kimenettel sem. Felmerülhet a kérdés: hogyan lehetséges mindez? A soros vonali konzol használatával! A soros konzol segítségével lényegében egy másik számítógép monitorját és billentyûzetét használjuk. Ennek megvalósításához elsõként kövessük a rendszerindító pendrive készítésének Készítsünk egy rendszerindító lemeztban leírt lépéseit, vagy töltsük le a megfelelõ ISO image-et a telepítéshez, lásd Telepítõ CD készítése.

A következõ lépésekkel tehetjük képessé a soros konzolon keresztüli rendszerindításra: (CD-lemez használata esetén az elsõ lépésre nincs szükség)

1. A rendszerindító pendrive átállítása soros konzolra

   Ha a korábban elõkészített pendrive-val most csak egyszerûen elindítanánk a FreeBSD-t, akkor a megszokott telepítési módban indulna el. Mi viszont azt akarjuk, hogy a telepítéshez a FreeBSD a soros konzolon keresztül induljon el. Ehhez csatlakoztassuk az eszközt a számítógéphez, valamint a mount(8) paranccsal FreeBSD rendszerünkhöz pedig a hozzátartozó állományrendszert.

   ```
   # mount /dev/da0a /mnt
   ```

   A konkrét eszköznevet és csatlakozási pontot módosítsuk a saját környezetünknek megfelelõen.

   Most, miután már fizikailag és logikailag is csatlakoztattuk a pendrive-ot, be kell állítanunk a soros konzol használatára rendszerindítás közben. Ehhez egy loader.conf nevû állományt

kell elhelyeznünk a pendrive állományrendszerén a soros konzolra (mint rendszerkonzolra) vonatkozó beállítással:

```
# echo 'console="comconsole"' >> /mnt//boot/loader.conf
```

Miután a pendrive-on sikeresen elvégeztük a szükséges beállítást, válasszuk le a umount(8) parancs kiadásával:

```
# umount /mnt
```

Most már leválaszthatjuk a pendrive-ot, és ugorjunk közvetlenül a harmadik lépésre.

2. A null-modem kábel csatlakoztatása

   Össze kell kötnünk a két számítógépet egy null-modem kábellel. Nincs más teendõnk, mit összekapcsolni a két gép soros portjait. *Itt a szokásos soros kábel nem mûködik*, konkrétan null-modem kábelre van szükség, mivel benne néhány vezetéket máshogy kötöttek be.

3. A telepítõ CD beállítása soros konzolra

   Ha a telepítésre szánt ISO image-bõl készített lemezzel (lásd Telepítõ CD készítése) a FreeBSD normál módban indul el. A soros konzol használatához viszont kibontani, módosítani és újragenerálni kell az adott image-et mielõtt lemezre írnánk.

   A korábban, például a FreeBSD-8.1-RELEASE-i386-disc1.iso néven letöltött image-bõl a tar(1) segédprogrammal tudjuk kinyerni a benne tárolt összes állományt:

   ```
   # mkdir /a/hasznalt/iso/helye
   # tar -C /a/hasznalt/iso/helye -pxvf FreeBSD-8.1-RELEASE-i386-disc1.iso
   ```

   Ezt követõen módosítanunk kell a telepítõlemezt a soros konzol használatára. Ehhez egy loader.conf állományt kell hozzáadnunk a kibontott ISO image tartalmához. Ebben állítjuk be a soros konzolt rendszerkonzolnak:

   ```
   # echo 'console="comconsole"' >> /a/hasznalt/iso/helye/boot/loader.conf
   ```

   Ezután készítsünk egy új ISO image-et a módosított tartalom alapján. Ehhez a sysutils/cdrtools port részeként elérhetõ mkisofs(8) segédprogramot használjuk:

   ```
   # mkisofs -v -b boot/cdboot -no-emul-boot -r -J -V "soroskonzolos" -o
   soroskonzolos-FreeBSD-8.1-RELEASE-i386-disc1.iso /a/hasznalt/iso/helye
   ```

   Most már van egy megfelelõen összeállított ISO image-ünk, amelyet CD-lemezre tudunk írni a kedvenc CD-író alkalmazásunkkal.

4. A telepítés indítása

   Most már ideje elkezdeni a telepítést. Tegyük a boot.flp image-et tartalmazó lemezt a fej nélkül telepítendõ gép meghajtójába és kapcsoljuk be.

5. Kapcsolódás a fej nélküli gépre

   Ezután a cu(1) parancs felhasználásával kapcsolódjunk rá a gépre:

   ```
   # cu -l /dev/cuau0
   ```

   Ezt FreeBSD 7.*X* esetén így kell használnunk:

   ```
   # cu -l /dev/cuad0
   ```

Ezzel készen is vagyunk! Innentõl a `cu` által megnyitott kapcsolaton keresztül tudjuk vezérelni a fej nélküli számítógépet. Hamarosan betölti a rendszermagot, majd megkérdezi a használt terminál típusát. Itt válasszuk ki a színes FreeBSD konzolt (FreeBSD color console) és folytassuk a telepítést a megszokott módon.

2.13. Saját telepítõeszköz elkészítése

Az ismétlések elkerülése végett a továbbiakban a "FreeBSD lemez" a megvásárolható vagy a magunk által készített FreeBSD CD-re vagy DVD-re vonatkozik.

Adódhatnak olyan esetek, amikor létre kell hoznunk a FreeBSD telepítésére használt saját eszközünket és/vagy forrásunkat. Ez lehet egy tetszõleges fizikai eszköz, például szalag, vagy bármilyen olyan forrás, ahonnan a sysinstall képes állományokat elérni, például egy FTP oldal vagy egy MS-DOS® partíció.

Például:

- Egy FreeBSD lemezünk van és több hálózaton kapcsolódó számítógépünk. Készíteni akarunk egy helyi FTP oldalt a FreeBSD lemez felhasználásával, és így a hálózaton levõ gépre az internet helyett innen telepítjük a rendszert.
- Van egy FreeBSD lemezünk, azonban a FreeBSD-nek nem sikerült felismernie a CD/DVD-meghajtónkat, viszont az MS-DOS®/Windows®-nak igen. Felmásoljuk a FreeBSD telepítéséhez használt állományokat ugyanazon a számítógépen található egyik DOS partícióra, majd a FreeBSD-t ezekkel telepítjük.
- A gépben, amelyre telepíteni akarunk, nincs CD/DVD-meghajtó vagy hálózati kártya, viszont "Laplink stílusú" soros vagy párhuzamos kábellel hozzá tudunk kapcsolódni egy olyan számítógéprõl, amelyben viszont van.
- Készíteni akarunk a FreeBSD telepítésére használható szalagot.

2.13.1. Telepítõ CD készítése

A FreeBSD Projekt minden kiadás részeként architektúránként elérhetõvé tesz legalább két CD image-et ("ISO image-et"). Ha rendelkezünk CD-íróval, ezeket az image-eket fel-, illetve ki tudjuk írni ("égetni") CD-re, és a FreeBSD telepítésére tudjuk használni. Tehát ha van a kezünk ügyében CD-író és olcsón jutunk nagyobb sebességû interneteléréshez, akkor a FreeBSD telepítésének ez a legkönnyebb módja.

1. A megfelelõ ISO image-ek letöltése

   Az egyes kiadások ISO image-ei letölthetõek a ftp://ftp.FreeBSD.org/pub/FreeBSD/ISO-IMAGES-architektúra/változat címrõl vagy annak legközelebbi tükrözésérõl. Az *architektúra* és *változat* részeket igényeinknek megfelelõen helyettesítsük.

   Az említett könyvtár általában a következõ lemezek image-eit tartalmazza:

   Táblázat 4. FreeBSD 7.X és 8.X ISO image-ek nevei és jelentései

Állománynév	Tartalom
FreeBSD-változat-RELEASE-architektúra-bootonly.iso	Ezzel a CD image-dzsel tudjuk a FreeBSD CD-meghajtóról indításával elkezdeni a telepítést. Fontos tudnunk azonban, hogy ez az image nem tartalmazza a FreeBSD telepítéséhez szükséges komponenseket. Ezt a rendszer indítása után hálózaton keresztül (például egy FTP szerver segítségével) tudjuk megtenni.
FreeBSD-változat-RELEASE-architektúra-dvd1.iso.gz	Ez a DVD image minden, az alap FreeBSD rendszer telepítéséhez szükséges komponenst tartalmaz, bináris csomagokkal és dokumentációval együtt. Ezenkívül még "élõ" rendszert is tudunk indítani vele, közvetlenül a lemezrõl.
FreeBSD-változat-RELEASE-architektúra-memstick.img	Ez az image egy USB pendrive-ra írható, és minden olyan számítógépen használható, amely képes ilyen eszközrõl elindulni. Támogatja az "élõ" módot is, amellyel rendszerünket állíthatjuk helyre. Ez az image nem érhetõ el FreeBSD 7.3 vagy korábbi rendszerek esetén.
FreeBSD-változat-RELEASE-architektúra-disc1.iso	Ez az image tartalmazza az alap FreeBSD operációs rendszert és a hozzá tartozó dokumentációt, de semmilyen más további csomagot nem.

Állománynév	Tartalom
FreeBSD-változat-RELEASE-architektúra-disc2.iso	Ezen az image-en bináris csomagok találhatóak. Ilyen a FreeBSD 8.0 és az utána következõ változatoknál már nincs.
FreeBSD-változat-RELEASE-architektúra-disc3.iso	Ez egy másik image, amelyen szintén bináris csomagok találhatóak. Ilyen a FreeBSD 8.0 és az utána következõ változatoknál már nincs.
FreeBSD-változat-RELEASE-architektúra-docs.iso	A FreeBSD dokumentációja.
FreeBSD-változat-RELEASE-architektúra-livefs.iso	Ez az image a rendszerhelyreállításhoz használt "élõ" indítási módot támogatja, telepítést alapvetõen nem lehet vele végezni.

A FreeBSD 7.3 és a FreeBSD 8.1 elõtti 7.*X*, illetve 8.*X* kiadások egy ettõl eltérõ elnevezési sémát követnek: a hozzájuk tartozó ISO image-ek neveiben nem szerepel a `FreeBSD-` elõtag.

Le *kell* töltenünk az elsõ lemez vagy (ha elérhetõ) a `bootonly` lemez ISO image-einek egyikét. A kettõt egyszerre viszont ne töltsük le, mivel a `disc1` image tartalmaz mindent, ami a `bootonly` image-en megtalálható.

Akkor használjuk a `bootonly` jelzésû image-et, ha szélessávú internetelérésssel rendelkezünk. Segítségével el tudjuk kezdeni a FreeBSD telepítését, és szükség szerint a port/csomagrendszer (lásd Alkalmazások telepítése. csomagok és portok) használatával csomagokat tudunk letölteni és telepíteni.

A DVD image-ét (`dvd1`) akkor érdemes használni, ha a FreeBSD adott kiadásának telepítése mellett igényt tartunk valamennyi csomagra is.

A további lemezek image-ei is hasznosak lehetnek, de nem feltétlenül kellenek a telepítéshez, fõleg abban az esetben, amikor gyors internetelérésssel rendelkezünk.

2. A CD-k írása

Ezután lemezekre kell írnunk a letöltött image-eket. Amennyiben ezt egy másik FreeBSD rendszeren végezzük, ennek részleteirõl a Lézeres tárolóeszközök (CD-k) létrehozása és használata számol be (különösen a burncd és a cdrecord leírása).

Ha másik platformon végezzük ezt a mûveletet, akkor az adott platformon felkínált CD-író szoftverekkel kell dolgoznunk. Az image-ek szabványos ISO formátumúak, amelyet szinte az összes CD-író alkalmazás ismer.

Ha kíváncsiak vagyunk egy saját FreeBSD kiadás elkészítésére, olvassuk el a kiadások szervezésérõl szóló cikket (angolul).

2.13.2. Helyi FTP oldal létrehozása FreeBSD lemezzel

A FreeBSD lemezeken az FTP oldalakéhoz hasonló elrendezést találunk. Ez megkönnyíti a hálózatunkban található számítógépekhez a FreeBSD telepítésére használható helyi FTP oldal létrehozását.

1. Az FTP oldalnak otthont adó FreeBSD számítógépen tegyük a CD-t a meghajtóba, majd csatlakoztassuk a /cdrom könyvtárba.

   ```
   # mount /cdrom
   ```

2. Hozzunk létre egy anonim FTP hozzáférést az /etc/passwd állományban. A vipw(8) segítségével tehát illesszük be a következõ sort az /etc/passwd állományba:

   ```
   ftp:*:99:99::0:0:FTP:/cdrom:/nonexistent
   ```

3. Gondoskodjuk róla, hogy az FTP szolgáltatás engedélyezve legyen az /etc/inetd.conf állományban.

Most már bárki, aki képes csatlakozni ehhez a számítógéphez, a telepítés típusának ki tudja választani az FTP-t. Az FTP oldalak menüjében válassza az "Other" (Egyéb) pontot, majd adja meg az `ftp://gépnév` címet.

Ha az FTP-n csatlakozó kliensek rendszerindításhoz használt eszköze (általában a floppy) verziója nem egyezik meg tökéletesen a helyi FTP oldalon találhatóval, akkor a sysinstall nem engedi a telepítést. Ha a változatok nem hasonlóak és ezt felül akarjuk bírálni, akkor be kell lépnünk az **Options** (Beállítások) menübe, ahol át kell állítanunk a terjesztés nevét (distribution name) any (bármelyik)-re.

A fenti megközelítés kizárólag csak egy tûzfallal védett helyi hálózaton javasolt. FTP szolgáltatás létrehozása az interneten (és nem a helyi hálózatunkban) levõ számítógépek számára különbözõ támadásoknak és egyéb kellemetlenségeknek teszi ki a számítógépünket. Határozottan javasoljuk, hogy ebben az esetben különösen ügyeljünk a biztonságra.

2.13.3. Telepítõfloppyk létrehozása

Ha floppylemezrõl kellene telepítenünk (amit viszont *semmiképpen sem* ajánlanánk) egy nem támogatott hardvereszköz miatt, vagy mert egyszerûen szeretjük a dolgok nehezebbik oldalát megfogni, akkor ehhez elõször elõ kell készítenünk pár lemezt.

Legalább annyi 1,44 MB-os lemezre van szükségünk, mint amennyire ráférnek a base (alapterjesztés) könyvtárban található állományok. Ha DOS-ban hozzuk létre ezeket a lemezeket, akkor a használatukhoz meg *kell* formázni ezeket az MS-DOS® `FORMAT` parancsával. Windows® használata esetén az Windows Explorerben (Intézõben) tudjuk megformázni a lemezeket

(kattintsunk a jobb gombbal az A: meghajtóra, majd válasszuk a "Format" (Formázás) menüpontot).

Ne bízzunk a gyárilag formázott ("pre-formatted" jelzésû) lemezekben! Menjünk biztosra és formázzuk meg mi magunk is lemezeket. A felhasználóinktól régebben számtalan olyan panasz érkezett, amely a helytelenül megformázott lemezbõl fakadt, ezért erre most kiemelten felhívjuk a figyelmet.

A formázás abban az esetben sem bizonyul rossz ötletnek, ha egy másik FreeBSD gépen gyártjuk le a lemezeket, habár nem kell mindegyik lemezre DOS állományrendszert tennünk. Helyette a `bsdlabel` és `newfs` parancsok használatával UFS állományrendszert is tehetünk rájuk, ahogy (1,44 MB méretû lemezek esetén) ezt az alábbi parancsok mutatják:

```
# fdformat -f 1440 fd0.1440
# bsdlabel -w fd0.1440 floppy3
# newfs -t 2 -u 18 -l 1 -i 65536 /dev/fd0
```

Ezután a többi állományrendszerhez hasonlóan a lemezeket tudjuk csatlakoztatni és írni.

Miután megformáztuk a lemezeket, rájuk kell másolnunk az állományokat. A terjesztésekhez tartozó állományokat adott méretû darabokra szeleteltük, így kényelmesen ráférnek egy hagyományos 1,44 MB méretû floppyra. Menjünk végig az összes floppyn és mindegyikre pakoljuk fel a lehetõ legtöbb állományt egészen addig, amíg így az összes szükséges terjesztést össze nem szedtük. A floppykon minden terjesztés kerüljön egy hozzá tartozó alkönyvtárba, például: a:\base\base.aa, a:\base\base.ab és így tovább.

Az elsõ lemezre rá kell másolnunk a base.inf nevû állományt is, mivel ennek beolvasásával lesz képes kitalálni a telepítõ, hogy a terjesztések összeszedése és összefûzése során mennyi darabot keressen.

Ahogy elérkezünk a telepítõeszköz kiválasztásához a telepítés folyamatában, ott válasszuk a Floppy menüpontot, majd utána kövessük a felbukkanó üzeneteket.

2.13.4. Telepítés MS-DOS® partícióról

Amikor egy MS-DOS® partícióról akarunk telepíteni, elõkészítés gyanánt másoljuk a terjesztésekhez tartozó állományokat a partícióra egy freebsd könyvtárba. Ez lesz például a c:\freebsd. Ebben a könyvtárban igyekezzük minél jobban megtartani a CD vagy az FTP oldal könyvtárszerkezetét, ezért erre a CD-rõl történõ átmásolásra a DOS `xcopy` parancsát javasoljuk. Például így tudjuk elõkészíteni a FreeBSD legegyszerûbb változatának telepítését:

```
C:\> md c:\freebsd
C:\> xcopy e:\bin c:\freebsd\bin\ /s
C:\> xcopy e:\manpages c:\freebsd\manpages\ /s
```

A fentiekben feltételeztük, hogy ehhez a C: meghajtón elég szabad helyünk van, valamint az E: meghajtón érjük el a CD-t.

Ha nincs CD-meghajtónk, az ftp.FreeBSD.org címrõl letölthetjük a terjesztésket. Minden egyes terjesztés külön könyvtárban található, tehát például a *base* (alap) terjesztés az 12.0/base/ könyvtárban található.

Mindegyik telepítendõ terjesztést (ami még elfér) másoljuk át az MS-DOS® partíció c:\freebsd könyvtárába - a telepítéshez egyébként egyedül a `BIN` terjesztés szükséges.

2.13.5. Telepítõszalag létrehozása

Valószínûleg a szalagos módszer a legegyszerûbb, egyfajta élõ FTP-s vagy CD-s telepítés. A telepítõprogram arra számít, hogy a szalagon az állományok egymás után helyezkednek el. Tehát miután beszereztük a nekünk kellõ terjesztésekhez tartozó összes állományt, egyszerûen vegyük fel ezeket a szalagra:

```
# cd /freebsd/distdir
# tar cvf /dev/rwt0 dist1 ... dist2
```

Mielõtt telepítenénk, ellenõrizzük, hogy legyen elég helyünk valamelyik (a telepítés során majd kiválasztható átmeneti) könyvtárban ahhoz, hogy az itt létrehozott szalag *teljes* tartalma elférjen benne. Mivel a szalagok csak szekvenciálisan érhetõek el, ezért ennél a módszernél jó sok ideiglenes tárhelyre lesz szükségünk.

A telepítés megkezdése után a szalagnak már *azelõtt* a meghajtóban kell lennie, hogy rendszerindító floppyról elindítanánk a rendszert, máskülönben nem találja meg.

2.13.6. Mielõtt hálózatról telepítenénk

Háromféle hálózati telepítési mód létezik: Ethernet (szabványos Ethernet-vezérlõvel), soros port (PPP) vagy párhuzamos port (PLIP (laplink kábel)).

Valószínûleg az Ethernet-csatlakozó választásával érjük el a leggyorsabb hálózati telepítést. A FreeBSD ismeri a legtöbb PC-s Ethernet kártyát. Az ismert kártyák (és a hozzájuk tartozó beállítások) a FreeBSD egyes kiadásának hardverjegyzékében (Hardware Notes) találhatóak meg. Amennyiben egy támogatott PCMCIA Ethernet kártyát használunk, mindig a laptop bekapcsolása *elõtt* helyezzük be! A FreeBSD telepítés közben sajnos nem támogatja a PCMCIA kártyák menetközbeni behelyezését.

Ezenkívül még ismernünk kell a hálózaton kapott IP-címünket, az általa használt címosztály hálózati maszkját, a gépünk nevét. Ha PPP kapcsolaton keresztül telepítünk és nincs statikus IP-címünk, akkor minden bizonnyal az internet-szolgáltatónktól kaptunk egyet dinamikusan. A konkrét hálózati beállításokat a hálózatunk rendszergazdájától is érdemes megkérdezni. Ha a hálózaton levõ többi gépre névvel és nem IP-címmel hivatkozunk, akkor szükségünk lesz még egy név(feloldó) szerverre és az internet eléréséhez egy átjáró címére is (ha PPP-t használunk, ez a szolgáltatónk IP-címe lesz). Ha FTP-rõl HTTP proxy használatával telepítünk, akkor a proxy címe is kelleni fog. Ha magunktól nem vagyunk képesek ezekre a kérdésekre válaszolni, akkor az ilyen típusú telepítés megkezdése *elõtt* tényleg segítséget kell kérnünk egy rendszergazdától vagy az

internet-szolgáltatónktól.

Ha modemet használunk, akkor a PPP szinte biztosan megfelel nekünk. Gondoskodjunk róla, hogy már a telepítés korai szakaszában rendelkezésünkre áll az internet-szolgáltatónkkal kapcsolatosan minden hasznos információ.

Ha PAP vagy CHAP használatával kapcsolódunk a szolgáltatónkhoz (másképp szólva Windows®-ban így tudunk szkriptek nélkül csatlakozni), mindössze a `dial` parancsot kell kiadnunk a ppp parancssorában. Minden más esetben tudnunk kell a modemünk saját "AT parancsaival" tárcsázni az internet-szolgáltatónkat, hiszen ehhez a PPP tárcsázó csak egy nagyon kezdetleges terminálemulációt nyújt. Ezzel kapcsolatban olvassuk el a kézikönyv és a GYIK idevágó részeit. Ha gondjaink akadnának, a naplózás a `set log local …` parancs kiadásával átirányítható közvetlenül a képernyõre.

Ha kötött módon tudunk csatlakozni egy másik (2.0-R vagy késõbbi verziójú) FreeBSD géphez, akkor megpróbálkozhatunk a párhuzamos "laplink" kábellel. A párhuzamos porton keresztüli adatátvitel sebessége a soros vonalénál jóval nagyobb (egészen 50 kbyte/mp), ezért vele a telepítés is gyorsabb.

Mielõtt NFS-rõl telepítenénk

A telepítés NFS-en keresztül szinte magától értetõdik. Egyszerûen csak másoljuk a FreeBSD terjesztéseihez tartozó állományokat az NFS szerverre és állítsuk be rá az NFS telepítõeszközt.

Ha a szerver csak "privilegizált portokat" ismer (ami általában alapértelmezett a Sun munkaállomásoknál), a telepítés megkezdése elõtt az **Options** (Beállítások) menüben be kell állítani az `NFS Secure` (Biztonságos NFS) opciót.

Ha egy gyenge minõségû és kis adatátviteli sebességû Ethernet kártyánk van, akkor emellett még hasznos lehet beállítani az `NFS Slow` (Lassú NFS) opciót is.

Az NFS-en keresztüli telepítés mûködéséhez a szervernek támogatnia kell az alkönyvtárak csatlakoztatását is, tehát például ha a FreeBSD 12.0 terjesztésünk a ziggy:/usr/archive/stuff/FreeBSD könyvtárban található, akkor `ziggy` nevû gépnek lehetõvé kell tennie a /usr/archive/stuff/FreeBSD könyvtár közvetlen csatlakoztatását is, nem csak a /usr vagy /usr/archive/stuff könyvtárakét.

A FreeBSD /etc/exports állományában ezt az `-alldirs` beállítással vezérelhetjük. Más NFS szervereken esetleg más megszokásokat kell követnünk. Amennyiben a szervertõl `permission denied` (hozzáférés megtagadva) üzeneteket kapjuk, valószínû, hogy ezt nem állítottuk be megfelelõen.

Chapter 3. A UNIX alapjai

3.1. Áttekintés

Ez a fejezet a FreeBSD operációs rendszer alapvetõ funkcióit és parancsait mutatja be. Az itt tárgyalásra kerülõ anyag nagy része érvényes bármelyik más UNIX®-szerû operációs rendszer esetén is. Ezért, ha már ismerjük az említésre kerülõ ismereteket, minden további gond nélkül átugorhatjuk ezt a fejezetet. Azonban ha még teljesen ismeretlen számunkra a FreeBSD, minden bizonnyal ez lesz az, amit alaposan át kell majd olvasnunk.

A fejezet elolvasása során megismerjük:

- az ún. "virtuális konzolok" használatát FreeBSD alatt;
- hogyan mûködnek együtt a UNIX® állományokra vonatkozó engedélyei a FreeBSD saját kiegészítéseivel;
- egy FreeBSD állományrendszer alapértelmezett kialakítását;
- a FreeBSD lemezszervezését;
- hogyan csatlakoztassunk és válasszunk le állományrendszereket;
- mik azok a folyamatok, démonok és jelzések;
- mik azok a parancsértelmezõk, és miként tudjuk megváltoztatni az alapértelmezett bejelentkezési környezetünket;
- hogyan használjuk az alapvetõ szövegszerkesztõket;
- mik az eszközök és az eszközleírók;
- FreeBSD alatt milyen bináris formátumokat használhatunk;
- szükség esetén hogyan olvassuk el a megfelelõ man oldalakat.

3.2. Virtuális konzolok és terminálok

A FreeBSD számos módon használható. Ezek közül az egyik az, ha parancsokat gépelünk be a szöveges terminálon. Így érhetõ el egyszerûen a UNIX® operációs rendszer rugalmasságának és erejének jelentõs része. Ebben a szakaszban megtudhatjuk, mik azok a "terminálok" és "konzolok" és miként tudjuk ezeket FreeBSD alatt használni.

3.2.1. A konzol

Ha nem állítottuk volna be, hogy a FreeBSD indulása során automatikusan induljon el a grafikus felület is, akkor a rendszer egy bejelentkezõ képernyõt fog mutatni közvetlenül a rendszerindítás befejezõdése után. Ekkor valami ilyesmit kell majd látnunk:

```
Additional ABI support:.
Local package initialization:.
Additional TCP options:.
```

```
Fri Sep 20 13:01:06 EEST 2002

FreeBSD/i386 (pc3.example.org) (ttyv0)

login:
```

Egyes rendszereken ugyan némileg eltérhetnek az üzenetek, de hasonlót kell látnunk. Minket most az utolsó két sor érdekel. Az utolsó elõtti sorban ez olvasható:

```
FreeBSD/i386 (pc3.example.org) (ttyv0)
```

Ez a sor arról értesít minket, hogy a rendszerünk éppen most indult el: egy "FreeBSD" konzolt látunk, amely egy Intel® x86 architektúrájú processzoron fut . A gépünk neve (mivel minden UNIX®-os gép rendelkezik egy névvel) `pc3.example.org`, és ennek a rendszerkonzolját látjuk most éppen - a ttyv0 terminált.

Végezetül az utolsó sor mindig:

```
login:
```

Ez az a rész, ahova a FreeBSD-be történõ bejelentkezéshez meg kell adnunk a "felhasználói nevünket" (user name). A következõ szakaszban errõl olvashatunk.

3.2.2. Bejelentkezés a FreeBSD-be

A FreeBSD egy többfelhasználós, többfeladatos rendszer. Így hívják hivatalosan azokat a rendszereket, amelyeket többen tudnak használni és egyetlen számítógépen egyszerre rengeteg programot képesek futtatni.

Minden többfelhasználós rendszernek valamilyen módon meg kell tudnia különböztetnie egy "felhasználóját" a többitõl. A FreeBSD-ben (és minden más UNIX®-szerû operációs rendszerben) ezt úgy érik el, hogy a programok futtatása elõtt minden felhasználónak "be kell jelentkeznie" a rendszerbe. Minden felhasználó rendelkezik egy egyedi névvel (ez a "felhasználói név") és ehhez egy titkos kulcssal (ez a "jelszó"). A FreeBSD a programok futtatásához ezt a kettõt fogja elkérni a felhasználótól.

Egybõl miután a FreeBSD elindult és befejezte a rendszerindításhoz használt szkriptjeinek lefuttatását , ez a kijelzés (vagy más néven "prompt") fog megjelenni és kér egy érvényes felhasználói nevet:

```
login:
```

A példa kedvéért most tegyük fel, hogy a felhasználói nevünk `pgj`. Az iménti prompthoz írjuk be, hogy `pgj` és nyomjuk le az `Enter` billentyût. Ezt követõen meg kell jelennie egy másik promptnak is, amely egy "jelszót" (password) kér:

```
login: pgj
Password:
```

Most pedig gépeljük be `pgj` jelszavát és nyomjunk utána egy `Enter` billentyût. Vigyázzunk, hogy a jelszót *nem látjuk* a beírás során! Emiatt most ne aggódjunk. Ezzel kapcsolatban elegendõ csak annyit tudni, hogy mindez biztonsági megfontolásokból történik.

Amennyiben jól adtuk meg a jelszavunkat, sikeresen bejelentkezünk a FreeBSD rendszerébe és készen állunk az összes elérhetõ parancs kipróbálására.

Bejelentkezés után a MOTD (message of the day) vagy más néven "a nap üzenete" jelenik meg, amelyet a parancssor követ (egy `#`, `$` vagy `%` jel). Innen tudhatjuk meg, hogy sikerült bejelentkeznünk.

3.2.3. Több konzol használata

A UNIX® parancsokat egy konzolon is szépen ki tudjuk adni, de a FreeBSD egyszerre ugyebár több programot is tud futtatni. A parancsok megadásához viszont egyetlen konzol használata elég nagy pazarlás lenne, hiszen egy olyan operációs rendszer mint a FreeBSD, tucatnyi programot képes futtatni egy idõben. Ebben az esetben jelenthetnek számunkra segítséget a "virtuális konzolok".

A FreeBSD beállítható úgy, hogy sok-sok különféle virtuális konzolt ajánljon fel számunkra. A virtuális konzolok között a billentyûzeten a megfelelõ gombok lenyomásával tudunk váltani. Mindegyik konzolnak megvan a saját kimeneti csatornája, és a virtuális konzolok közti váltás folyamán a FreeBSD gondoskodik a billentyûzetrõl érkezõ bemenet, valamint a monitorra irányított kimenet megfelelõ kezeléséről.

A konzolok közti váltásra a FreeBSD külön billentyûkombinációkat tart fenn . A FreeBSD-ben a különbözõ virtuális konzolok közti váltásra az `Alt`+`F1`, `Alt`+`F2` billentyûket, az `Alt`+`F8` billentyûkombinációval bezárólag használhatjuk.

A konzolok közti váltogatás során a FreeBSD ügyel a képernyõ tartalmának elmentésére és visszaállítására. Ennek eredményeképpen "úgy látszik", mintha több "virtuális" képernyõn és billentyûzeten adnánk parancsokat a FreeBSD-nek.

3.2.4. Az /etc/ttys állomány

A FreeBSD alapértelmezés szerint nyolc virtuális konzollal indul. Ez azonban nem egy elõre rögzített érték, hiszen könnyedén testreszabhatjuk úgy a telepített rendszerünket, hogy több vagy esetleg kevesebb virtuális konzollal induljon el. A virtuális konzolok száma és azok pontos beállítása az /etc/ttys állományon keresztül adható meg.

A FreeBSD virtuális konzoljait tehát az /etc/ttys állomány megfelelõ módosításával tudjuk behangolni. Itt minden egyes olyan sor, amely nem megjegyzés (vagyis azok a sorok, amelyek nem a `#` karakterrel kezdõdnek), tartalmazza az egyes terminálok vagy virtuális konzolok beállításait. Az állomány a FreeBSD telepítésében szereplõ, alapértelmezett változata kilenc virtuális konzol konfigurációját tartalmazza, amelyek közül nyolc aktív. Ezek a `ttyv` résszel kezdõdõ sorok:

```
# name  getty                           type    status          comments
#
ttyv0   "/usr/libexec/getty Pc"         cons25  on  secure
# Virtual terminals
ttyv1   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv2   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv3   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv4   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv5   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv6   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv7   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv8   "/usr/X11R6/bin/xdm -nodaemon"  xterm   off secure
```

Az állományban található oszlopok kimerítõ magyarázatát, illetve a virtuális konzolok beállításához használható kapcsolókat a ttys(5) man oldalon olvashatjuk.

3.2.5. Az egyfelhasználós mód konzolja

Az "egyfelhasználós mód" részletes leírása a Egyfelhasználós módban található. Fontos tudni, hogy amikor a FreeBSD-t egyfelhasználós módban futtatjuk, csupán egyetlen konzolunk van, és a virtuális konzolok nem érhetõek el. Egyébként az egyfelhasználós mód erre vonatkozó beállításai is megtalálhatóak az /etc/ttys állományban. Ehhez keressük meg a `console` kezdetû sort:

```
# name  getty                           type    status          comments
#
# Ha a konzolt "insecure" (nem biztonságos) típusúnak választjuk meg,
# akkor a használatához az egyfelhasználós mód aktivilásá elõtt a rendszer
# kérni fogja a rendszeradminisztrátori jelszót.
console none                            unknown off secure
```

A `console` felett látható megjegyzés jelzi, hogy át tudjuk írni ebben a sorban a `secure` (biztonságos) értékû paramétert `insecure` (nem biztonságos) értékûre. Ilyenkor, hogy ha a FreeBSD egyfelhasználós módban indul, kérni fogja a `root` felhasználó (a rendszeradminisztrátor) jelszavát.

Vigyázzunk, amikor ezt az értéket `insecure`-ra állítjuk! Ha ugyanis véletlenül elfeledkeznénk a `root` jelszaváról, akkor azzal az egyfelhasználós mód használata is veszélybe kerülhet. Habár ettõl függetlenül is lehetséges, azokra számára mégis nehéz helyzetnek bizonyulhat, akik nem mozognak elég otthonosan a FreeBSD rendszerindítási folyamatának és a hozzákapcsolódó programok ismeretében.

3.2.6. A videomód váltása konzolban

A FreeBSD konzol alapértelmezett videomódja átállítható 1024x768-ra, 1280x1024-re, vagy bármilyen olyan más méretre, amit a videokártyánk és monitorunk képes megjeleníteni. Az eltérõ videomódok használatához elõször újra kell fordítanunk a rendszermagunkat az alábbi két beállítás hozzáadásával:

```
options VESA
options SC_PIXEL_MODE
```

Miután a rendszermagot sikeresen újrafordítottuk a fenti beállításokkal, a vidcontrol(1) segédprogrammal tudjuk megállapítani, hogy a hardverünk milyen videomódokat enged használni. Az összes támogatott videomódot a következõképpen tudjuk lekérdezni:

```
# vidcontrol -i mode
```

A parancs eredményeképpen tehát megkapjuk a hardverünk által ismert videomódokat. Ezek közül tudjuk kiválasztani valamelyikõjüket és `root` felhasználóként a vidcontrol(1) segítségével beállítani:

```
# vidcontrol MODE_279
```

Ha az új videomód megfelel számunkra, akkor ezt a beállítást az /etc/rc.conf állományon keresztül véglegesíthetjük is:

```
allscreens_flags="MODE_279"
```

3.3. Engedélyek

A FreeBSD, mivel a BSD UNIX® egyik közvetlen leszármazottja, számos UNIX®-os alapötletre épül. Ezek közül az elsõ és talán a leginkább kihangsúlyozott, hogy a FreeBSD egy többfelhasználós operációs rendszer. Egy olyan rendszer, amely egyszerre több, egymástól független feladattal foglalkozó felhasználót képes kiszolgálni. A rendszer felelõs a hardveres eszközök, a különféle perifériák, a memória és a processzor idejének minden egyes felhasználó számára szabályos és pártatlan megosztásáért és a feléjük irányuló kérések szervezéséért.

Mivel a rendszer több felhasználót is képes támogatni, az általa kezelt erõforrások rendelkeznek engedélyek egy adott halmazával, amelyek eldöntik, ki tudja ezeket olvasni, írni és végrehajtani. Az engedélyek háromszor három bit formájában jelennek meg, amelyek közül az elsõ bitcsoport az állomány tulajdonosára, a második az állomány csoportjára, végül az utolsó pedig a mindenki másra vonatkozó engedélyeket tárolja.

Érték	Engedély	Könyvtárlistában
0	Nem olvasható, nem írható, nem hajtható végre	`---`
1	Nem olvasható, nem írható, végrehajtható	`--x`
2	Nem olvasható, írható, nem hajtható végre	`-w-`

Érték	Engedély	Könyvtárlistában
3	Nem olvasható, írható, végrehajtható	`-wx`
4	Olvasható, nem írható, nem hajtható végre	`r--`
5	Olvasható, nem írható, végrehajtható	`r-x`
6	Olvasható, írható, nem hajtható végre	`rw-`
7	Olvasható, írható, végrehajtható	`rwx`

A ls(1) `-l` kapcsolójának segítségével megnézhetjük a könyvtárak tartalmának részletes listáját, amiben megjelennek az állományok tulajdonosaira, csoportjára és a mindenki másra vonatkozó engedélyek is. Például ezt láthatjuk, ha kiadjuk az `ls -l` parancsot egy tetszõleges könyvtárban:

```
% ls -l
total 530
-rw-r--r--  1 root  wheel     512 Sep  5 12:31 egyik
-rw-r--r--  1 root  wheel     512 Sep  5 12:31 masik
-rw-r--r--  1 root  wheel    7680 Sep  5 12:31 e-mail.txt
...
```

A példabeli `ls -l` parancs kimenetének elsõ oszlopa így bomlik fel:

```
-rw-r--r--
```

Az elsõ (bal szélsõ) karakter mondja meg, hogy ez egy hagyományos állomány, könyvtár, speciális karakteres eszköz, socket vagy bármilyen más különleges pszeudoállomány. Ebben az esetben a `-` jelzi, hogy egy hagyományos állományról van szó. A következõ három karakter, ami ebben a példában az `rw-`, adja meg az állomány tulajdonosának engedélyeit. Az ezután következõ három karakter, az `r--` mutatja az állomány csoportjának engedélyeit. Az utolsó három karakter, vagyis itt az `r--` adja meg a többiek engedélyeit. A kötõjel arra utal, hogy az adott engedélyû tevékenység nem engedélyezett. Tehát ennél az állománynál az engedélyek a következõek: a tulajdonosa tudja olvasni és írni, a csoportja csak olvasni tudja, ugyanígy bárki más. A fenti táblázatnak megfelelõen az állomány engedélyének kódja `644` lesz, ahol az egyes számjegyek jelentik az állomány engedélyeinek három elemét.

Ez mind szép és jó, de vajon a rendszer milyen módon kezeli az állományok engedélyeit? A FreeBSD a legtöbb hardveres eszközt állománynak tekinti, amelyeket a programok meg tudnak nyitni, tudnak róluk olvasni és adatokat tudnak kiírni rájuk pontosan úgy, mint bármilyen más állomány esetén. Ezeket a speciális állományokat a /dev könyvtárban találjuk.

A könyvtárakat is állományokként kezeli, ezért azok is rendelkeznek olvasási, írási és végrehajtási engedélyekkel. Azonban a könyvtárak végrehajtását engedélyezõ bit némileg más jelentéssel bír, mint az állományok esetén. Amikor ugyanis egy könyvtárat végrehajthatónak jelölünk meg, az arra

fog utalni, hogy bele tudunk lépni, vagyis hogy ki tudjuk rá adni a "könyvtárváltás" (cd, change directory) parancsát. Ez továbbá arra is utal, hogy az ismert nevû állományokhoz hozzá tudunk férni (természetesen az egyes állományok engedélyeinek megfelelõen).

A könyvtárak tartalmát ennek megfelelõen viszont csak úgy láthatjuk, ha olvasási engedéllyel rendelkezünk a könyvtárra, míg egy általunk ismert állomány törléséhez a tartalmazó könyvtárhoz kell írási *és* végrehajtási engedélyekkel rendelkeznünk.

Ezeken kívül még léteznek további engedélyek is, de ezeket csak olyan különleges esetekben használják, mint például a felhasználóváltó programok (setuid program) vagy a ragadós könyvtárak (sticky directory) létrehozása. Az állományok engedélyeinek behatóbb megismeréséhez és beállításához mindenképpen nézzük át a chmod(1) man oldalt.

3.3.1. Szimbolikus engedélyek

A szimbolikus engedélyek (gyakran csak szimbolikus kifejezések) az állományok és könyvtárak engedélyeinek megadása során a számok helyett karaktereket használnak. A szimbolikus kifejezések (ki) (hogyan) (milyen engedélyt) alakúak, ahol az alábbi értékek adhatóak meg:

Elem	Betû	Jelentése
(ki)	u	tulajdonos
(ki)	g	csoport tulajdonos
(ki)	o	egyéb
(ki)	a	mindenki (a "világ")
(hogyan)	+	engedély megadása
(hogyan)	-	engedély visszavonása
(hogyan)	=	engedély explicit beállítása
(milyen engedély)	r	olvasás
(milyen engedély)	w	írás
(milyen engedély)	x	végrehajtás
(milyen engedély)	t	ragadós (sticky bit)
(milyen engedély)	s	UID vagy GID állítása

Ezek az értékek a chmod(1) paranccsal az eddigiekhez hasonló módon használhatóak, csak itt betûket kell megadnunk. Például az alábbi paranccsal akadályozhatjuk meg, hogy a tulajdonosán kívül bárki hozzáférhessen az *ÁLLOMÁNY* nevû állományhoz:

```
% chmod go= ÁLLOMÁNY
```

Amennyiben egy állománnyal kapcsolatban több változtatást is el kívánunk végezni, össze tudjuk ezeket fûzni egy vesszõkkel elhatárolt felsorolásban:

```
% chmod go-w,a+x ÁLLOMÁNY
```

3.3.2. A FreeBSD állományjelzõi

A korábban tárgyalt engedélyek mellett még a FreeBSD ismeri az ún. "állományjelzõk" (file flags) beállítását is. Ezek a jelzõbitek egy további biztonsági és irányítási szintet nyújtanak az állományok felett, viszont a könyvtárakra nem vonatkoznak.

Ezek az állományjelzõk az állományok felett további vezérlést adnak a kezünkbe, aminek révén gondoskodhatunk róla, hogy akár még a `root` felhasználó (a rendszer adminisztrátora) se legyen képes állományokat eltávolítani vagy módosítani.

Az állományjelzõk értékei egy egyszerû felületen keresztül, a chflags(1) segédprogrammal változtathatóak meg. Például a következõ paranccsal állíthatjuk a rendszer törölhetetlen (undeletable) jelzését az allomany1 állományon:

```
# chflags sunlink allomany1
```

A törölhetetlen jelzés eltávolításához egyszerûen csak írjuk be az elõzõ parancsot úgy, hogy a "sunlink" paraméter elejére még beszúrunk egy "no" szövegrészt. Így:

```
# chflags nosunlink allomany1
```

Az állományokra éppen érvényes jelzéseket az ls(1) parancs `-lo` kapcsolójának segítségével jeleníthetjük meg:

```
# ls -lo file1
```

Ennek megfelelõen az eredménynek valahogy így kellene kinéznie:

```
-rw-r--r--  1 trhodes  trhodes  sunlnk 0 Mar  1 05:54 allomany1
```

Sok jelzés csak a `root` felhasználón keresztül vehetõ fel vagy távolítható el. Más esetekben viszont az állomány tulajdonosa állíthatja ezeket. A rendszergazdáknak javasoljuk, hogy ezzel kapcsolatban a chflags(1) és chflags(2) man oldalakat tanulmányozzák át.

3.3.3. A setuid, setgid és sticky engedélyek

A korábban említett engedélyeken kívül létezik még további három, amelyekkel minden rendszergazdának illik tisztában lennie. Ezek név szerint a `setuid`, `setgid` és `sticky` típusú engedélyek.

Ezek a beállítások bizonyos UNIX® mûveletek esetén nagyon fontosak, mivel az átlagos felhasználók számára általában el nem érhetõ funkciók használatát támogatják. A megértésükhöz

elsõként a felhasználók valódi és effektív azonosítója közti különbségeket kell tisztáznunk.

A valódi azonosító tulajdonképpen az a felhasználói azonosító, amellyel a programot indítjuk el vagy futás elõtt birtokoljuk. A program futása közben azonban az effektív felhasználói azonosítóval fut. Például a passwd(1) segédprogram a jelszavát megváltoztatni kívánó felhasználó valódi azonosítójával indul, miközben a jelszavakat tároló adatbázis elérésékor már a `root` felhasználó effektív azonosítójával fut. Ezáltal a privilégiumokkal nem rendelkezõ felhasználók is meg tudják anélkül változtatni a jelszavaikat, hogy a `Permission Denied` hibaüzenettel találkoznának.

A mount(8) `nosuid` beállításával azonban az ilyen típusú binárisok minden különösebb jel nélkül csõdöt fognak mondani. Mellesleg a mount(8) man oldala szerint ez az opció nem is teljesen megbízható, mivel `nosuid` wrapperek segítségével meg lehet kerülni.

Ahogy azt az alábbi példa is szemlélteti, a setuid engedélyt a többi elé egy négyes (4) beszúrásával tudjuk beállítani:

```
# chmod 4755 suidexample.sh
```

A suidexample.sh állomány engedélyei ezt követõen már így fognak megjelenni:

```
-rwsr-xr-x   1 trhodes  trhodes    63 Aug 29 06:36 suidexample.sh
```

Most már jól látható, hogy az állomány tulajdonosához tartozó engedélyek között a végrehajthatóságot szabályozó bit lecserélõdött egy `s` bitre. Ennek köszönhetõen a `passwd` parancshoz hasonló módon kibõvített engedélyekkel leszünk képesek futtatni programokat.

Két terminál megnyitásával mindezt valós idõben is megvizsgálhatjuk. Az egyiken indítsuk el normál felhasználóként a `passwd` programot. Miközben a program várakozik az új jelszó megadására, a másik terminálon kérdezzük le a programhoz tartozó felhasználói információkat.

Tehát az egyik terminálon a következõt látjuk:

```
% passwd
Changing local password for trhodes
Old Password:
```

Eközben pedig a másikon:

```
# ps aux | grep passwd
trhodes  5232  0.0  0.2  3420  1608   0  R+    2:10AM   0:00.00 grep passwd
root     5211  0.0  0.2  3620  1724   2  I+    2:09AM   0:00.01 passwd
```

A `passwd` parancsot egyszerû felhasználóként adtunk ki, azonban jól látható, hogy valójában a `root` felhasználó azonosítójával fut.

A `setgid` a `setuid` engedélyhez hasonlóan mûködik, egyedül annyiban tér el, hogy a csoportra vonatkozó beállításokat módosítja. Amikor egy alkalmazást vagy segédprogramot ilyen engedéllyel futtatunk, akkor az adott programot birtokló csoport engedélyeit kapjuk meg.

Úgy tudjuk állományokon beállítani a `setgid` típusú engedélyt, ha az iménti példához hasonlóan a `chmod` parancs hívásakor még egy kettest (2) írunk az engedélyek elé:

```
# chmod 2755 sgidexample.sh
```

Az így beállított engedélyek az elõbbihöz hasonló módon szemlélhetõek meg, azonban ebben az esetben a csoporthoz tartozó engedélyeknél jelenik meg az `s` bit:

```
-rwxr-sr-x   1 trhodes  trhodes    44 Aug 31 01:49 sgidexample.sh
```

Az elõbb tárgyalt példákkal kapcsolatban fontos megemlítenünk, hogy habár a szkriptek is végrehajtható állományok, nem fognak a valóditól eltérõ effektív felhasználói azonosítóval futni. Ennek oka abban keresendõ, hogy a parancssori szkriptek nem hívhatják a setuid(2) rendszerhívást.

Ez a két speciális engedély (a `setuid` és a `setgid`) a programhoz tartozó engedélyek kiterjesztésével csökkentheti rendszerünk biztonságát. Ezzel szemben viszont a harmadik bemutatandó speciális engedély rendszerünk védelmének erõsítésére szolgál: ez az ún. `sticky` bit.

Ha a `sticky` típusú engedélyt könyvtárra adjuk meg, akkor a benne levõ állományok törlését kizárólag azok tulajdonosainak engedi. Ezzel az engedéllyel lényegében a /tmp könyvtárhoz hasonló nyilvános, bárki által elérhetõ könyvtárakban akadályozhatjuk meg az állományok idegen felhasználók általi törlését. Az engedély beállításához egy egyest (1) kell a többi elé fûznünk, mint például:

```
# chmod 1777 /tmp
```

Most már az `ls` parancs segítségével láthatjuk ennek a hatását:

```
# ls -al / | grep tmp
drwxrwxrwt  10 root  wheel          512 Aug 31 01:49 tmp
```

A `sticky` bit a beállítások végén felbukkanó `t` révén azonosítható be.

3.4. A könyvtárak elrendezése

A FreeBSD könyvtárszerkezetének ismerete alapvetõ jelentõségû a rendszer egészének megértése szempontjából. Ezen belül is a legfontosabb a gyökérkönyvtár, a "/". Ez az elsõ könyvtár, amelyet a rendszer a rendszerindítás során csatlakoztat és a többfelhasználós mód elõkészítéséhez elegendhetlenül szükséges alaprendszert tartalmazza. A gyökérkönyvtár emellett csatlakozási

pontokat szolgáltat a többfelhasználós mûködésre váltás során csatlakoztatandó további állományrendszerek számára.

A csatlakozási pont egy olyan könyvtár, ahová a szülõ állományrendszeren (ami gyakran maga a gyökér-állományrendszer) belül további állományrendszereket tudunk beoltani. Errõl bõvebben a A lemezek szervezéseban olvashatunk. A szabványos csatlakozási pontok: /usr, /var, /tmp, /mnt és /cdrom. Ezekre a könyvtárakra általában az /etc/fstab állományban találunk hivatkozásokat. Az /etc/fstab állomány a rendszer számára a különbözõ állományrendszerek és a hozzájuk tartozó csatlakozási pontok táblázatát tartalmazza. Az /etc/fstab állományban szereplõ legtöbb állományrendszer a rendszerindítás során automatikusan csatlakoztatásra kerül az rc(8) szkriptbõl, hacsak nem tartalmazzák a `noauto` beállítást. Ennek részleteit a Az fstab állományban találhatjuk meg.

Az állományrendszerek hierarchiájának teljes leírását a hier(7) man oldalon olvashatjuk. Mi egyelõre most megelégszünk a leggyakrabban megjelenõ könyvtárak rövid áttekintésével.

Könyvtár	**Mi található itt**
/	Az állományrendszer gyökere.
/bin/	Az egy- és többfelhasználós környezetekben is egyaránt alapvetõ felhasználói segédprogramok.
/boot/	Az operációs rendszer indítása során használt programok és konfigurációs állományok.
/boot/defaults/	A rendszerindítás alapértelmezett konfigurációs állományai. Lásd loader.conf(5)
/dev/	Eszközleírók, lásd intro(4).
/etc/	Rendszerkonfigurációs állományok és szkriptek.
/etc/defaults/	Az alapértelmezett rendszerkonfigurációs állományok, lásd rc(8).
/etc/mail/	A sendmail(8) programhoz hasonló levélküldõ rendszerek konfigurációs állományai.
/etc/namedb/	A `named` program konfigurációs állományai, lásd named(8).
/etc/periodic/	A cron(8) által naponta, hetente és havonta lefuttatandó szkriptek, lásd periodic(8).
/etc/ppp/	A `ppp` program konfigurációs állományai, lásd ppp(8).
/mnt/	Egy üres könyvtár, amelyet a rendszergazdák általában ideiglenes csatlakozási pontként használnak.
/proc/	A futó programokat tartalmazó állományrendszer, lásd procfs(5), illetve mount_procfs(8).

Könyvtár	Mi található itt
/rescue/	Statikusan linkelt programok vészhelyzet esetére, lásd rescue(8).
/root/	A `root` felhasználó könyvtára.
/sbin/	Az egy- és többfelhasználós környezetekben fontos rendszerprogramok és rendszerfelügyeleti eszközök.
/tmp/	Átmeneti állományok. A /tmp könyvtár tartalma általában NEM marad meg az újraindítás után. Erre a célra gyakran memóriában létrehozott állományrendszert szoktak csatlakoztatni a /tmp könyvtárba. Ez utóbbit az rc.conf(5) tmpmfs-re vonatkozó változóinak beállításával lehet automatikussá tenni (vagy a /etc/fstab megfelelõ módosításával, lásd mdmfs(8)).
/usr/	A felhasználói programok és alkalmazások többsége.
/usr/bin/	Általános segédprogramok, programozási eszközök és alkalmazások.
/usr/include/	Szabványos C include-állományok.
/usr/lib/	Függvénykönyvtárak.
/usr/libdata/	Egyéb hasznos adatállományok.
/usr/libexec/	(Más programok által használt) Rendszerdémonok és rendszereszközök.
/usr/local/	A helyi rendszeren telepített programok, függvénykönyvtárak stb. A FreeBSD portrendszere is ezt használja alapértelmezés szerint. A /usr/local könyvtáron belül a hier(7) man oldalon található /usr könyvtár általános felépítése használatos. Ez alól kivételt képez a man alkönyvtár, amely közvetlenül a /usr/local alatt található, nem pedig a /usr/local/share könyvtáron belül, valamint a portok dokumentációja a share/doc/port könyvtárban található.
/usr/obj/	A /usr/src könyvtárfában található források fordítása során keletkezõ architektúrafüggõ objektumok.
/usr/ports/	A FreeBSD Portgyûjtemény (választható).
/usr/sbin/	(A felhasználók által használt) Rendszerdémonok és rendszereszközök.

Könyvtár	Mi található itt
/usr/shared/	Architektúrafüggõ állományok.
/usr/src/	BSD és/vagy helyi források.
/usr/X11R6/	Az X11R6 rendszer programjai, függvénykönyvtárai stb. (választható)
/var/	Különféle napló, átmeneti, ideiglenes és pufferben tárolt állományok. A memóriában létrehozott állományrendszereket is olykor a /var könyvtárban találjuk. Ezt az rc.conf(5) állományban található varmfs-változók beállításával tehetjük automatikussá (vagy a /etc/fstab megfelelõ módosításával, lásd mdmfs(8)).
/var/log/	Mindenféle rendszernaplók.
/var/mail/	A felhasználók postafiókjait tároló állományok.
/var/spool/	A nyomtatók és a levelezés pufferéléséhez használt könyvtárak.
/var/tmp/	Átmeneti állományok. Az itt található állományok általában megmaradnak a következõ rendszerindítás alkalmával is, hacsak a /var nem egy memóriában létezõ állományrendszer.
/var/yp	A NIS állományai.

3.5. A lemezek szervezése

Az állománynév a legkisebb szervezési egység, amin keresztül a FreeBSD képes megtalálni az állományokat. Az állományok neveiben a kis- és nagybetût megkülönböztetjük, tehát a readme.txt és a README.TXT elnevezés két különbözõ állományra utal. A FreeBSD nem az állományok kiterjesztése (ami a konkrét példánkban a .txt volt) alapján dönti el, hogy az adott állomány vajon program, dokumentum vagy valamilyen más fajtájú adat.

Az állományok könyvtárakban tárolódnak. Egy könyvtár lehet akár üres (nincs benne egyetlen állomány sem), vagy többszáz állományt is tartalmazhat. Egy könyvtár ráadásul további könyvtárakat is tárolhat, és így az egymásban elhelyezkedõ könyvtárak segítségével könyvtárak egy hierarchiáját tudjuk felépíteni. Ezzel sokkalta könnyebben szervezhetõvé válnak az adataink.

Az állományokat és könyvtárakat úgy tudjuk elérni, ha megadjuk az állomány vagy a könyvtárt tároló könyvtár nevét, amit egy perjel, a / követ, valamint így összefûzve az eléréshez szükséges további könyvtárak felsorolása. Tehát, ha van egy ize nevû könyvtárunk, amelyben található egy mize könyvtár, amelyen belül pedig egy readme.txt, akkor ennek az állománynak a teljes neve, vagy másképpen szólva az *elérési útja* ize/mize/readme.txt lesz.

A könyvtárak és az állományok egy állományrendszerben tárolódnak. Minden állományrendszer

pontosan egy könyvtárat tartalmaz a legfelsõ szintjén, amelyet az adott állományrendszer *gyökérkönyvtárának* nevezünk. Ez a gyökérkönyvtár tartalmazhat aztán további könyvtárakat.

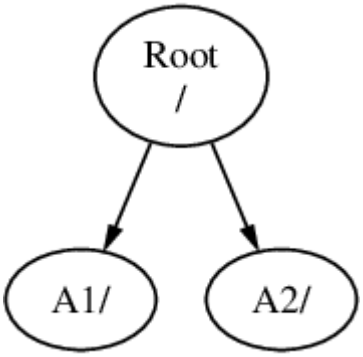
Eddig még valószínûleg minden nagyon hasonló a más operációs rendszerekben tapasztalható fogalmakhoz. Azonban adónak különbségek: például az MS-DOS® a `\` jellel választja el az állományok és könyvtárak neveit, miközben a Mac OS® erre a `:` jelet használja.

A FreeBSD az elérési utakban sem betûkkel, sem pedig semmilyen más névvel nem jelöli meg a meghajtókat. Tehát a FreeBSD-ben nem írhatjuk, hogy a c:/ize/mize/readme.txt.

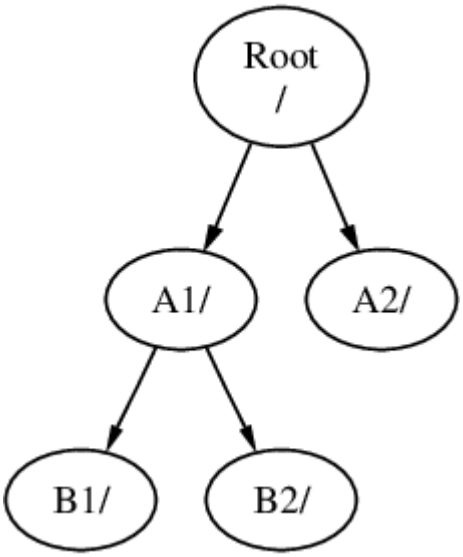
Helyette az egyik állományrendszert kijelölik *gyökér-állományrendszernek*. A gyökér-állományrendszer gyökérkönyvtárára hivatkoznak késõbb `/` könyvtárként. Ezután minden más állományrendszert a gyökér-állományrendszerhez *csatlakoztatunk*. Ennek értelmében nem számít, hogy mennyi lemezünk is van a FreeBSD rendszerünkben, hiszen minden könyvtár egyazon lemez részeként jelenik meg.

Tegyük fel, hogy van három állományrendszerünk, hívjuk ezeket `A`-nak, `B`-nek és `C`-nek. Minden állományrendszer rendelkezik egy gyökérkönyvtárral, amely két további könyvtárat tartalmaz: `A1` -et és `A2`-t (és ennek megfelelõen a többi `B1`-et és `B2`-t, valamint `C1` és `C2`-t).

Nevezzük `A`-t a gyökér-állományrendszernek. Ha a könyvtár tartalmának megjelenítéséhez most kiadnánk az `ls` parancsot, két alkönyvtárat látnánk, az `A1`-et és `A2`-t. A létrejött könyvtárfa valahogy így nézne ki:

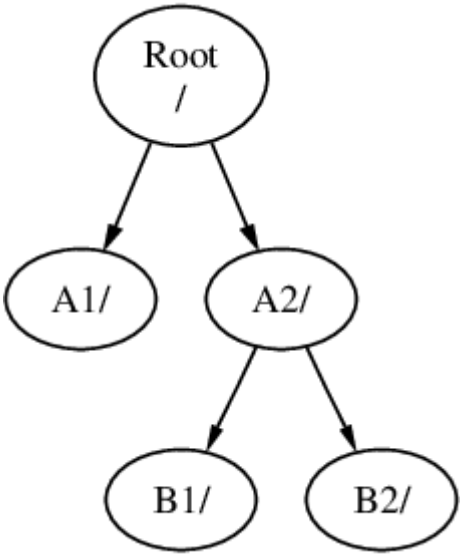


Egy állományrendszert csak egy másik állományrendszer valamelyik könyvtárába tudunk csatlakoztatni. Ezért most tételezzük fel, hogy a `B` állományrendszert az `A1` könyvtárba csatlakoztatjuk. Ezután a `B` gyökérkönyvtára átveszi a `A1` helyét az állományrendszerben, és ennek megfelelõen megjelennek a `B` könyvtárai is:



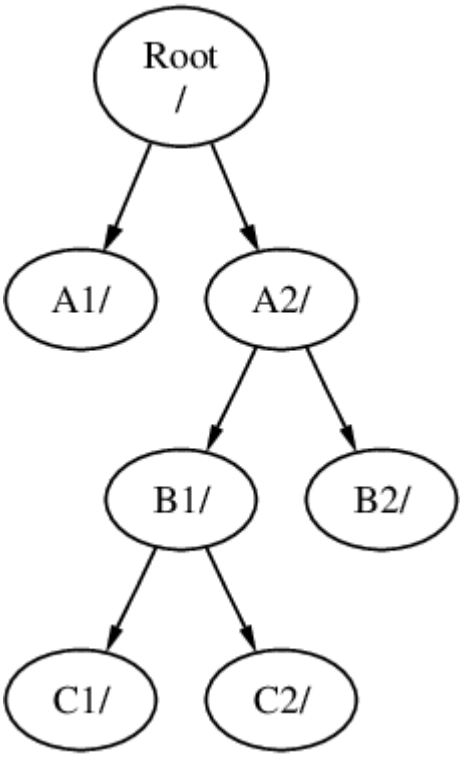
A B1 vagy B2 könyvtárakban található állományok bármelyike innentõl kezdve a /A1/B1, illetve a /A1/B2 elérési utakon érhetõek el. Az A1 könyvtárban található állományok erre az idõre rejtve maradnak. Akkor fognak újra felbukkanni, ha a B állományrendszert *leválasztjuk* az A állományrendszerrõl.

Ha a B állományrendszert az A2 könyvtárba csatlakoztatnánk, az iménti ábra nagyjából így nézne ki:

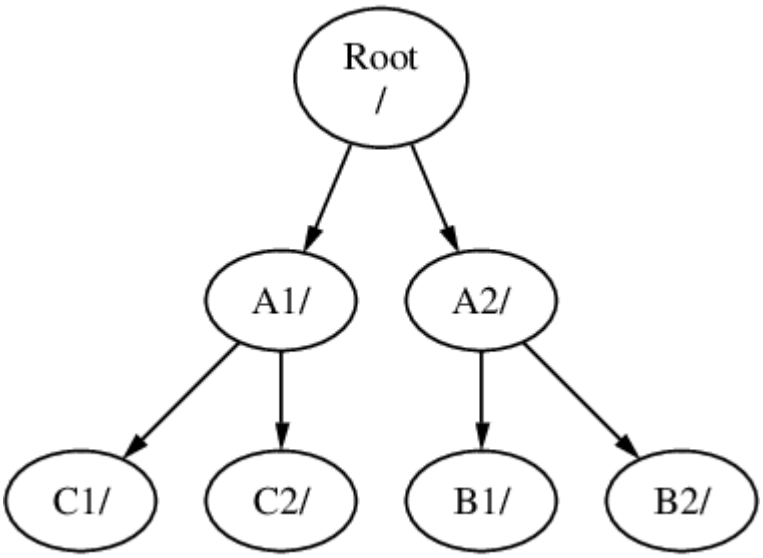


és ennek megfelelõen az elõbb tárgyalt elérési utak /A2/B1 és /A2/B2 lennének.

Az állományrendszerek egymáshoz is csatlakoztathatóak. A példát ennek megfelelõen úgy is folytathatjuk, hogy a C állományrendszert csatlakoztatjuk B állományrendszerben található B1 könyvtárhoz. Ennek eredménye a következõ elrendezés lesz:



Vagy a `C` állományrendszer az `A1` könyvtáron keresztül csatlakoztatható akár közvetlenül az `A` állományrendszerhez is:



Az MS-DOS® operációs rendszert ismerõk számára ez hasonló lehet a `join` parancshoz (habár teljesen nem egyezik meg vele).

Általában azonban ezzel nem kell törõdnünk, hiszen többnyire csak a FreeBSD telepítése során hozunk létre állományrendszereket és választjuk meg a csatlakozási pontjukat. A késõbbiekben ez legfeljebb akkor kerül elõ ismét, amikor újabb lemezeket adunk hozzá a rendszerhez.

Teljességgel megengedhetõ, hogy elhagyjuk a többit és csak egyetlen óriási gyökér-állományrendszert használjunk. Ennek viszont megvannak a maga hátrányai és az egyetlen elõnye.

Több állományrendszer használatának elõnyei

- A különbözõ állományrendszereknek különbözõ *csatlakoztatási beállításai* (mount options) lehetnek. Például, ha kellõen elõvigyázatosak akarunk lenni, a gyökér-állományrendszer írásvédett módon is csatlakoztatható, aminek köszönhetõen lehetetlenné válik a rendszer számára fontos állományok véletlen törlése vagy felülírása. Ha elkülönítjük a felhasználók számára írható állományrendszereket (például a /home könyvtárakat) a többi állományrendszertõl, lehetõvé válik számunkra, hogy *nosuid* beállítással csatlakoztassuk ezeket. Ez a beállítás megakadályozza, hogy ezekben a *suid*/*guid* bitekkel rendelkezõ végrehajtható állományok használhatóak legyenek, ezáltal növeli a rendszer biztonságosságát.
- A FreeBSD az állományrendszer használatától függõen magától határoz a benne található állományok optimális kiosztását illetõen. Így tehát a gyakorta módosított, kisebb állományokat tartalmazó állományrendszerek esetén teljesen más technikákat alkalmaz, mint például a nagyobb, kevésbé változó állományok esetén. Azonban egyetlen állományrendszer használatával ez a gyorsítási módszer odavész.
- Noha a FreeBSD állományrendszerei nagyon jól tûrik a hirtelen áramkimaradásokat, egy döntõ ponton bekövetkezõ váratlan leállás mégis kárt okozhat a szerkezetükben. Ha azonban több állományrendszerre osztjuk a tárolandó adatainkat, sokkal valószínûbbé válik, hogy egy ilyen eset után a rendszerünk talpra tud állni, és szükség esetén nekünk is könnyebb lesz a biztonsági mentéseinkbõl helyreállítani a sérült állományokat.

Egyetlen állományrendszer használatának elõnyei

- Az állományrendszerek mérete rögzített. Miután a FreeBSD telepítése során létrehoztunk egy adott méretû állományrendszert, elõfordulhat, hogy késõbb szükségünk lesz a méretének növelésére. Ilyenkor nehezen kerülhetjük el az ilyenkor szokásos teendõket: biztonsági mentés készítése, az új méretnek megfelelõ állományrendszer létrehozása, majd ezután a lementett adataink visszaállítása.

A FreeBSD-ben azonban megtalálható a growfs(8) parancs, amelynek segítségével az állományrendszerek mérete használat közben növelhetõ, és ezzel megszûnik a méretre vonatkozó korlátozás.

Az állományrendszerek partíciókban tárolódnak. A FreeBSD UNIX®-os eredete miatt azonban ez a kifejezés nem a hétköznapi "partíció" jelentését takarja (mint például egy MS-DOS® partíció). Minden partíciót egy betû azonosít a-tól h-ig. Mindegyik partíció csak egyetlen állományrendszert tartalmazhat, aminek révén az állományrendszereket vagy az állományrendszerek hierarchiájában található csatlakozási pontjukkal vagy pedig az ezeket tartalmazó partíció betûjével azonosíthatjuk.

A FreeBSD ezeken felül külön lemezterülen tárolja a *lapozóállomány*t (swap space). A lapozóállományt használja a FreeBSD *virtuális memória* (virtual memory) megvalósításához. Ennek köszönhetõen a számítógép képes úgy viselkedni, mintha jóval több memóriával rendelkezne, mint valójában. Így, amikor a FreeBSD kifogy a memóriából, egyszerûen kirakja a memóriából a lapozóállományba az éppen nem használt adatokat, majd amikor ismét szüksége lesz rájuk, visszatölti ezeket (és ilyenkor megint kirak valami mást).

Némely partícióhoz kötõdnek bizonyos megszokások.

Partíció	Megszokás
a	Általában ez tartalmazza a gyökér-állományrendszert.
b	Általában ez tartalmazza a lapozóállományt.
c	Mérete általában a tartalmazó slice méretével egyezik meg. Ennek köszönhetõen a segédprogramok (például egy hibás szektorokat keresõ program) a c partíción keresztül képesek akár az egész slice-szal dolgozni. Normális esetben ezen a partíción nem hozunk létre állományrendszert.
d	A d partícióhoz egykoron kapcsolódott különleges jelentés, azonban mostanra ez már megszûnt, és a d egy teljesen átlagos partíciónak tekinthetõ.

Minden állományrendszert tartalmazó partíciót a FreeBSD egy ún. *slice*-ban tárol. A FreeBSD számára a slice elnevezés utal mindarra, amit általában partíciónak neveznek, és ismét megemlítjük, mindez a UNIX®-os eredet miatt. A slice-okat 1-tõl 4-ig sorszámozzák.

A slice-ok sorszáma 1-tõl indulva az eszközök neve után egy s betûvel elválasztva következik. Így tehát a "da0*s1*" jelentése az elsõ slice lesz az elsõ SCSI-meghajtón. Lemezenként négy fizikai slice hozható létre, de ezeken belül tetszõleges típusú logikai slice-ok helyezhetõek el. Ezen további slice-ok sorszámozása 5-tõl kezdõdik, így ennek megfelelõen a "ad0*s5*" lesz az elsõ IDE-lemezen található elsõ kiterjesztett slice. Ezeket az eszközöket foglalják el a különbözõ állományrendszerek.

A slice-ok, a "veszélyesen dedikált" (Dangerously Dedicated) fizikai meghajtók, és minden más olyan meghajtó, amely *partíciókat* tartalmaz, a-tól h-ig jelölõdnek. Ez a betû az eszköz neve után következik, így ennek megfelelõen a "da0*a*" lesz az elsõ "da" meghajtó "a", vagyis a "veszélyesen dedikált" partíciója. Az "ad1s3*e*" lesz a második IDE-lemezmeghajtón a harmadik slice-ban szereplõ ötödik partíció.

Végezetül, a rendszerben minden lemezt azonosítunk. A lemez neve a típusára utaló kóddal kezdõdik, amely után aztán egy sorszám jelzi, hogy melyik lemezrõl is van szó. Azonban eltérõen a slice-okétól, a lemezek sorszámozása 0-tól indul. Az általánosan elterjedt kódolások a Lemezes eszközök kódjaiban találhatóak.

Amikor hivatkozunk egy partícióra, a FreeBSD elvárja tõlünk, hogy nevezzük meg az adott partíciót tartalmazó slice-ot és lemezt is. Emiatt egy partícióra mindig úgy hivatkozunk, hogy elõször megadjuk a tartalmazó lemez nevét, ettõl s-sel elválasztva a tartalmazó slice sorszámát, majd ezt a partíció betûjelével zárjuk. Erre példákat a Példák lemezek, slice-ok és partíciók neveireban láthatunk.

Az érhetõség kedvéért a Egy lemez kialakításának sablonja bemutatja egy lemez kiosztásának fogalmi sablonját.

A FreeBSD telepítéséhez elõször be kell állítani a lemezen található slice-okat, majd létrehozni

benne a FreeBSD-hez használni kívánt partíciókat, kialakítani rajtuk az állományrendszereket (vagy a lapozóállományt) és eldönteni, melyik állományrendszert kívánjuk csatlakoztatni.

Táblázat 5. Lemezes eszközök kódjai

Kód	Jelentés
ad	ATAPI (IDE) lemez
da	közvetlen hozzáférésû SCSI lemez
acd	ATAPI (IDE) CDROM
cd	SCSI CDROM
fd	Floppylemez

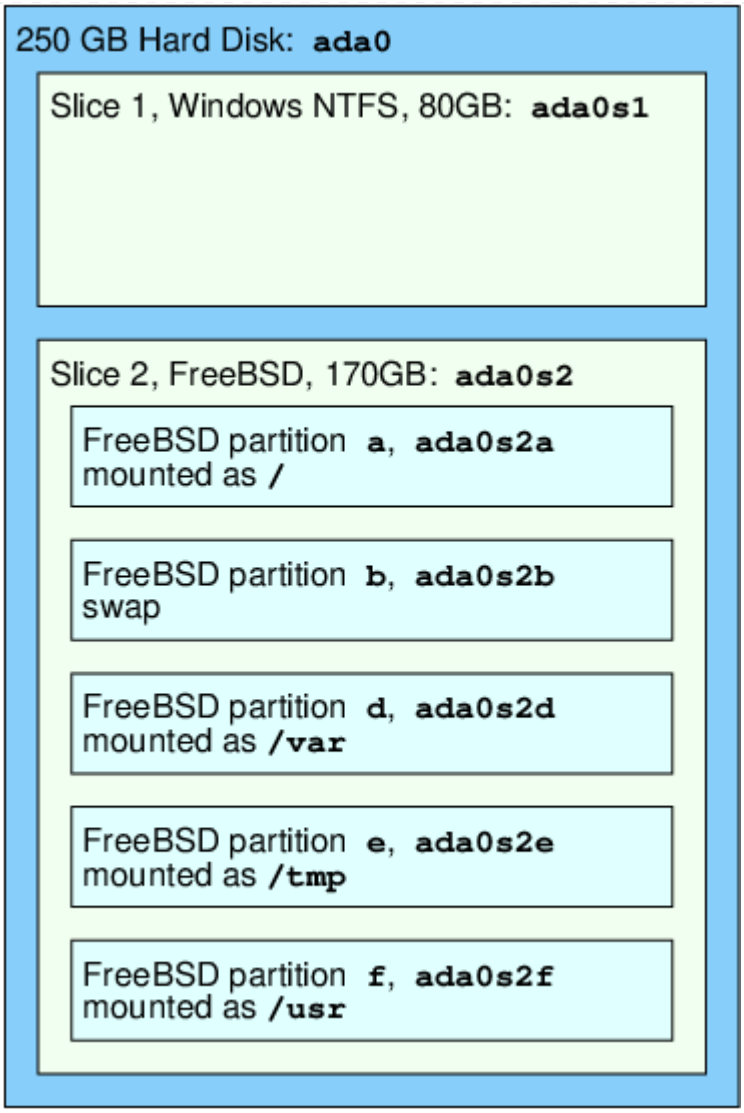
Példa 3. Példák lemezek, slice-ok és partíciók neveire

Név	Jelentés
`ad0s1a`	Az elsõ IDE lemezen (`ad0`) levõ elsõ slice (`s1`) elsõ partíciója (`a`).
`da1s2e`	A második SCSI-lemzen (`da1`) levõ második slice (`s2`) ötödik partíciója (`e`).

Példa 4. Egy lemez kialakításának sablonja

Az ábrán a rendszerhez csatlakoztatott elsõ IDE-lemez látható a FreeBSD szemszögébõl. Tegyük fel, hogy ez a lemez 4 GB méretû és két, egyenként 2 GB méretû slice-ot (avagy MS-DOS® partíciót) tartalmaz. Az elsõ slice egy MS-DOS® formátumú lemezt foglal magában, a C: meghajtót, illetve a második slice egy telepített FreeBSD-t tartalmaz. Ebben a példában a FreeBSD három adatot és egy lapozóállományt tároló partícióval rendelkezik.

A három partíció mindegyikén találhatunk egy-egy állományrendszert. Az `a` partíció lesz a gyökér-állományrendszer, az `e` lesz a rendszerünkben a /var és az `f` pedig a /usr könyvtár.



3.6. Állományrendszerek csatlakoztatása és leválasztása

Az állományrendszereket legkönnyebben egy-egy faként tudjuk magunk elõtt elképzelni, amelyek a / könyvtárból nõnek ki. A /dev, /usr és mellettük szereplõ, hozzájuk hasonló összes többi könyvtár csupán egy-egy ág, amelyeknek saját ágaik is lehetnek, mint például a /usr/local és így tovább.

Különféle okai vannak annak, hogy egyes könyvtárakat különálló állományrendszereken tárolunk. A /var könyvtár tartalmazza a log/, spool/ könyvtárakat és különféle átmeneti állományokat, azonban az ilyen állományok könnyen megszaporodhatnak és megtölthetik az állományrendszert. Mivel a gyökér-állományrendszert nem tanácsos elárasztani mindenféle állománnyal, ezért gyakran a hasznunkra válhat, ha a /var könyvtárat leválasztjuk a / könyvtárból.

A másik gyakori ok, ami az imént említett fa egyes ágainak különbözõ állományrendszereken történõ tárolását indokolja, hogy ezek gyakran más fizikai vagy virtuális lemezeken, például a rendszerhez csatlakoztatott Hálózati állományrendszereken vagy éppen CD-meghajtókon találhatóak.

3.6.1. Az fstab állomány

A rendszerindítás folyamata során az /etc/fstab állományban felsorolt állományrendszerek maguktól kerülnek csatlakoztatásra (kivéve amikor a `noauto` beállítással szerepelnek).

Az /etc/fstab állományban található sorok az alábbi szerkezetûek:

```
eszköz       /csatlakozási-pont típus     beállítások      mentésigyak     ellszám
```

`eszköz`

A Az eszközök elnevezéseiben leírtak szerint megnevezett (létezõ) eszköz.

`csatlakozási-pont`

Egy (létezõ) könyvtár, ahova az állományrendszer csatlakozik.

`típus`

Az állományrendszer mount(8) parancs szerint ismert típusa. A FreeBSD alapértelmezett állományrendszere az `ufs`.

`beállítások`

Az írható-olvasható állományrendszerek esetén `rw`, az írásvédettek esetén pedig `ro`, amelyet igény szerint további beállítások követhetnek. A rendszerindítás során automatikusan nem csatlakoztatandó állományrendszerek esetén gyakran alkalmazott beállítás itt még a `noauto`. Egyéb lehetõségeket a mount(8) man oldalon láthatunk.

`mentésigyak`

Ezt általában a dump(8) parancs használja a menteni szükséges állományrendszerek megállapításához. Amennyiben hiányzik ez a mezõ, az automatikusan a nulla értéket jelöli.

`ellszám`

Megadja, hogy mely állományrendszereket kell ellenõrizni. A nullás `pass` értékkel rendelkezõ állományrendszerek nem kerülnek ellenõrzésre. A gyökér-állományrendszer (melyet minden más elõtt kell ellenõrizni) `passno` értéke egy, míg az összes többi állományrendszer `passno` értéke általában egytõl különbözõ. Ha egynél több állományrendszer is ugyanazt a `passno` értéket kapta, akkor az fsck(8) a lehetõségei szerint megpróbálja ezeket egyszerre ellenõrizni.

Az /etc/fstab felépítésérõl és a benne használható beállításokról bõvebben a fstab(5) man oldalon olvashatunk.

3.6.2. A `mount` parancs

Az állományrendszerek tényleges csatlakoztatására avagy "mountolására" a mount(8) parancs használható.

Legegyszerûbb formája:

```
# mount eszköz csatlakozási-pont
```

Ahogy a mount(8) man oldalán is olvashatjuk, itt rengeteg opció is megadható, de ezek közül a leggyakoribbak:

Csatlakoztatási opciók

`-a`

Csatlakoztatja az /etc/fstab állományban felsorolt összes állományrendszert, kivéve azokat, amelyek a "noauto" beállítást tartalmazzák, vagy kizártuk a `-t` kapcsolóval, esetleg korábban már csatlakoztattuk.

`-d`

A tényleges csatlakoztatás elvégzése nélkül végrehajt minden mást. Ez az opció leginkább `-v` opcióval együtt használható annak megállapítására, hogy a mount(8) valójában mit is akar csinálni.

`-f`

Egy nem tiszta állományrendszer csatlakoztatásának kényszerítése (veszélyes!) vagy egy korábban már csatlakoztatott állományrendszer írható állapotának felfüggesztése.

`-r`

Az állományrendszer írásvédett csatlakoztatása. Megegyezik a `-o` opciónál megadható `ro` (vagy a FreeBSD 5.2-nél régebbi verziója esetén a `rdonly`) beállítás használatával.

`-t` ***típus***

Az adott állományrendszert az adott típusnak megfelelõen csatlakoztatja, vagy az `-a` használata esetén csak az adott típusú állományrendszereket.

Az "ufs" az állományrendszerek alapértelmezett típusa.

`-u`

Frissíti az állományrendszerre vonatkozó csatlakoztatási beállításokat.

`-v`

Részletesebb kijelzés.

`-w`

Az állományrendszer csatlakoztatása írásra és olvasásra.

Az `-o` opció után vesszõvel elválasztott beállításokat adhatunk meg, többek közt az alábbiakat:

noexec

Az állományrendszeren található állományok végrehajtásának tiltása. Ez egy nagyon hasznos biztonsági beállítás.

nosuid

Az állományrendszeren nem használhatóak a felhasználó- (setuid) vagy csoportváltásra (setgid) vonatkozó engedélyek. Nagyon hasznos biztonsági beállítás.

3.6.3. Az `umount` parancs

Az umount(8) parancs paraméterként egy csatlakozási pontot, egy eszköznevet vagy a `-a`, illetve az `-A` opciókat várja.

A leválasztás kényszerítéséhez mindegyik alakban szerepelhet az `-f` opció, valamint a részletesebb kijelzést a `-v` opcióval kapcsolhatjuk be. Azonban szeretnénk mindenkit figyelmeztetni, hogy a `-f` használata alapvetõen nem ajánlott. Az erõszakkal leválasztott állományrendszerek összeomlaszthatják a számítógépet vagy kárt okozhatnak az állományrendszereken található adatokban.

Az `-a` és `-A` opciók használatosak az összes csatlakoztatott állományrendszer leválasztására, amelyek típusait a `-t` opció megadása után sorolhatjuk fel. Fontos különbség azonban, hogy az `-A` opció a gyökér állományrendszert nem próbálja meg leválasztani.

3.7. Folyamatok

A FreeBSD egy többfeladatos operációs rendszer. Ez azt jelenti, hogy képes látszólag egyszerre több programot is futtatni. Az így egyszerre futó programokat egyenként *folyamat*oknak (process) nevezzük. Minden kiadott parancsunk elindít legalább egy ilyen folyamatot, és a rendszerünk mozgásban tartásához bizonyos rendszerszintû folyamatok állandóan futnak a háttérben.

Minden folyamatot egy *folyamatazonosító*nak (process ID vagy *PID*) nevezett szám azonosít egyértelmûen, és az állományokhoz hasonlóan, minden folyamatnak van tulajdonosa és csoportja is. A tulajdonos és a csoport ismeretében állapítja meg a rendszer, hogy az adott folyamat a korábban említett engedélyek szerint milyen állományokhoz és eszközökhöz férhet hozzá. Ezenkívül a legtöbb folyamatnak van még egy szülõfolyamata is. A szülõfolyamat az a folyamat, amely az adott folyamatot elindította. Például amikor parancsokat adunk egy parancsértelmezõn keresztül, akkor maga a parancsértelmezõ is egy ilyen folyamat lesz ugyanúgy, ahogy a benne kiadott parancsok által elindított programok. Ennek megfelelõen az így létrehozott összes folyamat szülõje maga a parancsértelmezõ folyamata lesz. Az említettek alól egyik kivétel az init(8) nevû speciális folyamat. Az `init` lesz a rendszerben mindig az elsõ folyamat, ezért a PID-je is mindig 1. Az `init` programot a FreeBSD indulásakor a rendszermag fogja automatikusan elindítani.

A rendszerben futó programok vizsgálatához két, különösen hasznos parancsot találhatunk: ezek a ps(1) és a top(1). A `ps` parancs használatos a pillanatnyilag futó programok statikus listájának megjelenítésére. Ebben olvashatjuk a futó programok azonosítóit, mennyi memóriát használnak éppen, milyen paranccsal indították ezeket stb. A `top` parancs mutatja az összes aktívan futó programot, majd néhány másodpercenként automatikusan frissíti ezt a listát, aminek révén folyamatosan láthatjuk, miként viselkednek a futó programok.

A `ps` alapértelmezés szerint csupán az általunk futtatott programokat mutatja. Például:

```
% ps
  PID  TT  STAT      TIME COMMAND
  298  p0  Ss     0:01.10 tcsh
 7078  p0  S      2:40.88 xemacs mdoc.xsl (xemacs-21.1.14)
37393  p0  I      0:03.11 xemacs freebsd.dsl (xemacs-21.1.14)
48630  p0  S      2:50.89 /usr/local/lib/netscape-linux/navigator-linux-4.77.bi
```

```
48730  p0  IW     0:00.00 (dns helper) (navigator-linux-)
72210  p0  R+     0:00.00 ps
  390  p1  Is     0:01.14 tcsh
 7059  p2  Is+    1:36.18 /usr/local/bin/mutt -y
 6688  p3  IWs    0:00.00 tcsh
10735  p4  IWs    0:00.00 tcsh
20256  p5  IWs    0:00.00 tcsh
  262  v0  IWs    0:00.00 -tcsh (tcsh)
  270  v0  IW+    0:00.00 /bin/sh /usr/X11R6/bin/startx -- -bpp 16
  280  v0  IW+    0:00.00 xinit /home/nik/.xinitrc -- -bpp 16
  284  v0  IW     0:00.00 /bin/sh /home/nik/.xinitrc
  285  v0  S      0:38.45 /usr/X11R6/bin/sawfish
```

Ahogy az a fenti példában is látszik, a ps(1) kimenete oszlopokra tagolható. Ezek közül a PID tartalmazza a korábban már ismertetett folyamatazonosítókat. Az azonosítók 1-tõl indulva egészen 99999-ig sorszámozódhatnak, illetve ha kifutnánk belõlük, akkor a számozás kezdõdik elölrõl (azonban a használatban levõ azonosítók sosem kerülnek újra kiosztásra). A TT oszlopban láthatjuk azt a terminált, amelyen az adott program éppen fut, de ezt pillanatnyilag akár nyugodtan figyelmen kívül is hagyhatjuk. A STAT oszlopban a program állapotát kapjuk meg, de szintén átugorható. A TIME a program processzoron eltöltött idejét mutatja - ez általában nem arra utal, hogy mennyi ideje fut maga a program, hiszen a legtöbb program sok idõt tölt tétlenül, mielõtt egyáltalán szüksége lenne processzora. Végezetül a COMMAND oszlopban olvashatjuk azt a parancsot, amellyel a programot elindították.

A ps(1) számos különféle beállítást ismer az általa megjelenített információk megválasztásához. Az egyik ilyen leghasznosabb beállítás az auxww: az a segítségével az összes futó programot láthatjuk, nem csak a sajátjainkat; az u megadásával láthatóvá válik a folyamat tulajdonosának a felhasználói neve, valamint a memóriahasználata is; az x megmutatja a démon (avagy háttér)folyamatok adatait is és a ww hatására pedig a ps(1) az összes folyamathoz a teljes parancssort kiírja, még akkor is, ha nem férne ki a képernyõre.

A top(1) kimenete is hasonló. Ha elindítjuk, általában ezt láthatjuk:

```
% top
last pid: 72257;  load averages:  0.13,  0.09,  0.03    up 0+13:38:33  22:39:10
47 processes:  1 running, 46 sleeping
CPU states: 12.6% user,  0.0% nice,  7.8% system,  0.0% interrupt, 79.7% idle
Mem: 36M Active, 5256K Inact, 13M Wired, 6312K Cache, 15M Buf, 408K Free
Swap: 256M Total, 38M Used, 217M Free, 15% Inuse

  PID USERNAME PRI NICE  SIZE    RES STATE    TIME   WCPU    CPU COMMAND
72257 nik       28   0  1960K  1044K RUN      0:00 14.86%  1.42% top
 7078 nik        2   0 15280K 10960K select   2:54  0.88%  0.88% xemacs-21.1.14
  281 nik        2   0 18636K  7112K select   5:36  0.73%  0.73% XF86_SVGA
  296 nik        2   0  3240K  1644K select   0:12  0.05%  0.05% xterm
48630 nik        2   0 29816K  9148K select   3:18  0.00%  0.00% navigator-linu
  175 root       2   0   924K   252K select   1:41  0.00%  0.00% syslogd
 7059 nik        2   0  7260K  4644K poll     1:38  0.00%  0.00% mutt
```

```
...
```

A kimenet két részre osztható. A fejlécben (vagyis az elsõ öt sorban) látható az utoljára futtatott program azonosítója (PID), a rendszer átlagos terhelése (load average, amellyel mérjük, hogy a rendszerünk mennyire lefoglalt), a rendszer indítása óta eltelt idõ (up mint uptime) és a jelenlegi idõ. A fejlécben még megtalálhatjuk azt is, mennyi program fut (esetünkben ez most 47), mennyi memóriát és lapozóállományt használnak, és mennyi idõt tölt a rendszer a processzor különbözõ állapotaiban.

A fejléc alatt a ps(1) kimenetéhez hasonló módon oszlopokba rendezve találhatjuk meg a folyamatok adatait: az azonosítóikat, a tulajdonosaik nevét, a felhasznált processzoridõt, a futtatott parancsot. A top(1) alapértelmezés szerint mutatja a futó programok által használt memória mennyiségét is: ez további két oszlopra oszlik, ahol az egyikben a teljes memóriafoglalást (SIZE), a másikban pedig a jelen pillanatban aktívan használt memóriát (RES) láthatjuk. A példában látható is, hogy a getenv(3) (navigator-linu) alkalmazásnak majdnem 30 MB-nyi memóriára van szüksége, de ebbõl aktívan csak 9 MB-ot használ.

A top(1) a kijelzést minden második másodpercben magától frissíti, de ez az `s` kapcsolóval állítható.

3.8. Démonok, jelzések és a futó programok leállítása

Amikor elindítunk egy szövegszerkesztõt, nem sok gondunk akad az irányításával, könnyen utasíthatjuk az állományok betöltésére és így tovább. Mindezt azért tehetjük meg, mert a szövegszerkesztõ erre lehetõséget biztosít és mivel a szövegszerkesztõ egy *terminál*hoz kapcsolódik. Egyes programok azonban nem úgy lettek kialakítva, hogy állandóan a felhasználó utasításaira támaszkodjanak, ezért az elsõ adandó alkalommal lekapcsolódnak a terminálról. Például egy webszerver egész nap csak webes kéréseket válaszol meg, és általában semmi szüksége nincs a felhasználók utasításaira. A szerverek között leveleket közvetítõ programok is ugyanezen osztályba tartoznak.

Ezeket a programokat *démon*onoknak hívjuk. A démonok a görög mitológiában jelentek meg: sem a jót, sem pedig a gonoszt nem képviselték, egyszerû apró szellemecskék voltak, akik az emberiség javát szolgálták, pontosan úgy, ahogy ma teszik azt a különféle web- és levelezõ szerverek. Ezért is ábrázolták sokáig a BSD kabalafiguráját is egy tornacipõs, vasvillás vidám démonként.

A démonként futó programok nevéhez a hagyományok szerint hozzá szokták fûzni a "d" betût. A BIND a Berkeley Internet Name Domain (névfeloldó) szolgáltatása, azonban a hozzá tartozó program neve `named`, az Apache webszerver programját `httpd`-nek nevezik, a sornyomtató kezeléséért felelõs démon pedig az `lpd` és így tovább. Ez csupán egy hagyomány, megszokás, nem pedig egy kõbe vésett szabály: például a Sendmail levelezõ démonának neve `sendmail` és nem pedig `maild`.

Néha azért szükségünk lehet arra, hogy felvegyük valahogy a kapcsolatot a démonként futó programokkal is. Ennek egyik lehetséges módja a *jelzés*ek (signal) küldése (de alapvetõen bármilyen futó programnak küldhetünk). Több különféle jelzés küldhetõ - egyeseknek közülük megkülönböztetett jelentése van, másokat magukat az alkalmazások értelmeznek, amelyrõl a dokumentációjukban tájékozódhatunk. A kill(1) vagy kill(2) paranccsal más tulajdonában levõ futó programoknak nem tudunk jelzéseket küldeni, ami alól egyedüli kivétel a `root` felhasználó.

Bizonyos esetekben a FreeBSD maga is küld néha jelzéseket. Amikor egy alkalmazást rosszul programoznak le és megpróbál egy számára tiltott memóriaterülethez hozzáférni, a FreeBSD küld neki egy *Segmentation Violation* (SIGSEGV, szegmentálási hiba) jelzést. Ha egy alkalmazás az alarm(3) rendszerhíváson keresztül kér egy adott idõ utáni bekövetkezõ értesítést, akkor kap errõl egy Alarm (SIGALRM) jelzést és így tovább.

A folyamatok leállítására két jelzés használható: a SIGTERM (befejeztetés) és a SIGKILL (leállítás). A SIGTERM a folyamatok leállításának illedelmes módja, mivel ekkor a futó program képes *elkapni* ezt a jelzést és észrevenni, hogy le akarjuk állítani. Ilyenkor a leállítás elõtt lehetõsége van szabályosan lezárni a naplóit és általánosságban véve befejezni mindent, amit éppen csinál. Elõfordulhat azonban, hogy a folyamatok figyelmen kívül hagyják a SIGTERM jelzést, ha például éppen egy félbeszakíthatatlan feladat közepén tartanak.

A SIGKILL jelzést azonban egyetlen futó program sem hagyhatja figyelmen kívül. Ez lenne a "Nem érdekel, mivel foglalkozol, azonnal hagyd abba!" jelzés. Amikor SIGKILL jelzést küldünk egy folyamatnak, a FreeBSD leállítja a folyamatot ott és ahol tart .

További használható jelzések: SIGHUP, SIGUSR1 és SIGUSR2. Ezek általános célú jelzések, amelyeket az alkalmazások eltérõ módokon kezelnek.

Tegyük fel, hogy megváltoztattuk a webszerverünk beállításait tartalmazó állományt - valamilyen módon szeretnénk tudatni a szerverrel, hogy olvassa be újra a beállításait. Ezt megtehetjük úgy, hogy leállítjuk és újraindítjuk a httpd démont, de ezzel kiesést okozhatunk a szerver mûködésében, amit viszont nem engedhetünk meg. A legtöbb démont úgy készítették el, hogy a SIGHUP jelzés hatására olvassa be újra a beállításait tartalmazó állományt. Így a httpd leállítása és újraindítása helyett egyszerûen elegendõ egy SIGHUP jelzés küldése. Mivel azonban ez nem szabványosított, a különbözõ démonok ezt a jelzést többféleképpen is értelmezhetik. Ezért a használata elõtt ennek mindenképpen járjunk utána a kérdéses démon dokumentációjában.

A jelzéseket a kill(1) paranccsal tudjuk elküldeni, ahogy ezt a következõ példában is láthatjuk.

Procedure: Jelzés küldése egy futó programnak

Ebben a példában megmutatjuk, hogyan lehet jelzést küldeni az inetd(8) démonnak. Az inetd a beállításait az /etc/inetd.conf állományban tárolja, és az inetd a SIGHUP jelzés hatására képes újraolvasni ezt.

1. Keressük meg annak a folyamatnak az azonosítóját, amelynek a jelzést kívánjuk küldeni. Ezt a ps(1) és a grep(1) használatával tehetjük meg. A grep(1) parancs segítségével más parancsok kimenetében tudunk megkeresni egy általunk megadott szöveget. Ezt a parancsot átlagos felhasználóként futtatjuk, azonban az inetd(8) démont a root birtokolja, ezért az ps(1) használata során meg kell adnunk az ax kapcsolókat is.

   ```
   % ps -ax | grep inetd
     198  ??  IWs    0:00.00 inetd -wW
   ```

   Innen kiderül, hogy az inetd(8) azonosítója 198. Elõfordulhat, hogy az eredményben maga a grep inetd parancs is megjelenik. Ez a ps(1) listázási módszere miatt következhet be.

2. A jelzés elküldésére használjuk a kill(1) parancsot. Mivel az inetd(8) démont a `root` felhasználó futtatja, ehhez elõször a su(1) parancs kiadásával nekünk is `root` felhasználóvá (rendszeradminisztrátorrá) kell válnunk.

```
% su
Password:
# /bin/kill -s HUP 198
```

Ahogy az a legtöbb UNIX® esetén elfogadott, a sikeres végrehajtás esetén a kill(1) sem válaszol semmit. Amikor viszont nem egy saját programunknak akarunk jelzést küldeni, akkor a `kill: PID: Operation not permitted` (a mûvelet nem engedélyezett) hibaüzenetet látunk. Ha véletlenül elgépeltük volna a futó program azonosítóját, akkor a küldendõ jelzés nem a megfelelõ folyamatnál fog kikötni (ami nem éppen jó), vagy ha szerencsénk van, akkor a jelzést egy éppen használaton kívüli azonosítóra küldtük. Az utóbbi esetben a következõ láthatjuk: `kill: PID: No such process` (nincs ilyen folyamat).

Miért `/bin/kill`?

A legtöbb parancsértelmezõ beépítetten tartalmazza a saját `kill` parancsát, tehát ilyenkor közvetlenül maga a parancsértelmezõ küldi a jelzést, nem pedig a /bin/kill programon keresztül. Ez gyakran a javunkra válhat, azonban a küldhetõ jelzések megadása parancsértelmezõnként eltérhet. Így, ahelyett, hogy egyenként ismernünk kellene mindegyiket, sokkal egyszerûbb közvetlenül a `/bin/kill …` parancsot használni.

A többi jelzés küldése is nagyon hasonló módon történik, hiszen elegendõ csupán a `TERM` vagy a `KILL` behelyettesítése a parancs megfelelõ helyére.

A rendszerünkben óvatosan bánjunk a futó programok leállítgatásával, és legyünk különös tekintettel az 1-es azonosítóval rendelkezõ, speciális feladattal bíró init(8) folyamatra. A `/bin/kill -s KILL 1` parancs kiadásával ugyanis gyorsan le tudjuk állítani a rendszerünket. *Mielõtt* egy kill(1) parancsot lezárnánk az `Enter` billentyûvel, *mindig* gyõzõdjünk meg róla, hogy valóban tényleg a jó paramétereket adtuk meg.

3.9. Parancsértelmezõk

A FreeBSD-ben hétköznapi munkánk legnagyobb részét a parancsértelmezõknek (shell) nevezett parancssoros felületen tudjuk elvégezni. A parancsértelmezõ fõ feladata a beérkezõ parancsok elfogadása és végrehajtatása. Sok parancsértelmezõ ezenfelül rendelkezik beépített funkciókkal is, amelyek olyan hétköznapi feladatokban igyekeznek segíteni, mint például az állományok kezelése és tömeges elérése reguláris kifejezések használatával, a parancssor szerkesztése, parancsok makrózása és a környezeti változók használata. A FreeBSD alapból tartalmaz néhány parancsértelmezõt, ilyen például az `sh`, a Bourne Shell, és a `tcsh`, a továbbfejlesztett C-shell. Sok más parancsértelmezõ, mint például a `zsh` és `bash` is elérhetõ a FreeBSD Portgyûjteményébõl.

De melyik parancsértelmezõt is válasszuk? Ez igazából ízlés kérdése. Ha inkább C programozók

vagyunk, akkor valószínûleg egy olyan C-szerû shelllel tudunk kényelmesen dolgozni, amilyen például a `tcsh`. Ha viszont egy linuxos rendszert használtunk korábban vagy éppen még soha nem használtunk volna a UNIX® parancssorát, érdemes a `bash`-sel megpróbálkoznunk. A lényeg az, hogy minden parancsértelmezõnek vannak olyan egyedi jellemezõi, amiért használatóak vagy éppen nem használatóak a munkánkban, ezért magunknak kell kiválasztani a nekünk megfelelõt.

A shellek egyik legáltalánosabb jellemzõje az állományok neveinek kiegészítése. Miután begépeljük egy parancs vagy állománynév elsõ néhány karakterét, a `Tab` billentyû lenyomásával megkérhetjük a parancsértelmezõt, hogy magától egészítse ki ("találja ki") a fennmaradó részt. Nézzük erre egy példát. Tegyük fel, hogy van két állományunk, izemize és ize.mize, és szeretnénk letörölni az ize.mize nevût. Ehhez a következõt kell begépelnünk: `rm iz[Tab].[Tab]`.

Erre a parancsértelmezõ a következõ parancsot írja ki: `rm ize[SIPOLÁS].mize`.

A [SIPOLÁS] itt a konzol sípjára vonatkozik, amellyel jelzi, hogy nem tudta teljesen kiegészíteni az állomány nevét, mivel egynél több is megfelel a megadott alaknak. Az izemize és az ize.mize is egyaránt az `iz` elõtaggal kezdõdik, azonban ebbõl a parancsértelmezõ csak az `ize` elõtagot tudta kikövetkeztetni. Ha most begépelünk még egy `.` karaktert és újra megnyomjuk a `Tab` billentyût, a parancsértelmezõ ezúttal képes lesz az állomány teljes nevét megállapítani.

A parancsértelmezõk másik általános jellemzõje a környezeti változók használata. A környezeti változók lényegében a parancsértelmezõ környezetéhez tárolt név-érték párok. Ezt a környezetet látja minden olyan program, amit a parancsértelmezõbõl meghívunk, és ezért tartalmazni is szokott sok ilyen beállítást. Íme a leggyakoribb környezeti változók felsorolása és rövid leírása:

Változó	Leírás
`USER`	A bejelentkezett felhasználó neve.
`PATH`	Vesszõvel elválasztott könyvtárak, ahol a parancsértelmezõ a végrehajtható állományokat keresi.
`DISPLAY`	Az aktuálisan használt X11 megjelenítõ hálózati neve, amennyiben létezik ilyen.
`SHELL`	A használt parancsértelmezõ.
`TERM`	A felhasználó által használt terminál típusa. Ebbõl a terminál képességeit lehet megállapítani.
`TERMCAP`	A terminálok adatbázisából származó, különbözõ terminálfunkciókhoz tartozó helyettesítõ (escape) kódok.
`OSTYPE`	Az operációs rendszer típusa, például FreeBSD.
`MACHTYPE`	A rendszer alatt futó gép architektúrája.
`EDITOR`	A felhasználó által használt szövegszerkesztõ.
`PAGER`	A felhasználó által lapozásra használt program.

Változó	Leírás
MANPATH	Vesszõvel elválasztott könyvtárak, ahol a parancsértelmezõ a man oldalakat keresi.

A környezeti változók beállítása parancsértelmezõnként valamennyire eltér. Például egy C stílusú parancsértelmezõ, mint például a tcsh vagy a csh, a setenv paranccsal állítja a környezeti változókat. A Bourne-féle parancsértelmezõk, mint például az sh vagy a bash, az export parancsot használják a környezeti változók beállítására. Például a csh vagy a tcsh használata során a következõképpen tudjuk be- vagy átállítani az EDITOR környezeti változó értékét /usr/local/bin/emacs -re:

```
% setenv EDITOR /usr/local/bin/emacs
```

Ugyanez a Bourne-féle parancsértelmezõkben:

```
% export EDITOR="/usr/local/bin/emacs"
```

A legtöbb parancsértelmezõben a nevük elõtt szerepeltetett $ jel segítségével kérhetjük a környezeti változók értékének behelyettesítését a parancssorba. Ennek megfelelõen az echo $TERM parancs kiíratja a TERM változó aktuális értékét, mivel ebbe a parancsértelmezõ már az echo meghívása elõtt behelyettesíti a TERM értékét.

A parancsértelmezõk számos speciális karaktert, ún. metakaraktert az adatok különleges reprezentációjaként kezelnek. Köztük a leggyakrabban használt a *, amely tetszõleges számú karaktert helyettesít egy állomány nevében. Az ilyen metakarakterek segítségével tudunk egyszerre több állományt is megnevezni. Például ha begépeljük az echo * parancsot, akkor majdnem ugyanazt kapjuk eredményül, mintha az ls parancsot adtuk volna ki, hiszen a parancsértelmezõ ilyenkor veszi az összes * metakarakterre illeszkedõ állományt, és a kiíratásukhoz pedig rendre behelyettesíti ezeket a parancssorba az echo paramétereként.

Ha nem szeretnénk, hogy a parancsértelmezõ értelmezze a speciális karaktereket, akkor egy "backslash" (visszaper) (\) karaktert eléjük téve mindezt megakadályozhatjuk. Az echo $TERM parancs ugyebár kiíratja a terminálra vonatkozó környezeti változó beállítását, azonban a echo \$TERM változatlanul kiírja a $TERM szöveget.

3.9.1. A parancsértelmezõnk megváltoztatása

A parancsértelmezõnk legegyszerûbben a chsh parancs használatával változtatható meg. A chsh kiadása után elindítja az EDITOR környezeti változónak megfelelõ szövegszerkesztõt, ha nem lenne ilyen, akkor alapértelmezés szerint a vi hívódik meg. Az így megnyitott állományban változtassuk meg kedvünk szerint a "Shell: " kezdetû sort.

A chsh parancsnak megadhatjuk az -s opciót is, amin keresztül szövegszerkesztõ használata nélkül be tudjuk állítani a parancsértelmezõt. Például ha a parancsértelmezõnket a bash-re akarjuk lecserélni, akkor ezt írjuk be:

```
% chsh -s /usr/local/bin/bash
```

A használni kívánt parancsértelmezõnek szerepelnie *kell* az /etc/shells állományban. Ha a kiválasztott parancsértelmezõt a Portgyûjteménybõl telepítettük fel, akkor az már minden bizonnyal bekerült oda. Ha viszont saját magunk raktuk volna fel, akkor ide is fel kell vennünk.

Például ha a bash-t manuálisan telepítettük és másoltuk a /usr/local/bin könyvtárba, akkor így kell eljárnunk:

```
# echo "/usr/local/bin/bash" >> /etc/shells
```

Majd próbálkozzunk újra a chsh paranccsal.

3.10. Szövegszerkesztõk

A FreeBSD beállításának nagy része szöveges állományok szerkesztésével történik. Emiatt sosem árt legalább egy szövegszerkesztõt ismernünk. A FreeBSD alaprendszerében, valamint a Portgyûjteményben is találhatunk néhányat belõlük.

A legegyszerûbben megtanulható és legkönnyedebb szövegszerkesztõt ee-nek, avagy "easy editornak" hívják. Az ee indításához írjuk be az `ee állománynév` parancsot, ahol az *állománynév* lesz a szerkesztendõ állomány neve. Így például az /etc/rc.conf állomány szerkesztéséhez gépeljük be az `ee /etc/rc.conf` parancsot. Miután elindult az ee, az összes szerkesztéshez használható parancsa megjelenik a képernyõ felsõ részében. Itt a "kalap" (^) karakter a `Ctrl` billentyû lenyomására utal, így tehát a ^e jelölés a `Ctrl` + `e` billentyûkombinációt jelenti. Ha ki akarunk lépni az ee-bõl, nyomjuk le az `Esc` billentyût, majd a felbukkanó menübõl válasszuk a szerkesztõ elhagyását (leave editor). Ha az állományt módosítottuk, kilépés elõtt még a szövegszerkesztõ rákérdez, hogy mentse-e a változtatásainkat.

A FreeBSD nagyobb tudású szövegszerkesztõket, mint például a vi-t, is tartalmaz az alaprendszer részeként, miközben a többi, mint például az Emacs vagy a vim a Portgyûjtemény részeként (editors/emacs és editors/vim) érhetõ el. Ezek a szerkesztõk sokkal több lehetõséget és erõt képviselnek, amiért cserébe viszont valamivel nehezebb megtanulni a használatukat. Ha viszont rengeteg szöveget akarunk majd szerkeszteni, akkor egy vim vagy Emacs használatának megismerésével sok idõt megspórolhatunk.

Számos alkalmazás, amely állományokat akar módosítani vagy szöveges bemenetre van szüksége, automatikusan szövegszerkesztõt nyit meg. Ezt az EDITOR környezeti változó beállításával tudjuk meghatározni. Errõl részletesebben a parancsértelmezõknél olvashatunk.

3.11. Eszközök és eszközleírók

Az eszköz elnevezést leginkább a rendszerben folyó, hardverrel kapcsolatos tevékenységek kapcsán használják lemezekre, nyomtatókra, grafikus kártyákra és billentyûzetekre. A FreeBSD indulása

során többnyire azt láthatjuk, hogy milyen eszközöket sikerült felismernie. Ezeket a rendszerindításkor megjelenõ üzeneteket a /var/run/dmesg.boot állományban nézhetjük meg újra.

Például az acd0 az elsõ IDE CD-meghajtót, míg a kbd0 a billentyûzetet képviseli.

A UNIX® operációs rendszerben a legtöbb eszközt a /dev könyvtárban található, eszközleíróknak (device node) nevezett speciális állományokon keresztül érhetjük el.

3.11.1. Eszközleírók létrehozása

Amikor egy újfajta eszközt adunk hozzá a rendszerhez vagy csak annak egy új példányát, mindig létre kell hoznunk hozzá egy új eszközleírót.

DEVFS (DEVice File System, Eszköz-állományrendszer)

Az eszközöket tartalmazó állományrendszer, avagy DEVFS, ad hozzáférést a rendszermag által ismert eszközök neveihez a globális állományrendszer nevein keresztül. Így ahelyett, hogy magunknak kellene létrehoznunk és módosítanunk az eszközleírókat, a DEVFS erre a célra fenntart egy külön állományrendszert.

A devfs(5) man oldalon olvashatunk bõvebben errõl.

3.12. Bináris formátumok

Annak megértéséhez, hogy a FreeBSD miért az elf(5) formátumot használja, elõször is tisztában kell lennünk a UNIX® típusú rendszerekben használt végrehajtható állományok három "uralkodó" formátumával:

- a.out(5)

  A legõsibb és egyben a "klasszikus" UNIX®-os tárgykódformátum. Egy tömör és rövidke fejlécet használ, aminek az elején a formátum leírására szolgáló "bûvös szám" található (errõl bõvebben lásd a.out(5)). Három betöltött szegmenst tartalmaz: .text, .data és .bss, valamint egy szimbólumokat és karakterláncokat tároló táblát.

- COFF

  Az SVR3 tárgykódformátuma. A fejléc itt már tartalmaz egy table nevû szegmenst is, tehát a .text, .data és .bss szegmensekhez hasonlóan ebbõl is többet tud tárolni.

- elf(5)

  A COFF után következõ formátum, amelyben több szegmens is megtalálható, valamint létezik 32 bites és 64 bites változatban is. Egyetlen hátránya van: az ELF tervezése során rendszerarchitektúránként csupán egyetlen ABI-t (bináris alkalmazói felületet) feltételeztek. Ez azonban meglehetõsen helytelen, mivel még a kereskedelmi SYSV világában (ahol már legalább három ABI található: SVR4, Solaris és SCO) sem állja meg a helyét.

  A FreeBSD ezt a problémát a *megbélyegzés* (branding) segítségével próbálja megoldani, aminek révén el tudunk látni egy ismert ELF állományt a futtatásához megfelelõ ABI-ra vonatkozó

információkkal. Errõl részletesebben a brandelf(1) oldalán tájékozódhatunk.

A FreeBSD a "klasszikusok" táborából indult, ezért kezdetben az a.out(5) formátumot használta, mivel ez a technológia a BSD kiadások számos generációjában megmérettetett és bevált, egészen a 3.X ág elindulásáig. Habár már jóval elõtte lehetett fordítani és futtatni natív ELF binárisokat (és rendszermagokat) a FreeBSD rendszereken, a FreeBSD kezdetben ódzkodott váltani az alapértelmezés szerinti ELF formátumra. De vajon miért? Nos, amikor a Linux-tábor megtette a maga fájdalmas váltását az ELF-re, az nem annyira azért volt, hogy megszabaduljanak az a.out végrehajtható formátumtól, hanem mert a rugalmatlan, ugrótáblákon alapuló oszottkönyvtár-kezelési mechanizmusaik nagyon megnehezítették a gyártók és fejlesztõk számára az osztott függvénykönyvtárak létrehozását. Mivel az ELF formátumhoz rendelkezésre álló eszközök megoldást kínáltak az osztott könyvtárak gondjaira, és mivel általánosan elfogadták "a jövõbe vezetõ útként", a FreeBSD is felvállalta az átállással kapcsolatos költségeket és végrehajtotta azt. A FreeBSD az osztott könyvtárakat leginkább a Sun SunOS™ rendszeréhez hasonlóan kezeli, ami egy nagyon könnyen használható megoldás.

De miért van ilyen sok különbözõ formátum?

A ködös és sötét múltban egyszerûbb hardverek voltak. Ezek az egyszerû hardverek egyszerû, kicsi rendszereket támogattak. Az a.out tökéletesen megfelelõ volt egy ilyen egyszerû rendszer (egy PDP-11) binárisainak tárolására. Ahogy az emberek nekiláttak átültetni errõl az egyszerû rendszerrõl a UNIX®-ot más rendszerekre, az a.out formátumot továbbra is megtartották, mivel a UNIX® kezdeti, Motorola 68k-ra, VAXenre készített átírataihoz is elegendõ volt.

Ezután néhány éles elméjû hardvermérnök kitalálta, ha rá tudnák kényszeríteni a programokat egy-két ügyetlen trükkre, akkor a terveken meg tudnának spórolni néhány logikai kaput és ezzel a processzor is gyorsabban tudna futni. Miközben az a.out formátumot ilyen hardverre (amit manapság RISC-nek hívnak) is szerették volna áthozni, kiderült, hogy ebben az esetben szinte használhatatlan. Ezért az a.out formátum által felkínáltnál nagyobb teljesítmény elérése érdekében nekiláttak számos más formátumot is kidolgozni. Ekkor jöttek létre a COFF, ECOFF és más hasonló formátumok, amelyek elõbb-utóbb korlátokba ütköztek, még mielõtt a történelem megállapodott volna az ELF formátumnál.

Ráadásul a programok méretei egyre inkább kezdtek nõni, miközben a lemezek (valamint a fizikai memória) továbbra is viszonylag kicsik maradtak, ezért megszületett az osztott könyvtár ötlete, és a virtuális memóriát kezelõ alrendszer is sokat finomodott. Mivel ezek a különbözõ fejlesztések az a.out formátumra épültek, annak használatósága a beletömött módosítások számával együtt romlott. Emellett az emberek még szerettek volna betölteni különféle dolgokat futási idõben dinamikusan, vagy éppen a memória és a lapozóállomány megspórolásához kipucolni a programjaik egyes részeit az inicializáló kódrészletek lefutása után. A programozási nyelvek is fejlõdtek, és az emberek a fõprogram futása elõtt is akartak kódot futtatni. Az a.out formátum rengeteg apró foltozáson esett keresztül, amelyek egy ideig még tudták is tartani magukat. Azonban egy idõ után már az a.out formátum egyre növekvõ teljesítménycsökkenés nélkül már nem volt képes állni a sarat. Habár az ELF megszüntette a fennálló problémák jelentõs részét, egyúttal megnehezítette egy alapvetõen mûködõ rendszer leváltását. Ezért az ELF formátumnak meg kellett várnia azt a pillanatot, amikorra az a.out használata már kényelmetlenné vált.

Azonban ahogy múlt az idõ, az eszközökbõl, amelyekbõl a FreeBSD a fordításához szükséges eszközöket származtatta (különösen az assembler és a betöltõ),létrejött két párhuzamos fejlesztési

fa. A FreeBSD-fa kiegészült az osztott könyvtárak támogatásával és hibákat javított, miközben a GNU-fa alkotói, akik eredetileg készítették ezeket a programokat, újraírták az eszközeiket és a keresztfordításhoz egyszerûbb támogatást készítettek, cserélhetõvé tették a különbözõ formátumokat és így tovább. Sokan akartak FreeBSD-re keresztfordítani, azonban nem volt szerencséjük, mert a FreeBSD régebbi forrásait az as és ld már nem emésztette meg. Az új GNU eszköztár (a binutils) viszont ismeri már a keresztfordítást, az ELF formátumot, az osztott könyvtárakat, a C++ kiterjesztéseit stb. Idõközben egyre több gyártó ELF formátumú binárisokat adott ki, és jó érzés volt ezeket FreeBSD-n is futtatni.

Az ELF sokkal kifejezõbb az a.out formátumnál, és jóval több bõvítési lehetõséget enged az alaprendszerben. Az ELF formátumhoz tartozó eszközöket jobban karbantartják és támogatja a keresztfordítást, ami viszont sokaknak fontos. Az ELF talán némileg lassabb, mint az a.out, azonban ez nehezen mérhetõ le. Számos részletben eltérnek ugyan, például hogyan képeznek le lapokat, hogyan kezelik az inicializáló kódot stb., de ezek egyike sem igazán fontos. Idõvel az a.out támogatása ki fog kerülni a GENERIC rendszermagból, és végül majd teljesen eltávolításra kerül, ahogy a régi a.out formátumú programok szépen lassan kifutnak.

3.13. Bõvebben olvashatunk...

3.13.1. Man oldalak

A FreeBSD legátfogóbb dokumentációja a benne található man oldalak összessége. A rendszerben található szinte majdnem mindegyik programhoz létezik egy rövid használati útmutató, amely bemutatja az adott program alapvetõ mûködését és a különbözõ beállításait. Ezek a leírások a `man` parancs segítségével jeleníthetõek meg. A `man` parancs használata egyszerû:

```
% man parancs
```

ahol a `parancs` a megismerni kívánt parancsra utal. Például ha az `ls` parancsról szeretnénk többet megtudni, írjuk be:

```
% man ls
```

Az elérhetõ használati útmutatókat a következõ számozott szakaszokra osztották:

1. Felhasználói parancsok
2. Rendszerhívások és hibakódok
3. A C függvénykönyvtár függvényei
4. Eszközmeghajtók
5. Állományformátumok
6. Játékok és egyéb szórakoztató alkalmazások
7. Egyéb információk
8. Rendszerkarbantartási és -mûködtetési parancsok

9. Rendszermagfejlesztõk számára

Bizonyos esetekben ugyanaz a téma az útmutatók több szakaszában is elérhetõ. Például létezik `chmod` felhasználói parancs és a `chmod()` rendszerhívás. Ilyenkor a `man` parancsnak meg tudjuk adni pontosan, melyik szakaszra is vagyunk kíváncsiak:

```
% man 1 chmod
```

Ennek hatására a `chmod` felhasználói parancshoz tartozó oldal jelenik meg. Írott formában a használati útmutatók különbözõ szakaszaira hagyományosan a név után zárójelbe tett számmal hivatkoznak, így a chmod(1) a `chmod` felhasználói parancs és a chmod(2) a rendszerhívás.

Ez a módszer remekül mûködik abban az esetben, amikor ismerjük a parancs nevét, azonban mit tegyünk akkor, ha nem is emlékszünk a nevére? A `man` parancs a `-k` segítségével paraméterezhetõ úgy is, hogy a parancsok leírásai között keressen valamilyen kulcsszó mentén:

```
% man -k mail
```

Ezzel a paranccsal megkapjuk azon parancsok listáját, amelyek leírásában szerepel a "mail" kulcsszó. Ez egyébként mûködésében teljesen megegyezik a `apropos` paranccsal.

Szóval szeretnénk megtudni, hogy a /usr/bin könyvtárban levõ parancsok pontosan mit is csinálnak? Ehhez írjuk be:

```
% cd /usr/bin
% man -f *
```

vagy

```
% cd /usr/bin
% whatis *
```

ami ugyanezt teszi.

3.13.2. A GNU info állományok

A FreeBSD-ben megtalálható a Szabad Szoftver Alapítvány (Free Software Foundation, FSF) által készített számos alkalmazás. Ezek a programok a szokványos man oldalakon kívül még általában tartalmaznak egy `info`nak nevezett, sokkal részletesebb hipertext alapú leírást is, amelyeket az `info` paranccsal, vagy ha van fenn emacs, akkor annak az info módjában tudjuk megjeleníteni.

Az info(1) parancs használatához ennyit kell beírnunk:

```
% info
```

Itt a h lenyomásával kapunk egy rövid bemutatkozást. A parancsok rövid listáját a ? billentyû hozza elõ.

Chapter 4. Alkalmazások telepítése: csomagok és portok

4.1. Áttekintés

A FreeBSD rendszereszközök gazdag gyûjteményével érkezik az alaprendszer részeként. Azonban a külsõ alkalmazások telepítéséhez rengeteg teendõt kell elvégeznünk. A feladat elvégzésére ezért a FreeBSD két, egymást kiegészítõ technológiát kínál fel: a FreeBSD Portgyûjteményt (telepítés forráskódból) és a csomagokat (telepítés elõre elkészített bináris csomagokból). Mind a két módszerrel fel tudjuk telepíteni a kedvenc alkalmazásunk legújabb verzióját lokálisan vagy egyenesen a hálózatról.

A fejezet elolvasása során megismerjük:

- hogyan telepítsünk külsõ fejlesztésû bináris szoftvercsomagokat;
- hogyan fordítsunk le a forrásukból külsõ fejlesztésû szoftvereket a Portgyûjtemény segítségével;
- hogyan távolítsunk el korábban már telepített csomagokat és portokat;
- hogyan bíráljuk felül a Portgyûjtemény által használt alapértelmezett értékeket;
- hogyan keressük meg a megfelelõ szoftvercsomagokat;
- hogyan frissítsük a telepített alkalmazásokat.

4.2. Az alkalmazások telepítésének összefoglalása

Ha korábban már használtunk UNIX® rendszereket, valószínûleg ismerjük a külsõ alkalmazások telepítésének jellemezõ menetét:

1. Töltsük le a szoftvert, amelyet vagy forráskód vagy pedig bináris formátumban érhetünk el.
2. Bontsuk ki az alkalmazás letöltött változatát (ez általában a compress(1), gzip(1) vagy a bzip2(1) által tömörített tar állomány).
3. Keressük meg a dokumentációt (többnyire az INSTALL vagy a README állományban található, vagy a doc/ alkönyvtárban) és olvassuk el benne, hogyan tudjuk telepíteni a szoftvert.
4. Ha a szoftver forrását töltöttük le, fordítsuk le. Elképzelhetõ, hogy ennek során szerkesztenünk kell a Makefile állományt vagy lefuttatnunk a configure szkriptet, illetve más lépéseket is el kell végeznünk.
5. Próbáljuk a ki szoftvert, majd telepítsük.

Ez annak a forgatókönyve, amikor minden hiba nélkül lezajlik. Megeshet azonban, ha olyan szoftvert telepítünk, amelyet nem kifejezetten a FreeBSD-hez terveztek, akkor javítanunk kell a forráskódban a szoftver megfelelõ mûködéséhez.

Ha sikerül mûködésre bírni, folytathatjuk FreeBSD-n a szoftver telepítését a "megszokott" módon. Habár a FreeBSD erre a célra két lehetõséget is felkínál, amivel rengeteg erõfeszítéstõl megkímélhet minket: ezek a csomagok és a portok. Az írás pillanatában közel 37000 külsõ alkalmazás érhetõ el ilyen formában.

Egy adott alkalmazás esetén a hozzá tartozó FreeBSD-s csomag mindössze egyetlen letöltendõ állományt takar. A csomag tartalmazza az alkalmazás telepítéséhez szükséges összes parancs elõre lefordított változatát, ugyanígy magát a dokumentációt is. A letöltött csomagokat a FreeBSD csomagkezelõ parancsaival vehetjük használatba: ezek a pkg_add(1), pkg_delete(1), pkg_info(1) és így tovább. Az új alkalmazások telepítése ennek köszönhetõen egyetlen paranccsal elvégezhetõ.

Egy alkalmazás FreeBSD-s portja mögött lényegében állományok gyûjteménye áll, amelyek abban segítenek, hogy automatikusan tudjunk telepíteni a forráskód felhasználásával.

Ne felejtsük el, hogy normális esetben számos lépcsõt végig kell járnunk egy program sajátkezû lefordításához (letöltés, kitömörítés, javítgatás, fordítás, telepítés). A portot alkotó állományok tartalmazzák az összes olyan szükséges információt, amelyek átengedik ezt a feladatot a rendszernek. Kiadunk néhány egyszerû parancsot és az alkalmazás magától letöltõdik, kitömörítõdik, módosítja a forráskódját, lefordul és települ.

Valójában a portrendszer használható olyan csomagok létrehozására is, amelyeket késõbb a `pkg_add` és többi hozzá hasonló, hamarosan részletesebben is bemutatandó csomagkezelõ paranccsal is kezelni tudunk.

A csomagok és a portok egyaránt képesek *függõségeket* kezelni. Tegyük fel, hogy egy olyan alkalmazást akarunk telepíteni, amely egy adott függvénykönyvtár meglététõl függ a rendszeren. Az alkalmazás és a könyvtár is elérhetõ FreeBSD portként és csomagként. Akár a `pkg_add` parancsot, akár a portrendszert használjuk az alkalmazás hozzáadására, mind a kettõ észre fogja venni, hogy a szükséges könyvtárt még nem telepítettük, ezért elõször azt fogja automatikusan telepíteni.

Tudván, hogy a két említett megoldás szinte teljesen egyenértékû, felmerülhet a kérdés: a FreeBSD mégis miért rendelkezik mindkettõvel? A csomagoknak és a portoknak is megvannak a maguk elõnyei, és hogy a kettõ közül melyiket használjuk, csak az egyéni ízlésünkön múlik.

A csomagok használatának elõnyei

- Egy csomag általában kisebb, mint az alkalmazás forráskódját tartalmazó tömörített tar állomány.
- A csomagokat nem kell fordítani. Nagyobb alkalmazások, mint például a Mozilla, KDE vagy GNOME esetén ez kulcsfontosságú lehet, fõleg abban az esetben, ha a rendszerünk ehhez nem eléggé gyors.
- A csomagok használata nem várja el tõlünk, hogy behatóbban ismerjük, miként is kell FreeBSD-n szoftvereket lefordítani.

A portok használatának elõnyei

- A csomagokat általános esetben igen óvatos beállításokkal készítik el, hiszen a lehetõ legtöbb rendszeren mûködõképesnek kell lenniük. Ha viszont portból telepítünk, nyugodtan hangolhatjuk úgy a beállításokat, hogy (például) a Pentium® 4 vagy az Athlon processzoroknak kedvezõ kódot hozzanak létre.

- Bizonyos alkalmazások fordítás idején állítandó beállításokkal rendelkeznek arról, hogy mire lesznek képesek és mire nem. Például az Apache beépített konfigurációs opciók széles kelléktárával rendelkezik. Amikor viszont portból hozzuk létre, nem kell elfogadnunk ezek alapértelmezett értékeit, hanem a saját igényeinknek megfelelõen átállíthatjuk ezeket.

  Egyes esetekben több különféle beállítást tükrözõ csomag is létezhet ugyanahhoz az alkalmazáshoz. Például a Ghostscript elérhetõ ghostscript és ghostscript-nox11 csomagként is attól függõen, hogy telepítettük-e az X11 szervert. Ez természetesen egy meglehetõsen durva kijátszása a csomagrendszernek, és gyorsan lehetetlenné is válik a használata, ha az adott alkalmazás egy-két fordítási idejû beállításnál többel rendelkezik.

- Néhány szoftver licencelése tiltja a bináris terjesztést. Ezért ezek a szoftverek kizárólag csak forráskód formájában továbbíthatóak.
- Néhányan nem bíznak meg a bináris verziókban. Ha látjuk a forráskódot is, akkor (elméletben) át tudjuk nézni, és mi magunk is megkereshetjük a benne lappangó hibákat.
- Ha vannak saját javításaink, csak a forráskód birtokában tudjuk ezeket felhasználni.
- Sokan szeretik, ha egyszerûen csak "ott van" a szoftverek forráskódja. Ha éppen unatkoznak, beléjük tudnak nézni, ötleteket és kódot tudnak belõlük meríteni (persze csak akkor, ha ezt a licenc megengedi), vagy tovább tudják ezeket fejleszteni, orvosolni tudják a hibáikat stb.

A portok frissítésérõl a FreeBSD ports levelezési lista és a FreeBSD ports bugs levelezési lista valamelyikérõl szerezhetünk naprakész információkat.

Mielõtt bármelyik alkalmazást is telepítenénk, érdemes meglátogatnunk az http://vuxml.freebsd.org oldalt, ahol a hozzá tartozó ismert biztonsági problémákról olvashatunk.

Telepíthetjük a ports-mgmt/portaudit programot is, amely automatikusan ellenõrzi a telepített alkalmazások ismert sebezhetõségeit. Ez az ellenõrzés egyébként megejthetõ minden port lefordítása elõtt is. Ezalatt a `portaudit -F -a` parancs kiadásával ellenõrizhetjük utólag a telepített csomagokat.

A fejezet fennmaradó részében megmutatjuk, hogyan használjuk FreeBSD-ben a csomagokat és portokat külsõ alkalmazások telepítésére és karbantartására.

4.3. A számunkra szükséges alkalmazások felkutatása

Mielõtt telepítenénk bármilyen alkalmazást, tudnunk kell, hogyan is nevezik.

A FreeBSD-hez elérhetõ alkalmazások listája folyamatosan növekszik. Szerencsére számos módja van annak, hogy utánajárjunk a keresett szoftvernek:

- A FreeBSD honlapján találhatunk egy rendszeresen frissülõ listát az összes elérhetõ alkalmazásról, a http://www.FreeBSD.org/ports/ címen. Itt a portok különbözõ kategóriákba sorolva találhatóak meg, ahol név szerint megkereshetjük az alkalmazást (amennyiben ismerjük), vagy végigböngészhetjük az adott kategóriában elérhetõ alkalmazásokat is.
- Dan Langille a http://www.FreshPorts.org/ címen karbantartja a FreshPorts nevû oldalt. Ezen az

oldalon folyamatosan nyomon lehet követni a Portgyûjteményben megtalálható alkalmazások változásait, lehetõvé téve, hogy egy vagy több portot is "figyeljünk", vagy e-mailt küldjünk a frissítésükrõl.

- Amennyiben nem ismerjük a keresett alkalmazás nevét, próbáljuk meg felkutatni a FreshMeaten (http://www.freshmeat.net/) vagy hozzá hasonló oldalakon, majd nézzük meg a FreeBSD honlapján, hogy az adott alkalmazást portolták-e már a rendszerre.
- Ha pontosan ismerjük a port nevét, és csak a kategóriáját kellene megkeresnünk, használjuk a whereis(1) parancsot. Egyszerûen csak adjuk ki a `whereis név` parancsot, ahol a *név* a telepítendõ program neve. Ha sikerült megtalálni, részletes információt kapunk arról, hogy hol található, valahogy így:

```
# whereis lsof
lsof: /usr/ports/sysutils/lsof
```

A fenti példában megtudhatjuk, hogy az `lsof` parancs a /usr/ports/sysutils/lsof könyvtárban található.

- Vagy egy egyszerû echo(1) paranccsal is megkereshetjük a portfában a portokat. Mint például:

```
# echo /usr/ports/*/*lsof*
/usr/ports/sysutils/lsof
```

Ez a módszer a /usr/ports/distfiles könyvtárba letöltött összes illeszkedõ állományt is kilistázza.

- Egy másik lehetõség egy adott port megtalálására, ha a Portgyûjtemény beépített keresési mechanizmusát használjuk. Ennek használatához a /usr/ports könyvtárban kell lennünk. Miután beléptünk ide, futtassuk le a `make search name=programnév` parancsot, ahol a *programnév* a keresendõ program neve. Például, ha az `lsof` programot keressük:

```
# cd /usr/ports
# make search name=lsof
Port:   lsof-4.56.4
Path:   /usr/ports/sysutils/lsof
Info:   Lists information about open files (similar to fstat(1))
Maint:  obrien@FreeBSD.org
Index:  sysutils
B-deps:
R-deps:
```

A keresés eredményében leginkább a "Path:" kezdetû sorra kell odafigyelnünk, mivel ez árulja el, hol is találhatjuk meg a portot. Az itt szereplõ többi információ nem szükséges a port telepítéséhez, ezért azokkal itt most nem foglalkozunk.

Mélyebb keresésekhez használhatjuk a `make search key=szöveg` parancsot is, ahol a *szöveg* a keresendõ szöveg(részlet) lesz. Ezt a rendszer keresni fogja a portok neveiben, megjegyzésekben, leírásokban és függõségekben. Amikor nem ismerjük a keresett program

nevét, ez olyan portok keresésére alkalmas, amelyek egy adott témához kapcsolódnak.

A fenti esetek mindegyikében a keresés nem különbözteti meg a kis- és nagybetûket. Tehát az "LSOF" keresése ugyanazt az eredményt adja, mint az "lsof" esetén.

4.4. A csomagrendszer használata

FreeBSD alatt több különbözõ módon tudunk csomagokat használni:

- A sysinstall használatán keresztül a futó rendszeren tudjuk megnézni a telepített csomagokat, tudunk vele csomagokat telepíteni vagy törölni. Ezzel részletesebben a Csomagok telepítése foglalkozik.
- A szakasz további részében ismertetett egyéb parancssoros csomagkezelõ segédprogramok.

4.4.1. Csomagok telepítése

A pkg_add(1) segédprogram segítségével telepíthetünk FreeBSD-hez készült szoftvercsomagokat lokálisan vagy a hálózaton levõ egyik szerveren megtalálható állományokból:

Példa 5. Csomagok letöltése manuálisan és telepítése lokálisan

```
# ftp -a ftp2.FreeBSD.org
Connected to ftp2.FreeBSD.org.
220 ftp2.FreeBSD.org FTP server (Version 6.00LS) ready.
331 Guest login ok, send your email address as password.
230-
230-     This machine is in Vienna, VA, USA, hosted by Verio.
230-         Questions? E-mail freebsd@vienna.verio.net.
230-
230-
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /pub/FreeBSD/ports/packages/sysutils/
250 CWD command successful.
ftp> get lsof-4.56.4.tgz
local: lsof-4.56.4.tgz remote: lsof-4.56.4.tgz
200 PORT command successful.
150 Opening BINARY mode data connection for 'lsof-4.56.4.tgz' (92375 bytes).
100% |***************************************************| 92375       00:00 ETA
226 Transfer complete.
92375 bytes received in 5.60 seconds (16.11 KB/s)
ftp> exit
# pkg_add lsof-4.56.4.tgz
```

Ha nincsenek egyáltalán helyben csomagjaink (például egy FreeBSD CD-készletben), akkor a legjobban úgy járunk, ha használjuk a pkg_add(1) `-r` kapcsolóját. Ennek hatására a segédprogram

önmagától meghatározza a szükséges állományformátumot és verziót, majd FTP-n keresztül letölti és telepíti a csomagot.

```
# pkg_add -r lsof
```

Az iménti példában a program mindenféle további beavatkozás nélkül letölti a megfelelõ csomagot és felteszi. Ha a központi helyett egy másik szervert szeretnénk használni, felül kell bírálnunk az alapértelmezett beállításokat és igényeinknek megfelelõen be kell állítanunk a PACKAGESITE környezeti változó értékét. A pkg_add(1) a fetch(3) programot használja az állományok letöltésére, amely pedig számos egyéb környezeti változót is figyel, mint például az FTP_PASSIVE_MODE, az FTP_PROXY és az FTP_PASSWORD. Ha tûzfal mögött vagyunk, ezek közül néhányat biztosan be kell majd állítanunk, vagy FTP/HTTP proxyt kell használnunk. A fetch(3) man oldalán megtaláljuk ezen változók teljes felsorolását. Figyeljük meg, hogy az `lsof-4.56.4` helyett csak `lsof`-ot adtunk meg. Amikor ugyanis kérjük a csomag letöltését is, nem szabad verziószámot megadnunk. A pkg_add(1) mindig az alkalmazás legfrissebb verzióját fogja letöltetni.

Ha a FreeBSD-CURRENT vagy FreeBSD-STABLE verziókat használjuk, a pkg_add(1) mindig az alkalmazás elérhetõ legfrissebb verzióját fogja letölteni. Ha azonban valamelyik -RELEASE verziót használjuk, a csomagnak az adott kiadáshoz készült verzióját fogja leszedni. Ezt a mûködési módot a PACKAGESITE változó felülírásával viszont meg tudjuk változtatni. Például ha a FreeBSD 5.4-RELEASE változatával dolgozunk, a pkg_add(1) alapértelmezés szerint a `ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-5.4-release/Latest/` címrõl fogja letölteni a csomagokat. Ha mi viszont a FreeBSD 5-STABLE csomagok letöltését akarjuk elérni, állítsuk az PACKAGESITE értékét a `ftp://ftp.freebsd.org/pub/FreeBSD/i386/packages-5-stable/Latest/` címre.

A csomagok .tgz és .tbz formátumokban kerülnek terjesztésre. Ezek az ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/packages/ címen, vagy pedig a FreeBSD CD-ken találhatóak meg. A 4 CD-bõl álló készlet (illetve a PowerPak stb.) minden CD-jén találhatunk csomagokat a packages/ könyvtárban. A csomagokat tároló könyvtár struktúrája hasonló a /usr/ports könyvtárban kialakított könyvtárfához. Minden kategóriának saját könyvtára van, és minden csomag megtalálható az All (összes) kategóriában.

A csomagrendszer könyvtárszerkezete tehát megegyezik a portok szétosztásával, ezáltal így képesek egymással összedolgozni a teljes csomag/port rendszer megformálásában.

4.4.2. A csomagok kezelése

A pkg_info(1) egy olyan segédprogram, amellyel készíteni lehet egy listát a telepített csomagokról, és emellett még más egyéb információkat tudhatunk meg róluk.

```
# pkg_info
cvsup-16.1          A general network file distribution system optimized for CV
docbook-1.2         Meta-port for the different versions of the DocBook DTD
...
```

A pkg_version(1) összefoglalja az összes telepített csomag verzióját. Ezenkívül össze is hasonlítja a csomagok verzióját a portfában található aktuális verziókéval.

```
# pkg_version
cvsup                         =
docbook                       =
...
```

A második oszlopban látható jelek utalnak a telepített verzió a helyi portfában található verzióéhoz viszonyított korára.

Jel	Jelentés
=	A telepített csomag verziója megegyzik a helyi portfában található verziójával.
<	A telepített verzió a portfában levõnél régebbi.
>	A telepített verzió újabb, mint a portfában található. (A helyi portfa valószínûleg nem lett frissítve.)
?	A telepített csomag nem található a portok között. (Ez akkor történhet meg, amikor például egy portot eltávolítottak a Portgyûjteménybõl vagy átnevezték.)
*	A csomagnak több verziója is jelen van.
!	A telepített csomag szerepel az indexben, de a `pkg_version` valamiért nem volt képes összehasonlítani a verziószámát az indexben levõ bejegyzéssel.

4.4.3. Csomagok törlése

Egy korábban már telepített csomag eltávolításához használjuk a pkg_delete(1) segédprogramot.

```
# pkg_delete xchat-1.7.1
```

A pkg_delete(1) használatánál szükség van a csomag teljes nevének és verziószámának megadására. A fenti parancs tehát nem mûködik, ha csak az *xchat*-et adjuk meg az *xchat-1.7.1* helyett. A telepített csomag verzióját azonban könnyedén kitalálhatjuk a pkg_version(1) alkalmazásával. Esetleg egyszerûen dzsókerkaraktereket is használhatunk:

```
# pkg_delete xchat\*
```

Ebben az esetben az összes `xchat`-tel kezdõdõ csomagot letörli.

4.4.4. Egyebek

A csomagokra vonatkozó összes információ a /var/db/pkg könyvtárban található. Az egyes csomagok leírása és hozzájuk telepített állományok listája az ezen a könyvtáron belül elhelyezkedõ állományokban tárolódik.

4.5. A Portgyûjtemény használata

A most következõ szakaszokban megismerhetjük azokat az alapvetõ utasításokat, amelyekkel a Portgyûjteményen keresztül tudunk programokat telepíteni és eltávolítani. Az ehhez használható `make` targetek és környezeti változók részletesebb leírását a ports(7) man oldalán lelhetjük meg.

4.5.1. A Portgyûjtemény beszerzése

Mielõtt bármelyik portot is tudnánk telepíteni, elsõként magát a Portgyûjteményt kell megszereznünk - ez lényegében a /usr/ports könyvtárban megtalálható Makefile állományok, javítások és leírások gyûjteménye.

A FreeBSD telepítése közben a sysinstall rákérdez a Portgyûjtemény telepítésére is. Ha erre nemet válaszoltunk volna, a portok gyûjteményét az alábbi módokon szerezhetjük be:

Procedure: A CVSup használatával

A CVSup protokoll használatával viszonylag gyorsan el tudjuk érni és naprakészen tudjuk tartani a Portgyûjtemény egy példányát. A CVSup használatát alaposabban a A CVSup használata címû függelékben ismerhetjük meg.

A FreeBSD 6.2 változatától kezdve az alaprendszerben a CVSup protokollt a csup valósítja meg. A FreeBSD korábbi változatának használói ezt a programot a net/csup porton vagy csomagon keresztül tudják telepíteni.

Gondoskodjunk róla, hogy a /usr/ports üres legyen a csup elsõ futtatása elõtt! Ha más forrásból raktuk ide a Portgyûjteményt, a csup nem fogja lenyesegetni az azóta eltávolított javításokat.

1. Futtassuk a `csup` programot:

   ```
   # csup -L 2 -h cvsup.FreeBSD.org /usr/shared/examples/cvsup/ports-supfile
   ```

   Itt írjuk át a *cvsup.FreeBSD.org* címét a hozzánk legközelebb levõ CVSup szerver címére. Az összes elérhetõ tükörszerver címét a CVSup tükrözések (CVSup oldalak) címû részben olvashatjuk.

   Ha például el akarjuk kerülni a CVSup szerver megadását a parancssorban, akkor mindenképpen a ports-supfile állományból érdemes készíteni egy saját verziót.

   a. Ebben az esetben `root` felhasználóként másoljuk a

/usr/shared/examples/cvsup/ports-supfile állományt egy új helyre, például a /root könyvtárba vagy a saját felhasználói könyvtárunkba.

b. Szerkesszük át a ports-supfile állományt.

c. Írjuk át a *CHANGE_THIS.FreeBSD.org* értéket a hozzánk legközelebb található CVSup szerverére. A CVSup tükrözések (CVSup oldalak) címû részben megtaláljuk az összes ilyen tükörszervert.

d. És most indítsuk el a `csup` parancsot az alábbi módon:

```
# csup -L 2 /root/ports-supfile
```

2. A csup(1) parancs késõbbi futása során már letölti és érvényesíti az észlelt változtatásokat a saját Portgyûjteményünkben, de a telepített portokat nem fogja újrafordítani.

Procedure: A Portsnap használatával

A Portsnap egy másik módszert képvisel a Portgyûjtemény terjesztésére, a lehetõségeinek részletesebb megismeréséhez tekintsük át A Portsnap használata címû szakaszt.

1. Töltsük le a Portgyûjtemény tömörített pillanatképét a /var/db/portsnap könyvtárba. Ha akarjuk, ezután a lépés után már lekapcsolódhatunk az internetrõl.

```
# portsnap fetch
```

2. Ha még csak elõször futtatjuk a Portsnapet, bontsuk ki az imént letöltött állapotot a /usr/ports könyvtárba:

```
# portsnap extract
```

Ha viszont már korábban is létezett a /usr/ports könyvtárunk és most csak frissítjük, akkor helyette ezt a parancsot adjuk ki:

```
# portsnap update
```

Procedure: A sysinstall használatával

Ebben az esetben a sysinstall nevû programmal telepítjük a Portgyûjteményt valamilyen telepítõeszközrõl. Ilyenkor azonban a kiadás dátumának megfelelõ, valószínûleg régebbi változat kerül fel. Ha rendelkezünk internet-hozzáféréssel, akkor inkább az elõbb tárgyalt módszerek valamelyikét alkalmazzuk.

1. `root` felhasználóként adjuk ki a `sysinstall` parancsot, ahogy itt is láthatjuk:

```
# sysinstall
```

2. Menjünk le és álljunk meg a Configure (Beállítások) menüpontnál, és nyomjunk `Enter` billentyût.
3. Menjünk le és keressük meg a Distributions (Terjesztések) menüponot, majd nyomjuk meg az `Enter` billentyût.
4. Menjünk le, válasszuk ki a ports elemet a `Szóköz` megnyomásával.
5. Menjünk fel az Exit (Kilépés) ponthoz, nyomjuk meg az `Enter` billentyût.
6. Válasszuk ki a telepítéshez használni kívánt eszközt, mint például CD, FTP stb.
7. Menjünk fel az Exit (Kilépés) menüpontig, majd nyomjuk meg az `Enter` billentyût.
8. Végezetül lépjünk ki a sysinstall programból, aminhez nyomjuk meg az `X` billentyût.

4.5.2. Portok telepítése

A "váz" fogalma az elsõ, amit a Portgyûjteménnyel kapcsolatban tisztázni kell. Dióhéjban összefoglalva, egy port váza azon állományok legszûkebb halmaza, amelyek elárulják a FreeBSD számára, hogyan fordítsuk le hibamentesen és hogyan telepítsük az adott programot. Ehhez minden port vázában megtalálható:

- Egy Makefile nevû állomány. Ez tartalmazza azokat a különbözõ utasításokat, amelyek megmondják, hogyan kell lefordítani és hova kell telepíteni a rendszerünkben az adott alkalmazást.
- Egy distinfo nevû állomány. Ebben található információ a port lefordításához szükséges állományok letöltésérõl, valamint a letöltött állományok ellenõrzéséhez szükséges (az md5(1) és sha256(1) programokkal számolt) ellenõrzõösszegek.
- Egy files alkönyvtár. Itt találhatjuk meg azokat a javításokat, amelyek alkalmazásával le tudjuk fordítani a programot FreeBSD-n is. Ezek a javítások többnyire bizonyos állományok módosításaira vonatkozó apró állományok formájában jelennek meg. Természetüknél fogva szöveges formátumúak, és általában olyanok szerepelnek bennük, hogy "Töröld a 10. sort" vagy "Változtasd meg a 26. sort erre: ...". Ezeket a javításokat eredetileg patcheknek (foltoknak) nevezik, vagy másképp diffeknek (eltéréseknek) is, mivel a diff(1) program segítségével hozzák ezeket létre.

  Ez a könyvtár tartalmazhat további állományokat is portok elkészítéséhez.

- Egy pkg-descr nevû állomány. Ez a program részletesebb, gyakran többsoros bemutatása.
- Egy pkg-plist nevû állomány. Itt találjuk meg a port által telepítendõ összes állományt. Ez egyben közli a portrendszerrel is, hogy az eltávolítás során mely állományokat kell majd törölnie.

Egyes portokban szerepelhetnek még egyéb állományok is, mint például a pkg-message. Ezeket az állományokat a portrendszer különleges helyzetek kezelésére tartogatja. Ha még többet kívánunk megtudni ezekrõl az állományokról, vagy magukról a portokról általánosságban, lapozzuk fel a FreeBSD porterek kézikönyvét.

A port ugyan tartalmazza a forráskód lefordításához szükséges utasításokat, de konkrétan a forráskódot nem. Ezt egy CD-rõl vagy az internetrõl tudjuk megszerezni. A forráskód általában a szerzõje által kedvelt formában jelenik meg: ez gyakran egy gzip-pel tömörített tar állomány, de lehet tömörítve mással is, vagy éppen lehet tömörítetlen. A program forráskódját, legyen akármilyen formában is, nevezik "distfile"-nak (terjesztési állománynak). A FreeBSD portok telepítésének két módszerét tárjuk fel a következõkben.

A portok telepítéséhez `root` felhasználóként kell bejelentkeznünk.

Mielõtt telepítenénk bármelyik portot is, ajánlott frissíteni a Portgyûjteményünket és ellenõriznünk az adott portot a http://vuxml.freebsd.org címen található biztonsági adatbázisban.

Az újonnan telepítendõ alkalmazások biztonsági sebezhetõségeinek ellenõrzését automatikussá is tehetjük a portaudit használatával. Ez a segédeszköz is a Portgyûjteményben található (ports-mgmt/portaudit). Érdemes minden port telepítése elõtt letöltenünk a legfrissebb sebezhetõségi adatbázist a `portaudit -F` parancs kiadásával. Mellesleg az adatbázis rendszeres frissítése és ez a biztonsági felülvizsgálat a naponként elvégzendõ biztonsági ellenõrzések közt is megjelenik. Ezekrõl részletesebben a portaudit(1) és periodic(8) man oldalakon olvashatunk.

A Portgyûjtemény feltételezi, hogy mûködõ internet-hozzáféréssel rendelkezünk. Amennyiben ez nem így lenne, a terjesztési állományokat, forráskódokat saját magunknak kell bemásolnunk a /usr/ports/distfiles könyvtárba.

A kezdéshez lépjünk be a telepítendõ port könyvtárába:

```
# cd /usr/ports/sysutils/lsof
```

Miután beléptünk az lsof könyvtárába, láthatjuk a port vázát. A következõ lépés a fordítás, avagy a port "buildelése" (elkészítése). Ezt egy szimpla `make` parancs kiadásával kezdeményezhetjük. Miután megtettük, valami ilyesmit kell tapasztalnunk:

```
# make
>> lsof_4.57D.freebsd.tar.gz doesn't seem to exist in /usr/ports/distfiles/.
>> Attempting to fetch from ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof/.
===>  Extracting for lsof-4.57
...
[ide jön a kitömörítés kimenete]
...
>> Checksum OK for lsof_4.57D.freebsd.tar.gz.
===>  Patching for lsof-4.57
===>  Applying FreeBSD patches for lsof-4.57
===>  Configuring for lsof-4.57
...
[ide jön a configure szkript kimenete]
...
```

```
===>  Building for lsof-4.57
...
[ide jön a fordítás kimenete]
...
#
```

A fordítás befejeztével visszakapjunk a parancssort. A soron következõ lépés a port telepítése lesz. Ehhez mindössze egyetlen szóval kell kiegészítenünk a `make` parancs meghívását: ez a szó pedig az `install` (telepít) lesz.

```
# make install
===>  Installing for lsof-4.57
...
[a telepítés kimenete kimarad]
...
===>   Generating temporary packing list
===>   Compressing manual pages for lsof-4.57
===>   Registering installation for lsof-4.57
===>  SECURITY NOTE:
      This port has installed the following binaries which execute with
      increased privileges.
#
```

Miután ismét visszakaptuk a parancssort, már futtatni is tudjuk a frissen telepített alkalmazásunkat. Mivel az `lsof` programnak tovább jogosultságokra is szüksége van, egy errõl szóló biztonsági figyelmeztetést is láthatunk. A portok létrehozása és telepítése során érdemes figyelnünk az ehhez hasonló figyelmeztetésekre.

A telepítés befejeztével nem árt törölnünk a fordításhoz felhasznált alkönyvtárat (work) is. Ezzel nemcsak a drága lemezterületet spóroljuk meg, hanem megelõzzük a port késõbbi frissítése során felmerülõ esetleges problémákat is.

```
# make clean
===>  Cleaning for lsof-4.57
#
```

Az eljárásból két lépést meg is tudunk takarítani, ha egyszerûen csak a `make install clean` parancsot adjuk ki az elõbb három lépésben tagolt `make`, `make install` és `make clean` parancsok helyett.

Bizonyos parancsértelmezõk a `PATH` környezeti változóban felsorolt könyvtárakban található parancsokat gyorsítótárban tárolják, ezzel felgyorsítva a hozzájuk tartozó végrehajtható állományok keresését. Ha történetesen ilyen parancsértelmezõt használnánk, az új portok telepítése után szükségünk lehet a `rehash` parancs kiadására, mivel enélkül nem tudjuk elérni a frissen telepített parancsokat. Ezt a parancsot például a `tcsh` és a hozzá hasonló parancsértelmezõkben találhatjuk

> meg, az `sh` és rokonainál pedig a `hash -r` ennek a megfelelõje. A pontos információkat errõl a témáról a parancsértelmezõnk dokumentációjában lelhetjük meg.

Némely külsõ DVD termék, mint például a FreeBSD Malltól megrendelhetõ FreeBSD Toolkit, tartalmazhatnak terjesztési állományokat. Ezek remekül használhatóak a Portgyûjteménnyel. Ehhez csatlakoztatnunk kell a DVD-t a /cdrom könyvtárba. Ettõl eltérõ csatlakozási pontok használata esetén ne felejtsük el átállítani a `CD_MOUNTPTS` változót sem a `make` számára. Ekkor a fordításhoz szükséges állományokat úgy fogja kezelni a rendszer, mintha a merevlemezünkön lennének.

> Vigyázzunk arra, hogy néhány portot nem lehet CD-n terjeszteni. Ez részben azért lehet, mert a szükséges állományok letöltéséhez, illetve újbóli terjesztéséhez ki kell tölteni valamilyen regisztrációs nyomtatványt, vagy pedig egyéb okok miatt. Tehát ha olyan portot akarunk telepíteni, ami nincs rajta a CD-n, mindenképpen rendelkeznünk kell internetkapcsolattal.

A portrendszer a fetch(1) segédprogramot használja az állományok letöltésére, amely figyelembevesz különféle környezeti változókat, ilyenek többek közt az `FTP_PASSIVE_MODE`, `FTP_PROXY` és az `FTP_PASSWORD`. Ha tûzfal mögött vagyunk, szükségünk lehet ezek némelyikének helyes beállítására, vagy FTP/HTTP proxyt kell használnunk. A fetch(3) man oldala tartalmazza ezen változók teljes listáját.

A `make fetch` azon felhasználók számára nyújt segítséget, akik nem csatlakoznak minden esetben a hálózatra. Egyszerûen csak futtassuk le a könyvtárszerkezet legtetejérõl (/usr/ports) ezt a parancsot és a szükséges állományok letöltõdnek nekünk. A parancs mûködik az alsóbb szinteken is, például a /usr/ports/net könyvtárban. Azonban legyünk tekintettel arra, hogy ha egy port függ más portoktól vagy függvénykönyvtáraktól, ez a parancs *nem fogja* letölteni a hozzájuk tartozó állományokat. Ilyenkor a `fetch` helyett használjuk a `fetch-recursive` targetet.

> Ha a `make` parancsot egy felsõbb szinten futtatjuk, akkor ezzel létre tudjuk hozni az összes vagy csak kategóriánként az összes portot, hasonlóan az elõbb említett `make fetch` módszerhez. Ez azonban veszélyes, mivel egyes portok kizárják mások használatát. Emellett elõfordulhat az is, hogy bizonyos portok ugyanazon a néven telepítenek több, tartalmukban különbözõ állományt.

Nagyon ritkán adódhat, hogy a felhasználónak nem a `MASTER_SITES` által mutatott helyekrõl kell beszereznie a szükséges állományokat (innen töltõdnek ugyanis le). A `MASTER_SITES` beállítást az alábbi paranccsal bírálhatjuk felül:

```
# cd /usr/ports/könyvtár
# make MASTER_SITE_OVERRIDE= \
ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/distfiles/ fetch
```

Ebben a példában a `MASTER_SITES` értékét a `ftp.FreeBSD.org/pub/FreeBSD/ports/distfiles/` címre változtattuk meg.

A portok némelyike lehetővé teszi (esetleg meg is követeli), hogy engedélyezzük vagy letiltsuk a készülő program bizonyos elemeit hatékonysági, biztonsági vagy egyéb testreszabási irányelvek mentén. Ilyen többek közt a www/mozilla, a security/gpgme és a mail/sylpheed-claws. Ha elérhetőek ilyen beállítási lehetőségek, arról a rendszer egy üzenetben tájékoztat minket.

Az alapértelmezett könyvtárak felülbírálása

Néha hasznos (vagy kötelező) lehet eltérő munka- és célkönyvtárak alkalmazása. A `WRKDIRPREFIX` és a `PREFIX` változókkal ezek alapértelmezéseit tudjuk megváltoztatni. Például a

```
# make WRKDIRPREFIX=/usr/home/example/ports install
```

parancs a portot a /usr/home/example/ports könyvtárban fogja lefordítani és az eredményét a /usr/local könyvtárba telepíti. A

```
# make PREFIX=/usr/home/example/local install
```

parancs hatására a port a /usr/ports könyvtárban készül el és a /usr/home/example/local könyvtárba települ.

Természetesen a

```
# make WRKDIRPREFIX=../ports PREFIX=../local install
```

parancs ötvözi az előbbi kettőt (amelyet most túlságosan is hosszú lenne kiírni, de vélhetően sejthető belőle az alapötlet).

Lehetőség van ezen változókat a saját környezetünkben is beállítani. Ha erre lenne szükségünk, nézzünk utána az ezzel kapcsolatos teendőnek a parancsértelmezőnk man oldalán.

Az `imake` használatáról

Bizonyos portok az (X Window System részeként megjelenő) `imake` segédprogramra támaszkodnak, ahol viszont nem működik a `PREFIX` átállítása és mindenképpen a /usr/X11R6 könyvtárba akar telepíteni. Ehhez hasonlóan egyes Perl portok figyelmen kívül hagyják a `PREFIX` változót és közvetlenül a Perl fájába kerülnek. Az ilyen portok esetén nagyon nehéz vagy szinte lehetetlen betartatni a `PREFIX` használatát.

A portok újrakonfigurálása

Egyes portok lefordítása előtt megjelenik egy ncurses alapú menü, ahol ki tudunk választani bizonyos fordítási beállításokat. Gyakran előfordul, hogy a port lefordítása után a felhasználók szeretnék újra előhozni ezt a menüt és megadni vagy kivenni bizonyos beállításokat. Erre több mód is kínálkozik. Egyik ilyen lehetőség az, ha belépünk a port könyvtárába és kiadjuk a `make config` parancsot, amivel lényegében ismét előcsaljuk a beállításokat összefoglaló menüt. Másik ilyen

lehetõség a `make showconfig` alkalmazása, amivel a porthoz tartozó összes beállítást tudjuk egyszerre megjeleníteni. Ezek mellett még használható a `make rmconfig` parancs is, amivel törölni tudjuk az összes eddigi beállítást és így újrakezdhetjük a port konfigurációját. Ezek és a többi ilyen opció a ports(7) man oldalon kerül bõvebb kifejtésre.

4.5.3. A portok eltávolítása

Most már tudjuk, miként lehet portokat telepíteni, azonban valószínûleg még az is érdekelhet minket, hogy miként kell ezeket eltávolítani abban az esetben, ha például késõbb meggondolnánk magunkat velük kapcsolatban. A korábban telepített példaportot fogjuk eltávolítani (a figyelmetlenek kedvéért megemlítjük, hogy ez az `lsof` volt). A portok eltávolítása teljesen egybevág a csomagokéval (errõl a csomagokról szóló részben beszéltünk), mivel ekkor is használhatjuk a pkg_delete(1) parancsot:

```
# pkg_delete lsof-4.57
```

4.5.4. A portok frissítése

Elõször is a pkg_version(1) parancs felhasználásával listázzuk ki azokat a portokat, amik felett már eljárt az idõ és a Portgyûjteményben található belõlük újabb verzió:

```
# pkg_version -v
```

A /usr/ports/UPDATING állomány

Miután frissítettük a Portgyûjteményünket, de még mielõtt megpróbálnánk akármelyik portot is frissíteni, érdemes egy pillantást vetnünk a /usr/ports/UPDATING állományra. Itt megtalálhatóak azok a problémák és a hozzájuk tartozó lépések, amelyekkel a felhasználóknak a portok frissítése során szembe kell nézniük, beleértve az állományformátumok, a konfigurációs állományok helyének megváltozását vagy egyéb olyan módosításokat, amik a korábbi verziókkal összeférhetetlenséget szülhetnek.

Amennyiben az UPDATING állomány tartalma ellentmondana az itt olvasottakkal, mindig az UPDATING állományban leírtak az irányadóak.

Portok frissítése a portupgrade használatával

A portupgrade nevû segédprogramot a portok egyszerûbb frissítésére találták ki, és a ports-mgmt/portupgrade portban található meg. A `make install clean` paranccsal bármelyik más porthoz hasonlóan telepíthetjük:

```
# cd /usr/ports/ports-mgmt/portupgrade
# make install clean
```

A `pkgdb -F` paranccsal fésültessük át a telepített portok listáját, és javítsuk az általa jelentett ellentmondásokat. Érdemes rendszeresen elvégezni ezt, lehetõleg minden frissítés elõtt.

Miután kiadtuk a `portupgrade -a` parancsot, a portupgrade nekilát frissíteni az összes elavult portot a rendszerünkben. Ha minden egyes frissítést külön meg szeretnénk erõsíteni, használjuk a `-i` kapcsolót is.

```
# portupgrade -ai
```

Ha nem akarjuk az összes portot frissíteni, csupán egy bizonyos alkalmazását, használjuk a `portupgrade pkgname` paraméterezést. A `-R` kapcsoló megadásával a portupgrade elõször frissíti az adott alkalmazás függõségeit.

```
# portupgrade -R firefox
```

Ha a mûvelet során csomagokat kívánunk használni portok helyett, adjuk meg a `-P` kapcsolót. Ennek révén a portupgrade megkeresi a csomagokat a `PKG_PATH` környezeti változóban felsorolt könyvtárakban vagy ha itt nem találja, letölti ezeket egy távoli szerverrõl. Amennyiben a csomagokat sem helyben, sem pedig a távoli szerveren nem találja, a portupgrade helyettük portokat fog használni. Ilyenkor a portok használatát a `-PP` kapcsoló beállításával lehet elkerülni:

```
# portupgrade -PP gnome2
```

Csak a terjesztési állományok (vagy a `-P` esetén csomagok) letöltéséhez használjuk a `-F` kapcsolót. Mindezekrõl részletesebben a portupgrade(1) man oldalon olvashatunk.

Portok frissítése a Portmanager használatával

A Portmanager egy másik hasznos segédprogram a portok könnyû frissítéséhez. A ports-mgmt/portmanager porton keresztül érhetõ el:

```
# cd /usr/ports/ports-mgmt/portmanager
# make install clean
```

Használatával az összes telepített port egyetlen paranccsal frissíthetõ:

```
# portmanager -u
```

Ha a Portmanager minden egyes lépését külön meg kívánjuk erõsíteni, akkor a `-ui` kapcsolókat se felejtsük el megadni. A Portmanager emellett új portok telepítésére is használható. Eltérõen a `make install clean` parancsban megszokottaktól, a kiválasztott port összes függõségét még a fordítás és a telepítés elõtt fogja frissíteni.

```
# portmanager x11/gnome2
```

Ha bármilyen gondot tapasztalnánk a kiválasztott port függõségeit illetõen, a Portmanagert

felkérhetjük az összes függõség helyes sorrendben történõ újrafordítására. Amikor befejezte, a problémás portot is újra létrehozza.

```
# portmanager graphics/gimp -f
```

Bõvebb információkért lásd portmanager(1).

Portok frissítése a Portmaster használatával

A Portmaster szintén a portok frissítésére alkalmas segédprogram. A Portmaster esetében a hangsúly az "alaprendszerben" is megtalálható eszközök használatán van (tehát nem függ semmilyen más porttól) és a /var/db/pkg/ könyvtárban található információk alapján dönti el, hogy milyen portokat kell frissítenie. A ports-mgmt/portmaster portból érhetõ el:

```
# cd /usr/ports/ports-mgmt/portmaster
# make install clean
```

A Portmaster a portokat az alábbi négy kategória valamelyikébe sorolja be:

- Gyökér (root) portok (nem függenek semmitõl, semmi sem függ tõlük)
- Törzs (trunk) portok (nem függenek semmitõl, de mások függenek tõlük)
- Ág (branch) portok (vannak függõségeik és mások is függenek tõlük)
- Levél (leaf) portok (vannak függõségeik, de semmi sem függ tõlük)

A következõ paranccsal le tudjuk kérni az összes telepített portot és az `-L` kapcsolóval frissítéseket keresni hozzájuk:

```
# portmaster -L
===>>> Root ports (No dependencies, not depended on)
===>>> ispell-3.2.06_18
===>>> screen-4.0.3
        ===>>> New version available: screen-4.0.3_1
===>>> tcpflow-0.21_1
===>>> 7 root ports
...
===>>> Branch ports (Have dependencies, are depended on)
===>>> apache-2.2.3
        ===>>> New version available: apache-2.2.8
...
===>>> Leaf ports (Have dependencies, not depended on)
===>>> automake-1.9.6_2
===>>> bash-3.1.17
        ===>>> New version available: bash-3.2.33
...
===>>> 32 leaf ports

===>>> 137 total installed ports
```

```
        ===>>> 83 have new versions available
```

Az összes telepített port egyetlen egyszerû paranccsal frissíthetõ:

```
# portmaster -a
```

A Portmaster alapértelmezés szerint minden egyes törlendõ korábbi portról biztonsági másolatot készít. Amikor az új változat telepítése sikeresen lezajlott, akkor a Portmaster ezt a másolatot megsemmisíti. A `-b` paraméterrel azonban megkérhetjük, hogy ne törölje le a biztonsági mentést. Az `-i` megadásával a Portmaster interaktív módban indul el, és minden port frissítése elõtt a felhasználó megerõsítését fogja kérni.

Amennyiben valamilyen hiba lép fel a frissítés folyamán, az `-f` opció megadásával kérhetjük az összes port frissítését és újrafordítását is:

```
# portmaster -af
```

A Portmaster használatával új portokat is fel tudunk telepíteni a rendszerre úgy, hogy azok függõségeit is igyekszik frissíteni a lefordításuk elõtt:

```
# portmaster shells/bash
```

A további részleteket a portmaster(8) man oldalon találjuk.

4.5.5. A portok tárigénye

A Portgyûjtemény idõvel egyre több helyet fog elfoglalni a merevlemezünkön. Miután sikeresen létrehoztunk és telepítettünk egy szoftvert a hozzá tartozó portból, érdemes mindig eltakarítanunk magunk után a work könyvtárban menet közben keletkezett átmeneti állományokat a `make clean` parancs használatával. Az egész Portgyûjteményt egyetlen mozdulattal ezzel a paranccsal tudjuk végigsepregetni:

```
# portsclean -C
```

Az idõ elõrehaladtával a distfiles könyvtárban is rengeteg régi forrás tud felhalmazódni. Ezeket eltávolíthatjuk kézzel, vagy az alábbi parancs segítségével törölhetjük az összes olyan terjesztési állományt, amelyekre már egyetlen port sem hivatkozik:

```
# portsclean -D
```

Vagy törölhetjük az összes olyan terjesztési állományt, amelyre egyetlen pillanatnyilag feltelepített port sem hivatkozik a rendszerünkben:

```
# portsclean -DD
```

A `portsclean` segédprogram a portupgrade programcsomag része.

Ne felejtsük el eltávolítani azokat a portokat, amikre már nincs szükségünk a továbbiakban. Ebben a feladatban egy jól használható segédeszköz lehet a segítségünkre, a ports-mgmt/pkg_cutleaves port.

4.6. Telepítés utáni teendõk

Az új alkalmazás feltelepítése után minden bizonnyal szeretnénk elolvasni a hozzá társított dokumentációt, az egyedi beállításainknak megfelelõen módosítani a konfigurációs állományokat, engedélyezni a rendszerindítás során történõ automatikus indítását (ha démonról lenne szó) és így tovább.

Az egyes alkalmazások beállításához elvégzendõ lépések nyilvánvalóan egyedenként eltérõek. Azonban tudunk szolgálni néhány általános tanáccsal válaszként az ilyenkor felmerülõ "Na és akkor most mi legyen?" kérdésre:

- Kérdezzük meg a pkg_info(1) programtól, milyen állományok és hova kerültek fel a telepítés során. Például, ha a SzuperCsomag 1.0.0-át raktunk fel, akkor a

  ```
  # pkg_info -L SzuperCsomag-1.0.0 | less
  ```

  parancs kilistázza az összes állományt, amit a csomagból felraktunk. Ezek közül leginkább a man/ könyvtárban levõekre figyeljünk, mivel ezek lesznek az alkalmazás man oldalai. Ehhez hasonlóan az etc/ könyvtárban a konfigurációs állományok és a doc/ könyvtárban pedig a nagyobb lélegzetvételû dokumentációk foglalnak helyet.

  Ha nem emlékszünk pontosan rá, hogy az alkalmazások melyik verzióját is telepítettük, a

  ```
  # pkg_info | grep -i SzuperCsomag
  ```

  alakú parancs megkeresi az összes olyan csomagot, aminek a nevében szerepel a *SzuperCsomag* szövegrészlet. A fenti példában természetesen igény szerint változtassuk meg a *SzuperCsomag* szöveget a tényleges csomag nevére.

- Ahogy sikerült megtalálnunk az alkalmazáshoz tartozó man oldalakat, lapozzuk fel ezeket a man(1) segítségével. Ugyanígy nézzük át a mellékelt minta konfigurációs állományokat és az összes elérhetõ dokumentációt.
- Ha az alkalmazásnak van saját honlapja, kutassunk ott is információk után, olvassuk el a gyakran ismételt kérdéseket és így tovább. Ha nem tudnánk pontosan a honlap címét, a

  ```
  # pkg_info SzuperCsomag-1.0.0
  ```

kimenetéből könnyen elõkeríthetõ. Itt egy `WWW:` kezdetû sort kell keresnünk (már amennyiben létezik), amit az alkalmazás honlapjának címe kell kövessen.

- A rendszerrel együtt indítandó portok (ilyenek többek közt az internetes szolgáltatások), általában a /usr/local/etc/rc.d könyvtárba rakják a saját indítószkriptjüket. Érdemes leellenõrizni ezt a szkriptet és az igényeinknek megfelelõen módosítani, átnevezni. A Szolgáltatások indítása címû szakaszban ezt részleteiben is megismerhetjük.

4.7. Teendõ a sérült portokkal

Ha véletlenül ráakadnánk egy olyan portra, ami nem mûködik megfelelõen, nagyjából a következõket tudjuk tenni:

1. Derítsük ki a Hibajelentések adatbázisából, hogy készül-e már javítás az adott porthoz. Ha igen, akkor annak befejezése után már képesek leszünk használni.
2. Kérjük meg a port karbantartóját, hogy segítsen. A karbantartó elérhetõségének felderítéséhez gépeljük be a `make maintainer` parancsot, vagy keressük meg a Makefile állományban a karbantartó e-mail címét. Ne felejtsük el neki megemlíteni a levélben a port nevét és verzióját (vagyis mindenképpen küldjük el a `$FreeBSD:` sort a Makefile állományból) és a parancs kiadásától a hiba felbukkanásáig tartó kimenetet.

> Némely portokat nem egyedülálló személyek tartanak karban, hanem egy levelezési lista. A legtöbbjük neve, ha nem is mindé, nagyjából ilyen alakú: freebsd-listanév@FreeBSD.org. Egy ilyen jellegû kérdés megfogalmazása során ezt is vegyük figyelembe!
>
> Kifejezetten a ports@FreeBSD.org karbantartóval rendelkezõ portoknak nincs rendes gazdája. A hozzájuk kapcsolódó javítások és mindenféle segítség, ötlet errõl a levelezési listáról érkeznek. Ilyen esetekben számítunk az önkéntes segítõkre!

   Ha nem kapunk semmilyen választ, a hiba bejelentésére használhatjuk a send-pr(1) programot is (errõl bõvebben lásd a FreeBSD-s hibajelentések írása címû cikket).

3. Javítsuk meg mi magunk! A porterek kézikönyve részletesen taglalja a "portok" belsõ felépítését, így onnan elindulva akár magunktól is meg tudunk javítani egy esetlegesen sérült portot, vagy be is küldhetjük a sajátunkat!
4. Töltsük le a porthoz tartozó csomagot a hozzánk legközelebb levõ FTP oldalról. A "központi" csomaggyûjtemény az `ftp.FreeBSD.org` címen, a packages nevû könyvtárban található, de mielõtt ide fordulnánk, nézzük meg a hozzánk legközelebb levõ tükörszervert is! Ha egy csomagot így telepítünk, akkor több eséllyel fog mûködni és ráadásul még jóval gyorsabb is. A csomag telepítésére használjuk a pkg_add(1) programot.

Chapter 5. Az X Window System

5.1. Áttekintés

A FreeBSD az X11-en keresztül nyújt a felhasználók számára hatékony grafikus felhasználói felületet. Az X11 az X Window System szabadon elérhetõ változata, melyet az Xorg és az XFree86™ egyaránt implementál (valamint más egyéb programcsomagok is, amelyeket itt viszont nem tárgyalunk). A FreeBSD verziói a FreeBSD 5.2.1-RELEASE kiadással bezárólag a The XFree86™ Project, Inc. által kiadott X11 szervert, az XFree86™-ot tartalmazzák alapértelmezés szerint. A FreeBSD 5.3-RELEASE kiadástól kezdve az X11 alapértelmezett és hivatalos változata az Xorg, melyet az X.Org alapítvány a FreeBSD-éhez nagyon hasonló licenc alatt fejleszt. A FreeBSD-hez kereskedelmi X szerverek is elérhetõek.

Ebben a fejezetben az X11 telepítését és beállítását járjuk végig, miközben a hangsúlyt az Xorg 7.7 kiadására helyezzük. Az XFree86™ (vagyis a FreeBSD olyan régebbi változata, ahol az XFree86™ az alapértelmezett X11 rendszer) vagy az Xorg korábbi kiadásainak beállításával kapcsolatban mindig találhatunk információkat a FreeBSD kézikönyv http://docs.FreeBSD.org/doc/ címen található archivált változataiban.

Az X11 által támogatott megjelenítõkrõl bõvebben az Xorg honlapján olvashatunk.

A fejezet elolvasása során megismerjük:

- az X Window System különbözõ alkotóelemeit, és hogy ezek miként mûködnek együtt;
- hogyan telepítsük és állítsuk be az X11-et;
- hogyan telepítsük és használjuk a különféle ablakkezelõket;
- hogyan használjunk TrueType® betûtípusokat az X11-ben;
- hogyan állítsuk be rendszerünkön a grafikus bejelentkezést (XDM).

A fejezet elolvasásához ajánlott:

- külsõ programok telepítésének ismerete (Alkalmazások telepítése. csomagok és portok).

5.2. Az X áttekintése

Az X használata elsõre megdöbbentõ lehet azok számára, akik olyan más grafikus környezetekben járatosak, mint például a Microsoft® Windows® vagy a Mac OS®.

Míg az X minden komponensének részleteit és azok kapcsolatát nem szükséges megérteni a használatukhoz, néhány alapvetõ ismeret velük kapcsolatban elõsegíti kiaknázni az X erõsségeit.

5.2.1. Miért X?

Az X ugyan nem az elsõ UNIX®-ra íródott ablakozó rendszer, de fajtáját tekintve a legnépszerûbb. Az X eredeti fejlesztõcsapata az X elõtt egy másik ablakozó rendszeren dolgozott, aminek a neve "W" (mint "Window", azaz ablak) volt. Az X pedig az arab ábécében pontosan ezt a betût követi.

Az X-et hívhatjuk "X"-nek, "X Window System"-nek, és még sok más néven. Elõfordulhat azonban, hogy az "X Windows" elnevezés sértõ lehet egyes emberek számára. Errõl többet a X(7) man oldalon tudhatunk meg többet.

5.2.2. Az X kliens-szerver modellje

Az X-et már az elejétõl kezdve hálózatközpontúnak tervezték, és ezért az ún. "kliens-szerver" modellt használja.

Az X modelljében az "X szerver" egy olyan számítógépen fut, amelyhez billentyûzetet, monitort és egeret csatlakoztattunk. A szerver feladatai között találjuk a megjelenítés irányítását az egérrõl és a billentyûzetrõl, valamint a többi bemeneti és kimeneti eszközrõl érkezõ adatok feldolgozását és így tovább (például a digitális táblák is használhatóak beviteli eszközként, illetve egy projektor is lehet megjelenítõ). Mindegyik X alkalmazás (mint például az XTerm vagy a getenv(3)) egy kliens. A kliens üzeneteket küld a szervernek, például "Kérlek, rajzolj egy ablakot ezekre a koordinátákra", és a szerver pedig olyan üzeneteket küld, mint például "A felhasználó az OK gombra kattintott".

Az otthoni vagy a kisebb irodai környezetben az X szerver és az X kliensek általában ugyanazon a számítógépen futnak. Emellett azonban nagyon is lehetséges, hogy az X szerver egy kevésbé erõs gépen fusson, miközben az X alkalmazások (a kliensek) az irodát kiszolgáló erõsebb és drágább gépen fussanak. Egy ilyen konfigurációban az X kliensei és szerverei közti kommunikáció a hálózaton keresztül zajlik.

Jegyezzük meg, hogy az X szerver az a számítógép, ahol a monitor és a billentyûzet található, az X kliensek pedig azok a programok, amelyek az ablakokat jelenítik meg.

A protokollban semmi sem várja el, hogy a kliens és a szerver ugyanazon az operációs rendszeren vagy éppen ugyanolyan típusú számítógépen fusson. Ezért akár Microsoft® Windows®-on vagy Apple® Mac OS®-en is indíthatunk X szervert, és számos különbözõ szabad valamint kereskedelmi alkalmazás képes pontosan erre.

5.2.3. Az ablakkezelõ

Az X kialakításának filozófiája leginkább a UNIX® kialakításának filozófiájához hasonlítható, vagyis "eszközöket, ne szabályokat". Ez tehát azt jelenti, hogy az X nem köti meg, miként oldjuk meg vele a feladatokat. Helyette különféle eszközöket ad a felhasználó kezébe, és onnantól a saját felelõssége eldönteni, hogyan használja ki ezeket.

Ez a filozófia az X-ben egészen addig terjed, hogy nem rögzíti, hogyan nézzenek ki a képernyõn megjelenõ ablakok, miként kell ezeket mozgatni az egérrel, milyen billentyûk lenyomásával közlekedhetünk az ablakok között (ami a Microsoft® Windows® esetén az `Alt` + `Tab`), hogyan nézzen ki az ablakok címsora, a bezárás funkciónak legyen-e rajtuk gombja és így tovább.

Ehelyett az X az összes ezzel járó felelõsséget átadja az "ablakkezelõ" (window manager) részére. Tucatnyi ilyen ablakkezelõt találhatunk az X-hez: AfterStep, Blackbox, ctwm, Enlightenment, fvwm, Sawfish, twm, Window Maker és még sok más. Ezen ablakkezelõk mindegyike más és más kinézetet és hangulatot kínál fel: némelyikük támogatja a "virtuális munkaasztalok" (virtual desktop) létrehozását; néhányuk pedig megengedi, hogy mi magunk állítsuk be az asztal irányításához használt gombkombinációkat; köztük találhatunk olyat is, amelynek van "Start" gombja vagy ehhez

hasonló eszköze; némelyek közülük ismerik a "témákat", aminek révén a kinézetük és hangulatuk teljesen megváltoztatható. Az említett ablakkezelõk és társaik a Portgyûjtemény x11-wm kategóriájában érhetõek el.

Ráadásul a KDE és a GNOME munkakörnyezetek mindegyikének van saját integrált ablakkezelõje.

Az egyes ablakkezelõk mellesleg eltérõ beállítási módszerrel rendelkeznek. Némelyikük kézzel összeállított konfigurációs állományt vár, mások pedig külön grafikus eszközöket tartalmaznak erre a feladatra is. Az egyikük (a Sawfish) konfigurációs állományát például a Lisp programozási nyelv egyik dialektuásban kell megírni.

Az irányítás átadása

Az ablakkezelõ másik fontos feladata lekezelni, hogy az egérrel miként tudjuk átadni az ablakok között az irányítást, vagyis a fókuszt (focus policy). Minden ablakkezelõ rendszerben el kell tudnunk valahogy dönteni, hogy a beérkezõ billentyûleütések melyik ablakhoz vándoroljanak, valamint az ilyen értelemben aktív ablakot valamilyen módon jeleznünk is kell.

Ennek egyik ismert módszere a "fókusz kattintásra" megoldás, amely modellt a Microsoft® Windows® rendszerekben találhatjuk meg. Itt az ablakok akkor válnak aktívvá, amikor rájuk kattintunk az egérrel.

Az X viszont nem kötelezi el magát egyik vezérlésátadási módszer mellett sem, helyette az ablakkezelõ fogja majd eldönteni, melyik ablak birtokolja a fókuszt az adott pillanatban. A különbözõ ablakkezelõk különbözõ fókuszvezérlési technikákat ismernek. Mindegyikük ismeri a kattintásos fókuszt, azonban a többségük emellett még sok más megoldást is felkínál.

A legnépszerûbb fókuszvezérlési elvek:

A fókusz az egeret követi (focus-follows-mouse)

Az egérmutató alatt található ablak kapja meg fókuszt. Az érintett ablaknak nem kell feltétlenül az összes többi felett elhelyezkednie. Ilyenkor a fókuszt egyszerûen úgy vihetjük át egy másik ablakra, ha rámutatunk az egérrel, amihez még kattintanunk sem kell.

Hanyag fókusz (sloppy-focus)

Ez az elv az elõbbi apró kibõvítése. Amikor a fókusz az egérmutatót követi, és az egeret a leghátsó ablakra (vagy a háttérre) visszük, akkor valójában egyik ablak sem birtokolja az irányítást, ezért a leütött billentyûk elvesznek. A hanyag fókusz használatával azonban az irányítás csak abban az esetben kerül át máshová, amikor egy másik ablakba lépünk be, nem pedig akkor, amikor a jelenlegibõl lépünk ki.

Fókusz kattintásra (click-to-focus)

Az aktív ablakot egy egérkattintással választjuk ki. Ilyenkor a kiválasztott ablak "felemelkedhet" és a többi elõtt jelenhet meg. Ezt követõen az összes irányítás ebbe az ablakba vándorol, még abban az esetben is, amikor egy másik ablakra visszük az egérmutatót.

> Sok ablakkezelõ ismer ezekbõl különbözõ variációikat, valamint rajtuk kívül más egyéb vezérlési elvet is. Ezzel kapcsolatban az adott ablakkezelõ dokumentációjából deríthetünk ki a legtöbbet.

5.2.4. Widgetek

Az X megközelítése, vagyis az eszközök és nem a szabályok felsorakoztatása, kiterjed az egyes alkalmazásokban látható különféle widgetekre is.

A "widget" (window gadget, vagyis widget, de magyarul sok helyen a "mütyürke") elnevezést azokra a felhasználói felületen megjelenõ elemekre használjuk, amelyekkel valamilyen módon kapcsolatba léphetünk: kattinthatunk rájuk, "piszkálhatjuk" ezeket. Ilyenek többek közt a gombok, jelölõnégyzetek, rádiógombok, ikonok, listák és a többi. A Microsoft® Windows® nyelvén ezeket "vezérlõknek" (control) nevezzük.

A Microsoft® Windows® és az Apple® Mac OS® ezen a téren nagyon merev. Az alkalmazások fejlesztõinek gondoskodniuk kell róla, hogy a programjaik az elterjedt kinézetet és kialakítást kövessék. Az X viszont nem várja az egységes vezérlõeszközök vagy grafikai stílus használatát.

Ennek eredményeképpen az X cseppet sem kívánja meg az alkalmazásoktól, hogy közös kinézetben vagy viselkedésben osztozzanak. Természetesen léteznek népszerû eszközrendszerek és azoknak számos variációja is kialakult, beleértve az MIT Athenaját, a Motif®ot (amirõl a Microsoft® Windows® eszközeit is mintázták, az összes ferde élet és a három szürkeárnyalatot), az OpenLookot és társaikat.

Napjaink X alkalmazásai a KDE fejlesztéséhez használt Qt, esetleg a GNOME-hoz használt GTK+ könyvtárból származó, korszerû kinézetû widgeteket tartalmaznak. Ebbõl a szempontból megfigyelhetõ egyfajta tendencia a grafikus UNIX®-alkalmazások felépítésében, ami minden bizonnyal megkönnyíti a kezdõ felhasználók tájékozódását.

5.3. Az X11 telepítése

Az X11 FreeBSD-n alapértelmezett implementációja az Xorg. Az Xorg az X.Org alapítvány által kiadott, az X Window Systemet megvalósító nyílt forráskódú X szerver. Az Xorg az XFree86™ 4.4RC2 és X11R6.6 kódja alapján készült. A FreeBSD Portgyûjteményében jelenleg az Xorg 7.7 változata érhetõ el.

Az Xorg-ot a Portgyûjteményböl így tudjuk lefordítani, majd telepíteni:

```
# cd /usr/ports/x11/xorg
# make install clean
```

Az egész Xorg lefordításához legalább 4 GB szabad helyre van szükségünk.

Az X11-et természetesen telepíthetjük közvetlenül csomagok segítségével is. A pkg_add(1) használatával telepíthetõ bináris csomagok is elérhetõek az X11-hez. Amikor a pkg_add(1) programra bízzuk a csomag letöltését, ne adjunk meg verziószámot, a pkg_add(1) ugyanis mindig

automatikusan az alkalmazás legfrissebb verzióját tölti le.

Az Xorg csomagjának letöltéséhez és telepítéséhez egyszerûen csak ennyit írjunk be:

```
# pkg_add -r xorg
```

A fentebb megadott példák a teljes X11 rendszert telepíteni fogják, beleértve a szervereket, klienseket, betûtípusokat stb. Az X11 egyes részeihez külön találhatunk csomagokat és portokat.

Ha csak az X11 legszükségesebb elemeit szeretnénk telepíteni, akkor alternatívaként választhatjuk az x11/xorg-minimal portot.

A fejezet további részében szót ejtünk az X11, valamint egy irodai használatra alkalmas munkakörnyezet beállításáról.

5.4. Az X11 beállítása

5.4.1. Mielõtt nekilátnánk

Az X11 beállítása elõtt a célrendszer következõ adataira lesz szükségünk:

- A monitor jellemzõi
- A videokártya chipkészlete
- A videokártya memóriájának mérete

Az X11 a monitor jellemzõibõl állapítja meg, hogy milyen felbontásban és frissítési frekvenciával mûködtesse azt. Ezek általában a monitorhoz tartozó dokumentációból vagy a gyártó honlapjáról deríthetõek ki. Igazából két értékre van szükségünk: a függõleges és a vízszintes frissítési frekvenciára.

A videokártya chipkészlete határozza meg, hogy az X11 melyik meghajtóján keresztül kommunikál a grafikus hardverrel. Ez a legtöbb chipkészlet esetén magától megállapítható, de ennek ellenére mégis jó tisztában lenni ezzel arra az esetre, ha az automatikus felismerés mégsem mûködne.

A grafikus kártya memóriájának mérete határozza meg a rendszer által kihasználható felbontást és színmélységet. Ezt fontos tudunk ahhoz, hogy ismerjük a rendszerünk korlátait.

5.4.2. Az X11 beállítása

Az Xorg 7.3-as változatában gyakran mindenféle konfigurációs állomány használata nélkül egyszerûen csak adjuk ki a következõ parancsot:

```
% startx
```

A Xorg 7.4 verziójától kezdõdõen a számítógépünkhöz csatlakoztatott egerek és billentyûzetek HAL

segítségével automatikusan felismerhetõek. Ennek megfelelõen a x11/xorg port függõségeként telepítõdni fognak a sysutils/hal és devel/dbus portok, viszont az /etc/rc.conf állományban a következõ sorok hozzáadásával külön engedélyeznünk kell még ezeket:

```
hald_enable="YES"
dbus_enable="YES"
```

Ezeket a szolgáltatásokat még az Xorg beállítása elõtt el kell indítanunk (a parancssorból manuálisan vagy a rendszer újraindításával).

Bizonyos hardvereszközök esetén az automatikus felismerés még nem mûködik megbízhatóan vagy nem jól állítja be az értékeket. Ilyen esetekben kézzel kell megadnunk a szükséges beállításokat.

A különbözõ munkakörnyezetek, mint például a GNOME, a KDE vagy éppen az Xfce általában tartalmaznak olyan segédprogramokat, amelyekkel a felhasználó könnyedén be tudja állítani a megjelenítés paramétereit, többek közt a képernyõ felbontását. Tehát ha az alapértelmezések nem megfelelõek, viszont használni akarunk majd valamilyen munkakörnyezetet is, akkor egyszerûen csak telepítsük az adott környezetet és a hozzá tartozó eszközön keresztül állítsuk be a megjelenítést.

Az X11 beállítása egy többlépcsõs folyamat. Elsõ lépésünk egy alap konfigurációs állomány összeállítása lesz. Rendszeradminisztrátorként adjuk ki az alábbi parancsot:

```
# Xorg -configure
```

Ennek segítségével az X11 xorg.conf.new néven létrehozza a konfigurációs állomány vázát a /root könyvtárban (akár a su(1) parancsot használjuk, akár közvetlenül így jelentkezünk be, az így örökölt rendszeradminisztrátori szerepkör maga után vonja a $HOME könyvtár átállítását is). Az X11 megpróbálja megkeresni a célrendszerben elérhetõ grafikus eszközöket, és létrehozni egy olyan konfigurációs állományt, amely az észlelt eszközökhöz tartozó meghajtókat tölti be.

A következõ lépésünk legyen az imént létrehozott beállítás kipróbálása, amin keresztül ellenõrizhetjük, hogy az Xorg tényleg képes mûködni a célrendszer grafikus eszközén. Az Xorg 7.3 és azt megelõzõ változataiban ezt így tehetjük meg:

```
# Xorg -config xorg.conf.new
```

A Xorg 7.4 és késõbbi változataiban a próba eredménye egy fekete képernyõ lesz, amely meglehetõsen megnehezítheti az X11 helyes mûködésének megállapítását. A -retro kapcsoló használatával azonban továbbra is elérhetjük a korábbi verziókban megszokott viselkedési módot:

```
# Xorg -config xorg.conf.new -retro
```

Ha ezután a képernyõn egy fekete-fehér rácsot látunk egy X alakú egérmutatóval a közepén, akkor

jó a beállítás. A próbát úgy szakíthatjuk meg, ha elõször a Ctrl + Alt + Fn billentyûk együttes lenyomásával átváltunk valamelyik virtuális konzolra (például az F1 esetén az elsõre), majd megnyomjuk a Ctrl + C gombokat.

Az Xorg korábbi változataiban a 7.3 verzióig bezárólag a Ctrl + Alt + Backspace billentyûkombinációval tudjuk leállítani a mûködését. Amennyiben erre továbbra is szükségünk lenne, a 7.4 és késõbbi változatokban ezt úgy tudjuk engedélyezni, ha a begépeljük a következõ parancsot egy X terminálablakban:

```
% setxkbmap -option terminate:ctrl_alt_bksp
```

Egy másik lehetséges megoldás, ha a billenytûzet beállításához létrehozunk a /usr/local/etc/hal/fdi/policy könyvtárban egy konfigurációs állományt x11-input.fdi néven a hald számára. Ebben az állományban a következõknek kell szerepelnie:

```
<?xml version="1.0" encoding="utf-8"?>
<deviceinfo version="0.2">
  <device>
    <match key="info.capabilities" contains="input.keyboard">
    <merge key="input.x11_options.XkbOptions"
type="string">terminate:ctrl_alt_bksp</merge>
    </match>
</deviceinfo>
```

A hald a számítógép újraindításával fogja majd beolvasni ezt az állományt.

Ilyenkor az xorg.conf.new állomány ServerLayout vagy ServerFlags szekciójához vegyük még hozzá az alábbi sort:

```
Option  "DontZap"      "off"
```

Ha az egér még nem mûködne, mindenképpen be kell állítanunk a továbblépés elõtt. Ezzel kapcsolatban a FreeBSD telepítésérõl szóló fejezetben levõ Az egér beállításait ajánljuk elolvasásra. Fontos megemlíteni, hogy az Xorg 7.4 változatától kezdõdõen az xorg.confInputDevice szekcióit az eszközök automatikusan észlelt beállításai felülbírálják. A régebbi változatok viselkedését úgy tudjuk visszanyerni, ha a ServerLayout és ServerFlags szekciók valamelyikéhez hozzáadjuk az alábbi sort:

```
Option "AutoAddDevices" "false"
```

Ezt követõen a beviteli eszközök a lehetséges beállítási opciók (például a billentyûzet-kiosztás váltása) mentén a korábbiakban megszokott módon konfigurálhatóak.

Ahogy arról korábban szó esett, a 7.4 verziótól kezdõdõen a hald magától érzékelni fogja a számítógépre csatlakoztatott billentyûzetet. Elõfordulhat, hogy a

billentyûzet típusa vagy éppen kiosztása nem lesz megfelelõ. Ennek beállítására többnyire a népszerûbb munkakörnyezetek, mint például a GNOME, KDE vagy Xfce tartalmaznak külön segédprogramot. A setxkbmap(1) vagy a hald konfigurációs szabályával azonban akár közvetlenül is meg tudjuk változtatni a billentyûzethez társított tulajdonságokat.

Például ha egy 102 gombos billentyûzetet szeretnénk használni francia kiosztással, akkor ehhez a /usr/local/etc/hal/fdi/policy könyvtárban kell létrehoznunk egy x11-input.fdi nevû állományt a hald részére. Ebben az állományban szerepeljenek az alábbi sorok:

```
<?xml version="1.0" encoding="utf-8"?>
<deviceinfo version="0.2">
  <device>
    <match key="info.capabilities" contains="input.keyboard">
    <merge key="input.x11_options.XkbModel" type="string">pc102</merge>
    <merge key="input.x11_options.XkbLayout" type="string">fr</merge>
    </match>
  </device>
</deviceinfo>
```

Ha létezik már ilyen állományunk, akkor a billentyûzet megfelelõ beállításához egyszerûen csak másoljuk ki a fenti sorokat és adjuk hozzá.

Indítsuk újra a számítógépet, hogy a hald beolvassa az állományt.

Ugyanezt egy X terminálból is kényelmesen el tudjuk végezni:

```
% setxkbmap -model pc102 -layout fr
```

A paraméterként megadható billentyûzettípusokat és -kiosztásokat a /usr/local/shared/X11/xkb/rules/base.lst állományban találhatjuk meg.

Ezután az ízlésünknek megfelelõen hangoljuk be az xorg.conf.new állományt, nyissuk meg egy szövegszerkesztõben, például az emacs(1)-ben vagy az ee(1)-ben. Elsõként adjuk meg a célrendszerhez csatlakoztatott monitor frekvenciájára vonatkozó adatokat. Ezek általában a függõleges és a vízszintes frissítés értékei, melyeket az xorg.conf.new állomány "Monitor" szakaszában (Section) kell feltüntetni:

```
Section "Monitor"
        Identifier   "Monitor0"
        VendorName   "A monitor gyártója"
        ModelName    "A monitor típusa"
        HorizSync    30-107
        VertRefresh  48-120
EndSection
```

A konfigurációs állományból valószínûleg csak a `HorizSync` és `VertRefresh` kulcsszavak fognak hiányozni. Amennyiben ez tényleg így lenne, a megfelelõ vízszintes frissítés értékét a `HorizSync` kulcsszó után, a hozzá tartozó függõleges frissítés értékét pedig a `VertRefresh` kulcsszó után kell hozzátennünk a szakaszhoz. Az iménti példában már megadtuk a célrendszer monitorának frissítési értékeit.

Az X megengedi, hogy DPMS (Energy Star) energiagazdálkodási szabványt ismerõ monitorok lehetõséget is kihasználjuk. A xset(1) program vezérli a monitorok ki- és bekapcsolását, és segítségével készenléti vagy energiatakarékos üzemmódba tudjuk helyezni azokat. Ha engedélyezni kívánjuk a monitorunk DPMS lehetõségeit, egyszerûen csak tegyük hozzá az alábbi sort a monitorunkat leíró szakaszhoz:

```
        Option       "DPMS"
```

Ha már a xorg.conf.new konfigurációs állomány szerkesztésével vagyunk elfoglalva, válasszuk ki számunkra kedvezõ alapértelmezett felbontást és színmélységet is. Ezt a `"Screen"` (Képernyõ) nevû szakaszban tehetjük meg:

```
Section "Screen"
        Identifier "Screen0"
        Device     "Card0"
        Monitor    "Monitor0"
        DefaultDepth 24
        SubSection "Display"
                Viewport  0 0
                Depth     24
                Modes     "1024x768"
        EndSubSection
EndSection
```

A `DefaultDepth` kulcsszó után adjuk meg a rendszer alapértelmezett színmélységét. Ezt késõbb az Xorg(1) `-depth` paraméterével bírálhatjuk felül a parancssorból. A `Modes` kulcsszó után jelennek meg azok a felbontások, amelyekben az adott színmélység elérhetõ. Itt csak olyan VESA szabványú módok jelenhetnek meg, amelyet a célrendszer grafikus eszköze is támogat. A fenti példában az alapértelmezett színmélység képpontonként huszonnégy bit, és ebben a színmélységben az elfogadott felbontás 1024-szer 768 pixel.

Végezetül mentsük el a szerkesztett konfigurációs állományt és próbáljuk ki a korábban leírt módszer szerint.

A hibakeresés során maguk az X11 naplóállományai is hasznos eszköznek bizonyulhatnak, mivel ezek minden olyan eszközrõl tartalmaznak információt, amelyekhez az X11 szervernek sikerült csatlakoznia. Az Xorg naplóit a /var/log/Xorg.0.log elnevezést követõ állományokban találjuk meg. A konkrét naplók nevei Xorg.0.log-tól Xorg.8.log-ig és így tovább terjedhetnek.

Ha minden a legnagyobb rendben haladt eddig, a konfigurációs állományt el kell tennünk egy olyan

központi helyre, ahol az Xorg(1) képes lesz majd megtalálni. Ez a hely általában az /etc/X11/xorg.conf vagy a /usr/local/etc/X11/xorg.conf.

```
# cp xorg.conf.new /etc/X11/xorg.conf
```

Az X11 beállítását ezzel befejeztük. Az Xorg innentõl elindítható a startx(1) segédprogram vagy az xdm(1) használatával.

5.4.3. Témák idõsebbeknek és haladóknak

Az i810 grafikus chipkészlet beállítása

Az Intel® i810 integrált chipkészletének meghajtásához szükségünk lesz az agpart nevû AGP programozási felületre az X11-ben. Errõl az agp(4) meghajtó man oldalán olvashatuk többet.

Ennek segítségével ezt a hardvert is a többi grafikus kártyához hasonlóan állíthatjuk be. Vegyük figyelembe azonban, hogy az agp(4) meghajtót beépítve nem tartalmazó rendszermaggal futó rendszerekben a kldload(8) paranccsal utólag már nem tudjuk betölteni! Ezt a meghajtót már a rendszerindítás során be kell tudnunk tölteni: vagy a rendszermagba fordítjuk, vagy pedig a /boot/loader.conf állományban hivatkozunk rá.

Widescreen Flat Panel monitorok használata

Ebben a részben feltételezünk némi tapasztalatot a beállítások terén. Amennyiben a szabványos konfigurációs eszközök csõdöt mondtak a beállítás során, magukból a naplóállományokból is kinyerhetünk elegendõ információt ahhoz, hogy mûködésre bírjuk rendszerünket. Ehhez mindenképpen legyen kéznél egy szövegszerkesztõ!

A jelenlegi szélesvásznú (WSXGA, WSXGA+, WUXGA, WXGA, WXGA+ és társai) formátumok a 16:10-es és 10:9-es képarányokat ismerik, amik néha gondot okozhatnak. Például a 16:10-es képarány felbontásai:

- 2560x1600
- 1920x1200
- 1680x1050
- 1440x900
- 1280x800

Bizonyos szempontból egyszerûen csak a fenti felbontások valamelyikét kell felvenni a `"Screen"` szakasz `Mode` sorába, valahogy így:

```
Section "Screen"
Identifier "Screen0"
Device     "Card0"
Monitor    "Monitor0"
DefaultDepth 24
SubSection "Display"
```

```
    Viewport  0 0
    Depth     24
    Modes     "1680x1050"
EndSubSection
EndSection
```

Az Xorg elég intelligens ahhoz, hogy a szélesvásznú megjelenítéssel kapcsolatos információkat lekérje a monitor I2C/DDC adatai közül, ezért meg tudja állapítani, hogy az eszköz milyen frissítési frekvenciákat és felbontásokat bír el.

Ha az alábbi `ModeLine` értékek nem szerepelnének a meghajtókban, akkor velük kapcsolatban egy kicsit súgnunk kell az Xorg-nak. A /var/log/Xorg.0.log átrágásával elegendõ információt tudunk gyûjteni ahhoz, hogy manuálisan vegyünk fel használható `ModeLine` értékeket. Nem kell mást tennünk, mint ehhez hasonló sorokat keresnünk:

```
(II) MGA(0): Supported additional Video Mode:
(II) MGA(0): clock: 146.2 MHz   Image Size:  433 x 271 mm
(II) MGA(0): h_active: 1680  h_sync: 1784  h_sync_end 1960 h_blank_end 2240 h_border:
0
(II) MGA(0): v_active: 1050  v_sync: 1053  v_sync_end 1059 v_blanking: 1089 v_border:
0
(II) MGA(0): Ranges: V min: 48  V max: 85 Hz, H min: 30  H max: 94 kHz, PixClock max
170 MHz
```

Ezeket nevezik EDID-adatoknak (Extended display identification data, vagyis "bõvített megjelenítési azonosító adatoknak"). Belõlük a megfelelõ `ModeLine` sor létrehozása csupán annyiból áll, hogy a számértékeket a megfelelõ sorrendbe tesszük:

```
ModeLine <name> <clock> <4 horiz. timings> <4 vert.  timings>
```

Ezáltal a példában látott `"Monitor"` szakasz `ModeLine` sora így fog kinézni:

```
Section "Monitor"
Identifier      "Monitor1"
VendorName      "Bigname"
ModelName       "BestModel"
ModeLine        "1680x1050" 146.2 1680 1784 1960 2240 1050 1053 1059 1089
Option          "DPMS"
EndSection
```

Miután végrehajtottuk ezeket az egyszerû beállítási lépéseket, az X most már valószínûleg el fog indulni az új szélesvásznú monitorunkon.

5.5. Betûtípusok használata az X11-ben

5.5.1. Type1 betûtípusok

Az X11-hez tartozó alap betûtípusok nem mondhatóak kifejezetten ideálisnak például egy átlagos asztali kiadványszerkesztõ alkalmazás számára. A nagyobb méretû bemutatókon a betûi szögletesen és idétlenül néznek ki, a getenv(3)ben megjelenõ kisebb betûk pedig szinte teljességgel olvashatatlanok. Viszont manapság már rengeteg szabad, nagyon jó minõségû és könnyen használható Type1 (PostScript®) betûtípus érhetõ el az X11-hez. Például az URW betûtípus-gyûjtemény (x11-fonts/urwfonts) a szabványos Type1 betûtípusok (Times Roman™, Helvetica™, Palatino™ és még sok más) jó minõségû változatait tartalmazza. A Freefonts nevû gyûjtemény (x11-fonts/freefonts) is tartalmaz sok más betûtípust, de a legtöbbjüket inkább csak a Gimpben és a hozzá hasonló grafikai alkalmazásokban tudjuk használni, illetve nincsenek is még kellõ mértékben befejezve a hétköznapi munkákhoz. Ezeken felül az X11 minimális ügyeskedéssel beállítható a TrueType® betûtípusok használatára is. Errõl részleteket a X(7) man oldalon, illetve a TrueType® betûtípusokról szóló szakaszban olvashatunk.

A Portgyûjteményböl az imént említett Type1 betûtípusokat az alábbi parancsok segítségével telepíthetjük:

```
# cd /usr/ports/x11-fonts/urwfonts
# make install clean
```

Ugyanígy járjunk el a freefont és a többi gyûjtemény esetén is. Az X szerver akkor fogja észlelni ezeket a betûtípusokat, ha hozzáadjuk a következõ sort a konfigurációs állományához (/etc/X11/xorg.conf):

```
FontPath "/usr/local/lib/X11/fonts/URW/"
```

Vagy megtehetjük mindezt az X futtatása során is:

```
% xset fp+ /usr/local/lib/X11/fonts/URW
% xset fp rehash
```

Ez utóbbi beállítás viszont el fog veszni az X leállításával, hacsak nem vesszük hozzá az indítószkriptjéhez (ez az ~/.xinitrc a `startx` használata esetén, illetve az ~/.xsession, amikor egy XDM-szerû grafikus bejelentkezést használunk). Ezek mellett használhatjuk a /usr/local/etc/fonts/local.conf állományt is: errõl az élsimítással foglalkozó szakaszban szólunk részletesebben.

5.5.2. TrueType® betûtípusok

Az Xorg beépített támogatást tartalmaz a TrueType® betûtípusok rendereléséhez. Két különbözõ modul valósítja meg ezt a feladatot. Ebben példában a freetype nevû modult használjuk, mivel sokkal jobban illeszkedik a többi betûrenderelõhöz. A freetype modul használatához mindössze az

/etc/X11/xorg.conf állomány "Module" szakaszába kell beírnunk a következõ sort:

```
Load  "freetype"
```

Most pedig hozzunk létre egy könyvtárat a TrueType® betûtípusok számára (ez legyen például a /usr/local/lib/X11/fonts/TrueType), majd másoljuk az összes TrueType® betûtípusunkat ide. Vigyázzunk rá, hogy Macintosh®-ról TrueType® betûtípusok közvetlenül nem hozhatóak át, az X11 számára UNIX®/MS-DOS®/Windows® formátumban kell lenniük. Miután sikerült átmásolnunk az állományokat ebbe a könyvtárba, használjuk a ttmkfdir parancsot a fonts.dir állomány létrehozására, aminek révén az X betûrenderelõje tudni fogja, hogy új állományokat telepítettünk. A ttmkfdirx11-fonts/ttmkfdir néven elérhetõ a FreeBSD Portgyûjteményébõl.

```
# cd /usr/local/lib/X11/fonts/TrueType
# ttmkfdir -o fonts.dir
```

Ezután adjuk hozzá a TrueType® könyvtárat a betûtípusok könyvtáraihoz. Itt is a Type1 betûtípusoknál leírtak szerint kell eljárnunk, vagyis használjunk a

```
% xset fp+ /usr/local/lib/X11/fonts/TrueType
% xset fp rehash
```

parancsot, vagy adjunk hozzá a xorg.conf állományhoz egy további FontPath sort.

Ezzel végeztünk is. Innentõl kezdve a getenv(3), Gimp, a StarOffice™ és mindegyik X alkalmazás fel fogja ismerni a frissen telepített TrueType® betûtípusokat. A nagyon kicsi betûk (egy honlap megtekintése során, nagyfelbontásban) és a nagyon nagy betûk (a StarOffice™ használatakor) most már sokkal jobban fognak mutatni.

5.5.3. A betûk élsimítása

Az X11 által használt, a /usr/local/lib/X11/fonts/ és a ~/.fonts/ könyvtárakban található összes betûtípus élsimítása automatikusan elérhetõ az Xft-re felkészített alkalmazások számára. A mostanság megjelenõ legtöbb alkalmazás, mint például a KDE, GNOME és Firefox, ismeri az Xft-t.

A betûtípusok élsimításának be- és kikapcsolásához, valamint élsimítási jellemzõinek beállításához hozzuk létre (vagy ha már létezne, módosítsuk) a /usr/local/etc/fonts/local.conf állományt. Az Xft betûrendszer számos kifinomult lehetõsége hangolható ezzel az állománnyal, amelyekbõl ebben a szakaszban csupán rövidke ízelítõt fogunk adni. A pontosabb részletekrõl a fonts-conf(5) man oldalon tájékozódhatunk.

Az állománynak XML formátumúnak kell lennie. Különösen ügyeljünk a kis- és nagybetûkre, illetve gyõzõdjünk meg mindig róla, hogy lezártuk-e az összes taget. Az állomány a szokásos XML-fejléccel kezdõdik, amelyet egy DOCTYPE definíció követ, majd a <fontconfig> tag:

```
      <?xml version="1.0"?>
      <!DOCTYPE fontconfig SYSTEM "fonts.dtd">
```

```
    <fontconfig>
```

Ahogy azt már korábban is említettük, a /usr/local/lib/X11/fonts és a ~/.fonts/ könyvtárakban található összes betûtípus élsimítása elérhetõ az Xft-re felkészített alkalmazások számára. Amennyiben ezeken túl még további könyvtárakat is fel kívánunk venni, írjuk bele a /usr/local/etc/fonts/local.conf állományba, nagyjából ilyen alakban:

```
<dir>/az/en/betu/tipusaim</dir>
```

Az új betûtípusok, de legfõképpen az új betûtípusokat tartalmazó könyvtárak hozzáadása után a betûkkel kapcsolatos gyorsítótárak frissítéséhez mindenképpen javasolt lefuttatni az alábbi parancsot:

```
# fc-cache -f
```

Az élsimítás hatására a betûk kontúrjai egy kissé elmosódnak, aminek köszönhetõen a nagyon kis méretû szövegek sokkal olvashatóbbá válnak és eltûnnek a nagy méretû betûkrõl a "lépcsõk", azonban a normál méretû betûknél megfájdulhat tõle a szemünk. A 14 pontnál kisebb méretû betûk esetén az alábbi sorok hozzáadásával tudjuk kikapcsolni az élsimítást:

```
        <match target="font">
            <test name="size" compare="less">
                <double>14</double>
            </test>
            <edit name="antialias" mode="assign">
                <bool>false</bool>
            </edit>
        </match>
        <match target="font">
            <test name="pixelsize" compare="less" qual="any">
                <double>14</double>
            </test>
            <edit mode="assign" name="antialias">
                <bool>false</bool>
            </edit>
        </match>
```

Bizonyos egyenszélességû (monospaced) betûtípusok élsimítása esetén a betûk távolsága nem megfelelõ. Ez leginkább a KDE használata esetén merül fel. Ezt a problémát úgy is orvosolhatjuk, ha az ilyen betûtípusok térközét kézzel 100-ra állítjuk. Ehhez írjuk be a következõ sorokat:

```
       <match target="pattern" name="family">
           <test qual="any" name="family">
               <string>fixed</string>
           </test>
           <edit name="family" mode="assign">
```

```
            <string>mono</string>
        </edit>
    </match>
    <match target="pattern" name="family">
        <test qual="any" name="family">
            <string>console</string>
        </test>
        <edit name="family" mode="assign">
            <string>mono</string>
        </edit>
    </match>
```

(ezzel lefedjük összes rögzített méretû (fixed) betûtípust "mono"-ként), majd vegyük hozzá ezt is:

```
    <match target="pattern" name="family">
        <test qual="any" name="family">
            <string>mono</string>
        </test>
        <edit name="spacing" mode="assign">
            <int>100</int>
        </edit>
    </match>
```

Egyes betûtípusoknál, mint például a Helveticánál, gondok akadhatnak az élsimítással. Ez általában egy függõlegesen kettévágottnak látszó betû képében jelenik meg. De ami a legrosszabb, hogy emiatt némely alkalmazás képes összeomlani. Ennek elkerülésére tegyük hozzá még az alábbi sorokat a local.conf állományhoz:

```
    <match target="pattern" name="family">
        <test qual="any" name="family">
            <string>Helvetica</string>
        </test>
        <edit name="family" mode="assign">
            <string>sans-serif</string>
        </edit>
    </match>
```

Miután befejeztük a local.conf szerkesztését, ellenõrizzük, hogy szerepel-e az állomány végén a </fontconfig> tag. Ha ugyanis nem zárjuk le rendesen, akkor a változtatásaink érvénytelenné válnak.

Végezetül a felhasználók is megadhatják a saját beállításaikat a saját .fonts.conf állományuk segítségével. Ehhez nem kell mást tenni, mindössze létrehozni egy ~/.fonts.conf XML-állományt.

Még egy utolsó ötlet: LCD képernyõk esetén szükségünk lehet az ún. "sub-pixel sampling" (részképpont mintavételezési) technikára. Ezzel lényegében a (vízszintesen elválasztott) vörös, zöld és kék összetevõket külön-külön kezeljük a horizontális felbontás javítására. Bámulatos eredményeket lehet elérni a segítségével! A bekapcsolásához a következõ sorokat kell beszúrnunk

valahova a local.conf állományba:

```
        <match target="font">
            <test qual="all" name="rgba">
                <const>unknown</const>
            </test>
            <edit name="rgba" mode="assign">
                <const>rgb</const>
            </edit>
        </match>
```

A megjelenítõ fajtájától függõen lehet, hogy az `rgb` értéket `bgr`-re, `vrgb`-re vagy `vbgr` -re kell cserélnünk. Próbálgassuk és kiderül, hogy melyikkel mûködik jobban.

5.6. Az X bejelentkeztetõ képernyõje

5.6.1. Összefoglalás

Az X bejelentkeztetõ képernyõje (az X Display Manager vagy röviden csak XDM) az X Window System egyik kiegészítõ eleme, melyet a bejelentkezések lebonyolítására használunk. Számtalan helyzetben hasznosnak bizonyulhat, beleértve a legkisebb "X terminálokat" és a legnagyobb hálózati szervereket is. Mivel az X Window System független hálózattól és protokolltól, a hálózaton összekapcsolt, X klienseket és szervereket futtató különbözõ számítógépek széles kombinációja elõfordulhat. Az XDM egy grafikus felületen keresztül segít választani az elérhetõ szerverek között, valamint a felhasználók, például felhasználónév és jelszón keresztüli, hitelesítésében.

Az XDM tulajdonképpen a felhasználó számára ugyanazokat a funkciókat nyújtja, mint a getty(8) program (errõl bõvebben lásd Beállítás). Tehát: belépteti a felhasználót a szerverre, ahova csatlakozott, illetve elindítja helyette a hozzá tartozó munkamenet kezelõjét (ami általában egy X-es ablakkezelõ). Az XDM megvárja ennek a programnak a befejezõdését, ami egyben jelzi számára, hogy a felhasználó elvégezte a dolgát, és kilépteti a szerverrõl. Ezután az XDM újra várakozni kezd a következõ felhasználóra, miközben a bejelentkezéshez és a szerver kiválasztásához szükséges képernyõket jeleníti meg.

5.6.2. Az XDM használata

A XDM használatához elõször telepítenünk kell rendszerünkre a x11/xdm portot (mivel az Xorg újabb változatai ezt alapértelmezés szerint már nem telepítik). Ezt követõen az XDM démon a /usr/local/bin/xdm helyen található meg. A programot `root` felhasználóként bármikor tudjuk futtatni, és ez veszi kezelésbe a helyi gépen futó X szervert. Amennyiben az XDM-et a számítógép minden egyes indulása során el akarjuk indítani, egyszerûen csak adjuk hozzá a megfelelõ bejegyzést az /etc/ttys állományhoz. Ennek a formai szabályairól és használatáról bõvebben lásd Egy bejegyzés felvétele az /etc/ttys állományba. Az /etc/ttys alapértelmezett változatában az XDM démont ebben a formában találjuk meg a virtuális terminálok között:

```
ttyv8   "/usr/local/bin/xdm -nodaemon"  xterm   off secure
```

Ez a bejegyzés alapból nem aktív. Az engedélyezéséhez írjuk át az ötödik mezõben szereplõ `off` (kikapcsolva) értéket `on` (bekapcsolvá)-ra, majd indítsuk újra az init(8) programot a A init utasítása az /etc/ttys újraolvasásáraban leírtak szerint. Az elsõ mezõben találhatjuk a program által kezelt terminált, ez jelen esetünkben a `ttyv8`. Ennek megfelelõen az XDM a 9. virtuális terminálon kezdi meg a futását.

5.6.3. Az XDM beállítása

Az XDM beállításait tartalmazó könyvtár a /usr/local/lib/X11/xdm. Itt találhatjuk meg azokat az állományokat, amelyek megváltoztatásával befolyásolhatjuk az XDM megjelenését és viselkedését. Általában a következõ állományok bukkannak fel ezen a helyen:

Állomány	Leírás
Xaccess	A kliens hitelesítésének szabályrendszere.
Xresources	Az X erõforrásainak alapértelmezett értékei.
Xservers	Az ismert távoli és helyi X szerverek listája.
Xsession	A bejelentkezések során lefutó alapértelmezett szkript.
Xsetup_*	A bejelentkezõ felület indítása elõtt indítandó alkalmazásokkal kapcsolatos szkript.
xdm-config	A gépen futó összes X szerver globális beállításai.
xdm-errors	A szerver által jelentett hibák.
xdm-pid	A jelenleg futó XDM-hez tartozó azonosító.

Ebben a könyvtárban találunk még néhány olyan programot és szkriptet, amelyekkel be tudjuk állítani a munkaasztalunkat az XDM futása alatt. Ezen állományok céljait egyenként ismertetni fogjuk. A felépítésükrõl és használatukról az xdm(1) man oldala árul el többet.

Az alapértelmezett beállítás egy téglalap alakú bejelentkezõ ablak, aminek tetején nagy betûkkel a gép neve olvasható, valamint alatta a "Login:" (felhasználói név) és "Password:" (jelszó) mezõk várnak kitöltésre. Ez egy remek kiindulási alap az XDM-képernyõ kinézetének megváltoztatásához.

Xaccess

Az XDM-mel szabályozott X szerverek által használt protokoll az X Display Manager Connection Protocol (XDMCP). Ez az állomány tartalmazza a távoli számítógépekrõl érkezõ XDMCP-kapcsolatok vezérlésére vonatkozó szabályokat. Ezt a rendszer általában figyelmen kívül hagyja, hacsak az xdm-config állományban be nem állítottuk a távoli számítógépek csatlakoztathatóságát. Alapértelmezés szerint viszont semmilyen klienst nem enged csatlakozni.

Xresources

Ez tartalmazza a szerverválasztó és bejelentkezõ képernyõ alapértelmezéseit. Segítségével a bejelentkeztetést végzõ program kinézetét változtathatjuk meg. Formátuma hasonló az X11 dokumentációjában leírt app-defaults állományhoz.

Xservers

A szerverválasztó által felkínálandó távoli X szerverek felsorolását tartalmazza.

Xsession

A felhasználó bejelentkezése után ez az XDM-szkript fog lefutni. Általában minden felhasználóhoz tartozik egy saját ~/.xsession szkript, ami ezt felülbírálja.

Xsetup_*

Ezek fognak automatikusan lefutni a szerverválasztó vagy bejelentkeztetõ felületek megjelenése elõtt. Minden általunk használt X szerverhez tartozik egy ilyen szkript, amelyek neve Xsetup_-al kezdõdik és a helyi X szerver sorszámával folytatódik (például Xsetup_0). Ezek a szkriptek általában egy-két programot, mint például az `xconsole`, indítanak el a háttérben.

xdm-config

Az app-defaults nevû állományéhoz hasonló alakban tartalmaz beállításokat a program által kezelt minden egyes X szerverhez.

xdm-errors

Ebben található meg az XDM által futtatni próbált X szerverek kimenete. Itt érdemes hibaüzenetek után kutatni, ha az XDM által indított X szerver valamiért megállna. Ezek az üzenetek egyébként a felhasználó ~/.xsession-errors állományába is beíródnak.

5.6.4. Hálózati X szerver futtatása

Az X szerverünkhöz csak akkor tudnak kívülrõl más felhasználók is kapcsolódni, ha átírjuk a hozzáférésre vonatkozó szabályokat és engedélyezzük rajta a kapcsolódást. Az alapértelmezett szabályok nagyon óvatosak. Ha tehát engedélyezni akarjuk a kívülrõl érkezõ kapcsolódásokat, akkor ahhoz elõször az xdm-config állományból vegyük ki az alábbi sort:

```
! SECURITY: do not listen for XDMCP or Chooser requests
! Comment out this line if you want to manage X terminals with xdm
DisplayManager.requestPort:     0
```

Ezután indítsuk újra az XDM-et. Ne felejtsük el, hogy az app-defaults állományokban a megjegyzések "!" (felkiáltó)jellel kezdõdnek, nem pedig a megszokott "#" (kettõskereszt)tel. A fentieknél természetesen szigorúbb hozzáférési szabályok is szükségesek lehetnek - ezzel kapcsolatban nézzük meg Xaccess állományban szereplõ példákat, illetve lapozzuk fel az xdm(1) man oldalt.

5.6.5. Az XDM helyett

Az alapértelmezett XDM feladatát számos más program is képes ellátni. Ezek közül az egyik a kdm (a KDE része), amire ebben a fejezetben még vissza fogunk térni. A kdm különféle vizuális effekteket és egyéb kozmetikázást ígér, valamint lehetõvé teszi a felhasználók számára, hogy a

bejelentkezés elõtt kiválaszthassák a használni kívánt ablakkezelõt.

5.7. Munkakörnyezetek

Ebben a szakaszban a FreeBSD-n futó X-hez elérhetõ különbözõ munkakörnyezetekrõl (desktop environment) lesz szó. Maga a "munkakörnyezet" elnevezés sok mindenre utalhat egy mezei ablakkezelõtõl kezdve az asztali alkalmazások teljes garmadájáig, ahogy igaz ez a KDE vagy a GNOME esetében is.

5.7.1. A GNOME

Röviden a GNOME-ról

A GNOME egy felhasználóbarát munkakörnyezet, aminek segítségével a felhasználók számára gyerekjáték a számítógép használata és beállítása. A GNOME-ban találhatunk egy panelt (az alkalmazások indítására és különféle állapotjelzõk megjelenítéséhez), egy asztalt (ahova az alkalmazások és az adatok kerülnek), szabványos asztali eszközöket és alkalmazásokat, valamint számos konvenciót, aminek mentén az alkalmazások könnyen együtt tudnak mûködni és tartani egymással az összhangot. Más operációs rendszerek vagy környezetek ismerõi otthon érezhetik magukat ebben a GNOME által nyújtott vizuális környezetben. A FreeBSD és a GNOME kapcsolatáról bõvebb információkat a FreeBSD GNOME Projekt honlapján találhatunk. Ezen az oldalon a GNOME telepítésérõl, beállításáról és karbantartásáról egy meglehetõsen átfogó leírást olvashatunk.

A GNOME telepítése

A programot könnyen fel tudjuk telepíteni csomagból vagy a Portgyûjtemény segítségével:

A hálózatról a GNOME csomagját mindössze ennek a sornak a beírásával fel tudjuk telepíteni:

```
# pkg_add -r gnome2
```

A portfa felhasználásával pedig a GNOME-ot így tudjuk forrásból telepíteni:

```
# cd /usr/ports/x11/gnome2
# make install clean
```

Miután a GNOME-ot sikerült feltelepítenünk, meg kell mondanunk az X szervernek, hogy az alapértelmezett ablakkezelõ helyett a GNOME-ot indítsa el.

A GNOME-ot legkönnyebben a GDM, vagyis a GNOME Display Manager használatával indíthatjuk el. A GDM a GNOME részeként települ (habár alapból nincs bekapcsolva), és úgy tudjuk aktiválni, ha /etc/rc.conf állományba beírjuk a `gdm_enable="YES"` sort. Újraindítás után a GDM automatikusan elindul.

Ha a GDM mellett az összes GNOME szolgáltatást is el akarjuk indítani, vegyük fel a `gnome_enable="YES"` sort az /etc/rc.conf állományba.

A GNOME-ot parancssorból is elindíthatjuk, ha hozzá megfelelõen beállítjuk az .xinitrc nevû állományt. Ha már van egy saját .xinitrc állományunk, akkor nincs más teendõnk, mint átírni az aktuális ablakkezelõnket hívó sort a /usr/local/bin/gnome-session sorra. Ha nem csináltunk elõtte semmilyen különleges dolgot az említett konfigurációs állománnyal, akkor elegendõ csak ennyit beírnunk:

```
% echo "/usr/local/bin/gnome-session" > ~/.xinitrc
```

Ezt követõen írjuk be a `startx` parancsot, és a GNOME munkakörnyezete fog elindulni.

Ha az XDM-hoz hasonló régebbi bejelentkeztetõ képernyõt használunk, ez a módszer nem fog mûködni. Helyette hozzunk létre egy .xsession nevû futtatható állományt, amely ezt a parancsot tartalmazza. Ehhez nyissuk meg és cseréljük ki benne a korábbi ablakkezelõnk hívását a /usr/local/bin/gnome-session utasításra:

```
% echo "#!/bin/sh" > ~/.xsession
% echo "/usr/local/bin/gnome-session" >> ~/.xsession
% chmod +x ~/.xsession
```

Megcsinálhatjuk azt is, hogy a bejelentkezéskor választható legyen az ablakkezelõ. A KDE-rõl bõvebben címû szakaszban látni fogjuk, hogyan tudjuk ezt a a KDE bejelentkeztetõ képernyõje, a kdm esetén beállítani.

5.7.2. A KDE

Röviden a KDE-rõl

A KDE egy könnyen használható modern munkakörnyezet. Ízelítõül a KDE felhasználók számára felkínált lehetõségei közül:

- Gyönyörû, korszerû munkafelület
- Az asztal hálózaton keresztüli transzparens kezelése
- A KDE asztal és alkalmazásainak használatában egy beépített súgórendszer segíti a kényelmes és összefüggõ közlekedést
- A KDE alkalmazásainak összehangolt kinézete és hangulata
- Szabványosított menük és eszköztárak, billentyû-hozzárendelések, színsémák stb.
- Honosítás: a KDE több, mint 40 nyelven elérhetõ
- Központosított, összehangolt, párbeszédablak alapú asztalbeállítás
- Számos hasznos KDE-alkalmazás

A KDE-hez egy Konqueror nevû böngészõ is tartozik, mely a többi UNIX®-os böngészõ komoly ellenfelének bizonyul. A KDE-rõl többet a KDE honlapján olvashatunk. A KDE FreeBSD-re vonatkozó tudnivalóiról és a hozzá tartozó anyagokról a FreeBSD KDE csapat honlapján találhatunk információkat.

FreeBSD alatt a KDE két verziója érhetõ el: a harmadik változat már régóta használható, nagyon megbízható, amely mellett viszont a következõ generációt képviselõ negyedik változat is megtalálható a Portgyûjteményben. Akár egymás mellé is telepíthetõek.

A KDE telepítése

Ahogy a GNOME és a többi más munkakörnyezet esetében is, maga a program könnyen telepíthetõ csomagból vagy a Portgyûjtemény segítségével is:

A KDE3 csomagját hálózaton keresztül így tudjuk telepíteni:

```
# pkg_add -r kde
```

A KDE4 csomagját pedig hálózaton keresztül így tudjuk telepíteni:

```
# pkg_add -r kde4
```

A pkg_add(1) magától letölti az alkalmazás legfrissebb verzióját.

Ha a KDE3 környezetet forrásból akarjuk telepíteni, használjuk a portfát:

```
# cd /usr/ports/x11/kde3
# make install clean
```

Ha viszont a KDE4 környezetet akarjuk inkább a portfa felhasználásával forrásból telepíteni, akkor ezeket a parancsokat adjuk ki:

```
# cd /usr/ports/x11/kde4
# make install clean
```

Miután a KDE-t sikeresen telepítettük, tudatnunk kell az X szerverrel, hogy az alapértelmezett ablakkezelõ helyett ezt indítsa el. Ezt az .xinitrc állomány módosításával érhetjük el.

KDE3 esetén:

```
% echo "exec startkde" > ~/.xinitrc
```

KDE4 esetén:

```
% echo "exec /usr/local/kde4/bin/startkde" > ~/.xinitrc
```

Mostantól pedig mindig KDE lesz az asztalunk, amikor az X Window Systemet elindítjuk a `startx` paranccsal.

Ha az XDM-et használjuk bejelentkeztetõ képernyõként, a beállítást némileg máshogyan kell elvégeznünk. Ekkor az iménti helyett az .xsession állományt kell szerkesztenünk. A kdm-re vonatkozó utasítások a fejezet késõbbi részében találhatóak meg.

5.7.3. A KDE-rõl bõvebben

Most, miután telepítettük a KDE-t a rendszerünkre, a dolgok többsége felfedezhetõ a különféle súgók segítségével vagy egyszerûen a menükre történõ kattintással. A Windows®-hoz vagy Mac®-hez szokott felhasználók itt most már egészen otthonosan érezhetik magukat.

A KDE-hez a legtöbb segítséget a saját internetes dokumentációjából nyerhetjük. A KDE a saját böngészõjét, a Konquerort tartalmazza, valamint tucatnyi ügyes alkalmazást és temérdek mennyiségû dokumentációt. A szakasz további részeiben ezért inkább olyan problémákkal foglalkozunk, amelyek megoldásai céltalan kóborlással már nem fedezhetõek fel olyan egyszerûen.

A KDE bejelentkeztetõ képernyõje

Egy többfelhasználós rendszer karbantartója minden bizonnyal szeretné üdvözölni rendszere felhasználóit egy grafikus bejelentkezõ képernyõn keresztül. A korábbiakban erre a célra az XDM-et javasoltuk. Azonban a KDE erre ajánl egy alternatívát, a kdm-et, amely jóval látványosabb és sokoldalúbb. Ez különösen abban merül ki, hogy a felhasználók (egy menün keresztül) ki tudják választani a bejelentkezés után használni kívánt munkakörnyezetet (legyen az KDE, GNOME vagy bármi más).

A kdm használatához a KDE aktuális verziójától függõen különbözõ állományokat kell szerkesztenünk.

KDE3 esetén a /etc/ttys állományban szereplõ `ttyv8` sort kell az alábbiak szerint módosítanunk:

```
ttyv8 "/usr/local/bin/kdm -nodaemon" xterm on secure
```

KDE4 esetén a következõ sorokat kell felvennünk az /etc/rc.conf állományba:

```
local_startup="${local_startup} /usr/local/kde4/etc/rc.d"
kdm4_enable="YES"
```

5.7.4. Az Xfce

Röviden az Xfce-rõl

Az Xfce a GNOME által használt GTK+-ra épülõ munkakörnyezet, amely azonban sokkal könnyedebb és azoknak készült, akik egy szimpla, hatékony, mindazonáltal könnyen használható és beállítható munkafelületre vágynak. Látvány szempontjából leginkább a kereskedelmi rendszereken megtalálható CDE-hez hasonlítható. Íme az Xfce néhány jellemzõje:

- Egyszerû, könnyen kezelhetõ munkaasztal
- Tökéletesen konfigurálható egérrel, drag-and-droppal ("vonszolás") stb.

- A menükkel, kisalkalmazásokkal és alkalmazásindítókkal tarkított fõpanelje hasonló a CDE paneljéhez
- Beépített ablak-, állomány- és hangkezelõvel, GNOME kompatibilitási modullal és még sok minden mással rendelkezik
- Használhatunk témákat (mivel GTK+-ra épül)
- Gyors, könnyû és hatékony: ideális régebbi vagy lassabb, esetleg kevés memóriával rendelkezõ számítógépekhez

Az Xfce-rõl részletesebben az Xfce honlapján olvashatunk.

Az Xfce telepítése

Az Xfce-hez tartozik bináris csomag (legalábbis az leírás készítésének pillanatában). Ezt a következõ módon tudjuk telepíteni:

```
# pkg_add -r xfce4
```

Vagy a Portgyûjtemény használatával forrásból is felrakhatjuk:

```
# cd /usr/ports/x11-wm/xfce4
# make install clean
```

Ezután világosítsuk fel az X szervert, hogy a következõ indulása során mi már az Xfce-t kívánjuk használni. Ehhez csak ennyit kell tennünk:

```
% echo "/usr/local/bin/startxfce4" > ~/.xinitrc
```

Így az X következõ indításakor már az Xfce lesz a munkakörnyezetünk. Ahogy azt már korábban is jeleztük, az XDM használata során a GNOMEban leírtak szerint létre kell hoznunk az .xsession állományt, azonban ezúttal a /usr/local/bin/startxfce4 parancs használatával. Vagy a kdm-rõl szóló szakaszban tárgyaltak mentén beállíthatjuk úgy a bejelentkeztetõ képernyõt, hogy a bejelentkezés elõtt válasszuk ki a munkakörnyezetet.

Part II: Gyakori feladatok

Miután az alapokat már átvettük, a FreeBSD kézikönyv következõ része néhány gyakorta alkalmazott funkciót tárgyal. Az itt szereplõ fejezetek:

- Bemutatnak különféle hasznos és népszerû asztali alkalmazásokat: böngészõket, irodai elõsegítõ eszközöket, dokumentum-megjelenítõket stb.
- Bemutatják a FreeBSD alatt is elérhetõ multimédia eszközöket.
- Kifejtik egy saját FreeBSD rendszermag elkészítésének folyamatát, amellyel így bõvíteni tudjuk rendszerünk funkcionalitását.
- Részletesen bemutatják a nyomtatásért felelõs alrendszert, asztali és hálózati nyomtatók használata esetén egyaránt.
- Megmutatják, hogyan futassunk Linuxra íródott alkalmazásokat a FreeBSD rendszerünkön.

Egyes fejezetek elolvasásához ajánlott bizonyos mértékû felkészülés, amely megemlítésre is kerül az érintett fejezetek áttekintésében.

Chapter 6. II. Rész Gyakori feladatok

6.1. Áttekintés

A FreeBSD-n asztali alkalmazások széles spektrumát lehet futtatni, például böngészõket és szövegszerkesztõket. Legtöbbjük csomagként áll rendelkezésre, illetve automatizált módon lefordíthatóak a Portgyûjteményból. Az új felhasználók közül sokan szeretnének ilyen fajta alkalmazásokat használni, ezért ez a fejezet bemutatja, miként lehet a népszerûbb asztali alkalmazásokat minden különösebb erõfeszítés nélkül telepíteni, legyen szó az elõre csomagolt vagy a Portgyûjteményben megtalálható formájukról.

Amikor portként telepítünk egy programot, lényegében a forráskódját fordítjuk le. Ez bizonyos esetekben nagyon sokáig is eltarthat attól függõen, hogy pontosan mit is fordítunk le, illetve mekkora az erre a célra felhasznált számítógépünk vagy számítógépeink teljesítménye. Amennyiben a fordításra nem tudunk vagy nem kívánunk elegendõ idõt szánni, a Portgyûjteményben található programok többségét már elõre lefordított csomagból is telepíthetjük.

Mivel a FreeBSD-ben bináris szintû Linux kompatibilitás is található, ezért az eredetileg Linuxra fejlesztett alkalmazások is használhatóak a munkakörnyezetünkben. Azonban határozottan javasoljuk, hogy a linuxos alkalmazások használatához elõször figyelmesen olvassuk át a Bináris Linux kompatibilitáset. A linuxos bináris kompabilitást használó portok neve általában a "linux-" elõtaggal kezdõdik, amit ne felejtsük el figyelembe venni, amikor például a whereis(1) segítségével keressük valamelyiket. A fejezet további részében feltételezzük, hogy a linuxos alkalmazások telepítése elõtt aktiváltuk a bináris Linux kompatibilitást.

Íme a fejezetben tárgyalt kategóriák:

- Böngészõk (mint a Firefox, Opera, Konqueror)
- Irodai eszközök (mint a KOffice, AbiWord, The GIMP, OpenOffice.org)
- Dokumentum-megjelenítõk (mint az Acrobat Reader®, gv, Xpdf, GQview)
- Pénzügyi szoftverek (mint a GnuCash, Gnumeric, Abacus)

A fejezet elolvasásához ajánlott:

- a külsõ alkalmazások telepítésének ismerete (Alkalmazások telepítése. csomagok és portok);
- linuxos alkalmazások telepítésének ismerete (Bináris Linux kompatibilitás).

a multimédiás környezet kialakítására vonatkozó információkért a Multimédiaet érdemes elolvasni. Az elektronikus levelezés beállítását és használatát a Elektronikus levelezésbõl tudhatjuk meg.

6.2. Böngészõk

A FreeBSD-vel együtt nem települ semmilyen böngészõ. Helyette keressük meg a Portgyûjteményben a www könyvtárat, ahol ezzel szemben rengeteg böngészõ áll telepítésre készen. Ha nem lenne idõnk mindent lefordítani (ami egyes esetekben akár rengeteg idõnkbe is

kerülhet), ezek csomagolt formában is elérhetõek.

A KDE-hez és a GNOME-hoz eleve tartoznak HTML-böngészõk. Ezen komplett munkakörnyezetek beállításához a Munkakörnyezetekt olvassuk el.

Ha viszont csak egy kevés erõforrást igénylõ böngészõkre vágyunk, érdemes megnéznünk a Portgyûjteményben található www/dillo2, www/links vagy www/w3m portokat.

Ez a rész az alábbi alkalmazásokat említi:

Alkalmazás	Erõforrásigény	Telepítés forrásból	Fõbb függõségek
Firefox	közepes	nehéz	Gtk+
Opera	kevés	könnyû	Vannak FreeBSD-s és linuxos változatai is. A linuxos verzió használatához azonban szükség van a bináris Linux kompatibilitásra és a linux-openmotif portra.
Konqueror	közepes	nehéz	A KDE függvénykönyvtárai.

6.2.1. Firefox

A Firefox egy modern, szabad és nyílt forráskódú böngészõ, amely tökéletesen használható FreeBSD alatt. Megtalálható benne egy, a jelenlegi HTML szabványoknak nagyon jól megfelelõ megjelenítõ motor, a lapokra bontható böngészés támogatása, a kéretlenül felbukkanó ablakok blokkolása, különbözõ kiterjesztések, javított biztonsági lehetõségek és még sok minden más. A Firefox forrása a Mozilla kódján alapszik.

Csomagból így telepíthetõ:

```
# pkg_add -r firefox
```

Ekkor a Firefox 3.6 változata fog települni. Ha helyette a Firefox 3.5 változatát szeretnénk használni, akkor ezt a parancsot adjuk ki:

```
# pkg_add -r firefox35
```

Ha forrásból szeretnénk felrakni, használhatjuk a Portgyûjteményben található portját is:

```
# cd /usr/ports/www/firefox
# make install clean
```

A Firefox 3.5 telepítéséhez az iménti parancsban cseréljük ki a `firefox` részt a `firefox35` könyvtárra.

6.2.2. A Firefox és a Java™ plugin

Ennél és a következõ résznél feltételezzük, hogy már korábban telepítettük a Firefox alkalmazást.

A Java™ plugin jelenleg nem mûködik a Firefox 3.6 változatával.

A FreeBSD Alapítvány megegyezett a Sun Microsystems-szel, hogy terjesztheti a Java™ futtatókörnyezet (JRE™) és a Java™ fejlesztõkörnyezet (JDK™) FreeBSD-re lefordított bináris változatait. Ezek a csomagok elérhetõek a FreeBSD Alapítvány honlapjáról.

Ha tehát Java™-támogatást szeretnénk hozzáadni a Firefox böngészõhöz, elsõként fel kell telepítenünk a java/javavmwrapper portot. Ezután le kell töltenünk a Diablo JRE™ csomagot a http://www.freebsdfoundation.org/downloads/java.shtml címrõl, majd telepítenünk azt a pkg_add(1) segítségével.

Ezen az oldalon nem találunk bináris csomagokat FreeBSD 8.*X* rendszerekhez, azonban a FreeBSD 7.*X* rendszerekhez készült csomagok használhatóak 8.*X* esetén is. Ehhez mindössze a misc/compat7x portot kell elõtte telepítenünk.

A másik lehetõség a Diablo JRE™ (valamint a Diablo JDK™) telepítése a Portgyûjteményböl (java/diablo-jre16 és java/diablo-jdk16). Ehhez a szükséges forrásokat (distfile állományokat) licencelési megkötések miatt nekünk kell külön letölteni. A kapcsolódó utasításokat a `make maketarget` parancs meghívásával kaphatjuk meg.

Indítsuk el a böngészõnket, és írjuk be a címsorba, hogy `about:plugins` és nyomjuk le az `Enter` billentyût. Az eredményül kapott oldalon láthatjuk az eddig telepített pluginok listáját, ahol mostanra már a Java™ pluginnak is meg kell jelennie. Amennyiben ez nem következne be, mindegyik felhasználónál adjuk ki az alábbi parancsot:

```
% ln -s /usr/local/diablo-jre1.6.0/plugin/i386/ns7/libjavaplugin_oji.so \
  $HOME/.mozilla/plugins/
```

Vagy ha a Diablo JDK™ csomagot telepítettük:

```
% ln -s /usr/local/diablo-jdk1.6.0/jre/plugin/i386/ns7/libjavaplugin_oji.so \
  $HOME/.mozilla/plugins/
```

Ezt követõen indítsuk újra a böngészõnket.

Ezek a parancsok az i386 architektúra használatát feltételezik, de a csomagok az amd64 architektúra esetén is elérhetõek.

6.2.3. A Firefox és a Macromedia® Flash™ plugin

A Macromedia® Flash™ plugin nem érhetõ el közvetlenül FreeBSD-re. Azonban létezik egy, a plugin linuxos verziójára épített szoftveres réteg (wrapper). Ez a wrapper még többek közt az Adobe® Acrobat® és a RealPlayer® pluginjait is használhatóvá teszi.

Attól függõen, hogy a FreeBSD melyik változatát használjuk, különbözõ módokon tudjuk mûködésbe hozni:

1. FreeBSD 7.*X*

   Telepítsük a www/nspluginwrapper portot. Ehhez szükség lesz még az emulators/linux_base-fc4 portra is, amely viszonylag nagy méretû.

   Következõ lépésként telepítsük a www/linux-flashplugin9 portot. Ezáltal megkapjuk a Flash™ 9.*X* változatát, amely FreeBSD 7.*X* alatt remekül mûködik.

   A 7.1-RELEASE elõtti FreeBSD verziók esetén a www/linux-flashplugin7 portot telepítsük és hagyjuk ki a linprocfs(5) használatára vonatkozó részt.

2. FreeBSD 8.*X*

   Telepítsük a www/nspluginwrapper portot. Ehhez szükségünk lesz majd a valamivel nagyobb emulators/linux_base-f10 portra.

   Ezt követõen telepítsük a www/linux-f10-flashplugin10 portot. Ekkor a Flash™ 10.*X* változatát kapjuk, amely FreeBSD 8.*X* alatt remekül használható.

   Ezen változat beüzemeléséhez még létre kell hoznunk az alábbi linket:

   ```
   # ln -s /usr/local/lib/npapi/linux-f10-flashplugin/libflashplayer.so \
     /usr/local/lib/browser_plugins/
   ```

Miután a FreeBSD rendszerünk változatának megfelelõen elvégeztük a Flash™ port telepítését, a plugint az egyes felhasználóknak a `nspluginwrapper` paranccsal tehetjük elérhetõvé:

```
% nspluginwrapper -v -a -i
```

Ha Flash™ animációkat szeretnénk lejátszani, akkor ehhez a /usr/compat/linux/proc könyvtárba csatlakoztatnunk kell egy linprocfs(5) típusú linuxos proc állományrendszert. Ezt a következõ paranccsal tehetjük meg:

```
# mount -t linprocfs linprocfs /usr/compat/linux/proc
```

Az /etc/fstab állományban az alábbi sor hozzáadásával azonban ennek csatlakoztatása akár

automatikussá is tehetõ a rendszerindítás során:

```
linprocfs  /usr/compat/linux/proc  linprocfs      rw       0       0
```

Ezután indítsuk el a böngészõt, majd gépeljük be az `about:plugins` szöveget a címsorba és nyomjuk le az `Enter` billentyût. Ekkor a jelenleg elérhetõ pluginok listájának kell megjelennie.

6.2.4. A Firefox és az Swfdec Flash™ plugin

Az Swfdec egy Flash™ animációk dekódolásáért és megjelenítéséért felelõs programkönyvtár. Az Swfdec-Mozilla pedig egy Firefox böngészõkhöz készített plugin, amely az Swfdec könyvtáron keresztül játszik le SWF állományokat. Jelenleg még aktív fejlesztés alatt áll.

Ha nem akarjuk vagy netalán nem tudjuk forrásból lefordítani, akkor egyszerûen csak telepítsük csomagként a hálózaton keresztül:

```
# pkg_add -r swfdec-plugin
```

Ha valamiért mégsem érhetõ el hozzá csomag, akkor a Portgyûjteménybõl is telepíthetjük:

```
# cd /usr/ports/www/swfdec-plugin
# make install clean
```

Miután telepítettük a plugint, a használatához indítsuk újra a böngészõt.

6.2.5. Opera

Az Opera egy sokoldalú és szabványokkal kompatibilis böngészõ. Tartalmaz beépített levelezõ klienst és hírolvasót, IRC-klienst, RSS/Atom-olvasót és még sok mindent mást. Ennek ellenére az Opera viszonylag pehelysúlyúnak és gyorsnak számít. Két fajta módon is használható: létezik "natív" FreeBSD-s változata, valamint a Linux emulációval futó változata.

Az Opera FreeBSD-s változatát a megfelelõ csomag telepítésével érhetjük el:

```
# pkg_add -r opera
```

Habár egyes FTP oldalakon nem található meg az összes csomag, viszont a Portgyûjteménybõl még ekkor is be tudjuk szerezni az Operat:

```
# cd /usr/ports/www/opera
# make install clean
```

A linuxos Opera telepítéséhez `opera` helyett `linux-opera` nevet kell megadnunk a fenti parancsokban. Ennek a verziónak a használata akkor lehet elõnyös, ha olyan plugineket akarunk

elérni, amelyek csak Linuxra léteznek. Ilyen például az Adobe Acrobat Reader®. Ettõl eltekintve azonban a FreeBSD-s és a linuxos változatok szinte teljesen megegyeznek.

6.2.6. Konqueror

A Konqueror a KDE része, de a használatához elegendõ, ha csak a x11/kdebase3 portot telepítjük fel. A Konqueror több, mint egy egyszerû böngészõ: állománykezelõ és multimédiás nézegetõ is.

Számtalan plugin áll rendelkezésre a Konquerorhoz, melyeket a misc/konq-plugins portban találunk meg.

A Konqueror ismeri a Flash™t is. A Flash™ és a Konqueror kapcsolatával egy külön "Hogyan" is foglalkozik, amelyet a http://freebsd.kde.org/howtos/konqueror-flash.php címen olvashatunk el.

6.3. Irodai eszközök

Amikor irodai felhasználásról van szó, az új felhasználók gyakorta keresnek egy jó irodai programcsomagot vagy egy barátságos szövegszerkesztõt. Habár az egyes munkakörnyezetek, mint például a KDE, gyakran saját irodai eszközöket is tartalmaznak, FreeBSD alatt nincs alapértelmezett irodai programcsomag. A rendszer a munkakörnyezetektõl függetlenül igyekszik felkínálni mindazt, amire szükségünk lehet.

Ebben a részben a következõ alkalmazásokról esik szó:

Alkalmazás	Erõforrásigény	Telepítés forrásból	Fõbb függõségek
KOffice	kevés	nehéz	KDE
AbiWord	kevés	könnyû	Gtk+ vagy GNOME
The Gimp	kevés	nehéz	Gtk+
OpenOffice.org	sok	nagyon nehéz	JDK™, Mozilla

6.3.1. KOffice

A KDE közösség által kiadott munkakörnyezethez társul egy irodai programcsomag is, amely a KDE-tõl függetlenül is használható. Tartalmazza a többi irodai programcsomagban is megtalálható négy szabványos komponenst: a KWord szövegszerkesztõt, a KSpread táblazatkezelõt, a KPresenter prezentációkészítõt és végezetül a Kontourt, mellyel grafikus dokumentumokat tudunk elkészíteni.

A legfrissebb KOffice telepítése elõtt bizonyosodjuk meg róla, hogy a KDE legfrissebb verziójával is rendelkezünk.

Ha a KOffice-t csomagként akarjuk telepíteni, akkor adjuk ki az alábbi parancsot:

```
# pkg_add -r koffice
```

Amennyiben ez a csomag nem érhetõ el, telepíthetjük a Portgyûjteményböl is. Például a KDE3-hoz tartozó KOffice-t így rakhatjuk fel:

```
# cd /usr/ports/editors/koffice-kde3
# make install clean
```

6.3.2. AbiWord

Az AbiWord egy szabad szövegszerkesztõ program, a Microsoft® Word-höz hasonló kinézettel. Remekül használható levelek, beszámolók, feljegyzések, cikkek stb. írásához. Nagyon gyors, rengeteg funkciót ajánl fel, és kifejezetten felhasználóbarát.

Az AbiWord képes többféle állományformátumba exportálni és onnan importálni, beleértve az olyan zárt formátumokat is, mint például a Microsoft® .doc.

Az AbiWord csomagból telepíthetõ a következõ módon:

```
# pkg_add -r abiword
```

Amennyiben ez a csomag nem érhetõ el, lefordítható a Portgyûjteménybõl is, ami ráadásul sokszor egy frissebb verziót tartalmaz. Ezt így tudjuk megtenni:

```
# cd /usr/ports/editors/abiword
# make install clean
```

6.3.3. The GIMP

Képek készítésére vagy retusálásra a The GIMP a legfejlettebb képszerkesztõ program. Egyszerû rajzolóprogram gyanánt is használható, de akár minõségi fényképretusálásra is. Óriási mennyiségû plugin található hozzá és magában foglal egy szkriptes interfészt is. A The GIMP formátumok széles skáláját ismeri. Számos scanner és digitális rajztábla csatlakoztatható hozzá.

A hozzá tartozó csomag a következõ módon telepíthetõ fel:

```
# pkg_add -r gimp
```

Ha a csomagoknak beállított FTP oldalon nem található meg ez a csomag, megpróbálkozhatunk vele a Portgyûjteményen keresztül is. A gyûjtemény graphics könyvtárában ezen felül fellelhetjük a The Gimp Manualt, vagyis a The GIMP kézikönyvét. Így kell ezeket innen telepíteni:

```
# cd /usr/ports/graphics/gimp
# make install clean
# cd /usr/ports/graphics/gimp-manual-pdf
# make install clean
```

A Portgyûjtemény graphics könyvtárában a The GIMP fejlesztõi változatával is találkozhatunk a graphics/gimp-devel alkönyvtárban. A The Gimp Manual HTML

változata pedig a graphics/gimp-manual-html alkönyvtárban található.

6.3.4. OpenOffice.org

Az OpenOffice.org tartalmaz minden olyan elengedhetetlenül fontos alkalmazást, amelyek napjaink bármelyik irodájához hozzátartoznak: egy szövegszerkesztõt, egy táblázatkezelõt, egy prezentációszerkesztõt és egy rajzolóprogramot. A felhasználói felülete nagyon hasonlít a többi irodai programcsomagéhoz, és képes többféle elterjedt állományformátumot kezelni. Számos különbözõ nyelven elérhetõ - a honosítása kiterjed a felületekre, helyesírás-ellenõrzõkre és szótárakra is.

Az OpenOffice.org szövegszerkesztõje natív XML állományformátumot használ a hordozhatóság és a rugalmasság növeléséhez. A táblázatkezelõje tartalmaz egy makrónyelvet és könnyedén összekapcsolható külsõ adatbázisokkal. Az OpenOffice.org natívan és megbízhatóan fut Windows®-on, Solaris™-on, Linux®-on, FreeBSD-n és Mac OS® X-en. Az OpenOffice.org-ról bõvebb információt a projekt saját honlapján találhatunk. A FreeBSD-s változatra vonatkozó információkat és a csomagokat pedig a FreeBSD OpenOffice.org Porting Team honlapján lelhetjük meg.

Az OpenOffice.org telepítéséhez ennyit kell csak beírni:

```
# pkg_add -r openoffice.org
```

Ha a FreeBSD -RELEASE ágát használjuk, ennek mûködnie kell. Ettõl eltérõ esetben érdemes egy pillantást vetni a FreeBSD OpenOffice.org Porting Team honlapjára, ahonnan le tudjuk tölteni a verziókhoz megfelelõ csomagot, amelyet ezután a pkg_add(1)-al fel is tudunk telepíteni. A legfrissebb megbízható és a fejlesztõi változat egyaránt elérhetõ errõl a helyrõl.

Ahogy sikerült feltelepíteni a csomagot, egyszerûen csak be kell gépelni a következõ parancsot az OpenOffice.org futtatásához:

```
% openoffice.org
```

Az elsõ futtatás során válaszolnunk kell még néhány további kérdésre is, valamint a felhasználói könyvtárunkban keletkezik egy .openoffice.org könyvtár.

Ha nem érhetõek el OpenOffice.org csomagok, lefordíthatjuk a forrását is. Azonban mielõtt még ennek nekilátnánk, el kell fogadnunk, hogy ez a mûvelet a lemezünkön rettenetesen sok területet fog igényelni és meglehetõsen sokáig tart.

```
# cd /usr/ports/editors/openoffice.org-3
# make install clean
```

Ha egy honosított verziót szeretnénk fordítani, az utolsó parancs helyett írjuk inkább ezt:

```
# make LOCALIZED_LANG=nyelv install clean
```

A *nyelv* helyett itt természetesen a nyelvnek megfelelõ ISO-kódot kell megadni. Az itt támogatott nyelvek kódjának listája a port könyvtárán belül, a files/Makefile.localized állományban található meg.

Ahogy a fordítás befejezõdött, az OpenOffice.org így indítható el parancssorból:

```
% openoffice.org
```

6.4. Dokumentum-megjelenítõk

A UNIX® megjelenése óta néhány új népszerû dokumentumformátum is felbukkant, melyek szabványos megjelenítõi nem minden esetben részei az alaprendszernek. Ebben a részben azt tekintjük át, hogyan lehet ilyen megjelenítõket telepíteni.

Ez a rész az alábbi alkalmazásokat említi:

Alkalmazás	Erõforrásigény	Telepítés forrásból	Fõbb függõségek
Acrobat Reader®	kevés	könnyû	Bináris Linux kompatibilitás
gv	kevés	könnyû	Xaw3d
Xpdf	kevés	könnyû	FreeType
GQview	kevés	könnyû	Gtk+ vagy GNOME

6.4.1. Acrobat Reader®

A dokumentumok többsége manapság PDF (Portable Document Format, avagy "hordozható dokumentumformátum)" állományok formájában terjed. Az ilyen típusú állományok megnézésére az egyik legmegfelelõbb alkalmazás az Acrobat Reader®, melyet az Adobe adott ki Linuxra. De mivel a FreeBSD képes Linux binárisok futtatására, ezért így FreeBSD-re is elérhetõ.

Ha az Acrobat Reader® 8-at a Portgyûjteménybõl akarjuk telepíteni, akkor írjuk be:

```
# cd /usr/ports/print/acroread8
# make install clean
```

Licencelési megszorítások miatt csomag nem áll rendelkezésre.

6.4.2. gv

A gv egy PostScript® és PDF megjelenítõ. Eredetileg a ghostview alapján készült, de a Xaw3d-nek köszönhetõen sokkal szebben néz ki. Gyors és a felülete letisztult. A gv sok mindent tud, többek közt

beállítható benne a dokumentum tájolása, a papírméret, skálázás és az élsimítás. Szinte bármelyik mûvelet elvégezhetõ csak billentyûzetrõl vagy egérrel.

A gv csomagjának telepítéséhez a következõ parancsot használhatjuk:

```
# pkg_add -r gv
```

Ha pedig nem tudjuk letölteni a csomagot, használhatjuk a Portgyûjteményt is:

```
# cd /usr/ports/print/gv
# make install clean
```

6.4.3. Xpdf

Ha egy egyszerû FreeBSD-s PDF megjelenítõre lenne szükségünk, erre a célra az Xpdf pontosan megfelel. Nagyon kevés erõforrást igényel és nagyon megbízható. A szabványos X-beli betûtípusokat használja, és nincs szüksége sem a Motif®ra, sem pedig más X-es eszközkészletre.

Az Xpdf csomagjának felrakásához az alábbi parancs javasolt:

```
# pkg_add -r xpdf
```

Amennyiben nem áll rendelkezésre az említett csomag, vagy egyszerûen csak a Portgyûjteménybõl szeretnénk felrakni, adjuk ki ezeket a parancsokat:

```
# cd /usr/ports/graphics/xpdf
# make install clean
```

Ahogy a telepítés befejezõdik, már el is indíthatjuk az Xpdf alkalmazást, ahol a jobb egérgombbal tudjuk aktiválni a menüt.

6.4.4. GQview

A GQview egy képkezelõ. Állományokat tudunk megnyitni benne egyetlen kattintással, külsõ szerkesztõprogramot tudunk indítani vagy akár még a képek kicsinyített változatait is láthatjuk és így tovább. Megtalálható benne a diavetítés és az alapvetõ állománymûveletek. Képgyûjteményeket is kezelhetünk és könnyedén megtalálhatjuk a bennük levõ képek között az egyezõeket. A GQview teljes képernyõs nézegetést is megenged, illetve támogatja a honosítást.

A GQview csomag telepítéséhez ezt a parancsot kell kiadni:

```
# pkg_add -r gqview
```

Amikor ez a csomag nem tölthetõ le, vagy amikor inkább a Portgyûjteménybõl szeretnénk felrakni,

ezt írjuk be:

```
# cd /usr/ports/graphics/gqview
# make install clean
```

6.5. Pénzügyi szoftverek

Ha bármilyen ok folytán a FreeBSD-vel szeretnénk kezeli személyes pénzügyeinket, akadnak olyan kellõen komoly és könnyen kezelhetõ alkalmazások, amelyek csak a telepítésükre várnak. Néhány közülük kompatibilis az elterjedtebb állományformátumokkal, mint például amiben a Quicken és az Excel is tárolja az adatait.

Ebben a részben az alábbi programokat vesszük sorra:

Alkalmazás	Erõforrásigény	Telepítés forrásból	Fõbb függõségek
GnuCash	kevés	nehéz	GNOME
Gnumeric	kevés	nehéz	GNOME
Abacus	kevés	könnyû	Tcl/Tk
KMyMoney	kevés	nehéz	KDE

6.5.1. GnuCash

A GnuCash a GNOME része, és egy felhasználóbarát, mégis hatékony eszközt ad a felhasználók kezébe. A GnuCash segítségével nyilván tudjuk tartani a bevételeinket és kiadásainkat, bankszámláinkat és befektetéseinket. Felülete intuitív, miközben továbbra is professzionális minõségû.

A GnuCash-ben megtalálhatunk egy intelligens nyilvántartást, a számlák hierarchikus rendszerét, és számtalan billentyûkombinációt és automatikus kiegészítést, amivel felgyorsul a munkánk. Egyetlen tranzakciót képes felbontani több kisebb és részletesebb elemre. A GnuCash képes importálni és exportálni a Quicken QIF típusú állományait. Ezenkívül még kezeli a legtöbb nemzetközi dátumformátumot és pénznemet.

A GnuCash-t az alábbi módon tudjuk telepíteni a rendszerünkre:

```
# pkg_add -r gnucash
```

Ha ez a csomag nem érhetõ el, használhatjuk a Portgyûjteményt is:

```
# cd /usr/ports/finance/gnucash
# make install clean
```

6.5.2. Gnumeric

A Gnumeric egy táblázatkezelõ program, a GNOME munkakörnyezet része. Sok esetben képes a helyzethez alkalmazkodva automatikusan "kitalálni" a felhasználó gondolatait a cellák formátumának megfelelõ automatikus kiegészítõ rendszerével. Be tud olvasni számos népszerûbb formátumot, mint például az Excel, Lotus 1-2-3 vagy a Quattro Pro állományait. A math/guppi grafikonkészítõ programon keresztül támogatja grafikonok rajzolását is. Nagyszámú beépített funkcióval rendelkezik, és ismeri az összes megszokott cellaformátumot, legyen az szám, pénznem, dátum, idõ vagy bármi más.

A Gnumeric telepítését az alábbi paranccsal adhatjuk ki:

```
# pkg_add -r gnumeric
```

Ha valamiért nem érhetõ el ez a csomag, a Portgyûjteménybõl is fel tudjuk rakni:

```
# cd /usr/ports/math/gnumeric
# make install clean
```

6.5.3. Abacus

Az Abacus egy kicsi és egyszerûen használható táblázatkezelõ program. Számos olyan funkciót tartalmaz beépítve, amelyek kifejezetten hasznosnak bizonyulhatnak a statisztika, pénzügyek és a matematika területén. Importálni és exportálni tudja az Excel állományformátumát is. Az Abacus még PostScript® formátumú kimenetet is tud készíteni.

Az Abacus telepítéséhez csupán ennyit kell tennünk:

```
# pkg_add -r abacus
```

Amennyiben viszont nem érhetõ el ez a csomag, használhatjuk a Portgyûjteményt is:

```
# cd /usr/ports/deskutils/abacus
# make install clean
```

6.5.4. KMyMoney

A KMyMoney a KDE részeként kifejlesztett személyi pénzügyi nyilvántartó. A KMyMoney igyekszik az összes kereskedelmi pénzügyi nyilvántartó programban megtalálható fontosabb lehetõséget magában foglalni és rendelkezésre bocsátani. Mindezek mellett egy könnyen használható és nagyon ügyes kettõs könyvelést is találhatunk benne. A KMyMoney képes beolvasni a szabványos Quicken Interchange Format (QIF) szerint készült állományokat, követni a befektetéseket, többféle pénznemet kezelni és sokfajta kimutatást tudunk vele készíteni. A megfelelõ bõvítmény hozzáadásával még az OFX formátumú állományok olvasására is alkalmas.

A KMyMoney csomagként így telepíthetõ:

```
# pkg_add -r kmymoney2
```

Ha ez a csomag nem érhetõ el, akkor a Portgyûjteményen keresztül is fel tudjuk rakni:

```
# cd /usr/ports/finance/kmymoney2
# make install clean
```

6.6. Összefoglalás

Miközben a FreeBSD igen népszerû az internetszolgáltatók körében a teljesítménye és megbízhatósága révén, a hétköznapi használatban is remekül beválik. Többezernyi olyan alkalmazás érhetõ el hozzá csomagként vagy portként, amelyekkel az igényeinknek megfelelõ munkakörnyezetet tudjuk kiépíteni.

Íme egy rövidke emlékeztetõ azokról az asztali alkalmazásokról, melyeket a fejezetben tárgyaltunk:

Alkalmazás	Csomag	Port
Opera	opera	www/opera
Firefox	firefox	www/firefox
KOffice	koffice-kde3	editors/koffice-kde3
AbiWord	abiword	editors/abiword
The GIMP	gimp	graphics/gimp
OpenOffice.org	openoffice	editors/openoffice.org-3
Acrobat Reader®	acroread	print/acroread8
gv	gv	print/gv
Xpdf	xpdf	graphics/xpdf
GQview	gqview	graphics/gqview
GnuCash	gnucash	finance/gnucash
Gnumeric	gnumeric	math/gnumeric
Abacus	abacus	deskutils/abacus
KMyMoney	kmymoney2	finance/kmymoney2

Chapter 7. Multimédia

7.1. Áttekintés

A FreeBSD a hangkártyák széles választékát ismeri, ami által képesek vagyunk számítógépünkkel hi-fi minõségû hangzást létrehozni. Ennek részeként rögzíteni és visszajátszani tudunk többek közt MPEG Audio Layer 3 (MP3), WAV és Ogg Vorbis formátumokban. A FreeBSD Portgyûjteménye ezenkívül tartalmaz még olyan alkalmazásokat is, amelyekkel szerkeszteni lehet a felvett hangokat, effekteket hozzátenni és vezérelni a hangkártyánkhoz csatlakoztatott MIDI eszközöket.

Némi kísérletezéssel a FreeBSD még videoállományok és DVD-k lejátszására is rávehetõ. A különféle videoanyagok kódolására, konvertálására és visszajátszására alkalmas programok száma azonban jóval kisebb, mint a hanganyagok esetén. Például az írás pillanatában nincs a FreeBSD Portgyûjteményében a formátumok közti konvertálásra alkalmas, a videókat olyan jól újrakódolni tudó alkalmazás, amilyen az audio esetén az audio/sox. Azonban ezen a területen a szoftverek palettája gyorsan változik.

Ebben a fejezetben bemutatjuk a hangkártyánk beállításához szükséges lépéseket. Az X11 telepítése és beállítása (Az X Window System) során ugyan már foglalkoztunk a videokártyánkkal kapcsolatos hardveres problémákkal, azonban a jobb visszajátszás érdekében további cselfogásokat is be kell majd vetnünk.

A fejezet elolvasása során megismerjük:

- hogyan állítsuk be úgy a rendszerünket, hogy felismerje a hangkártyánkat;
- hogyan bizonyosodjuk meg róla, hogy a kártyánk valóban mûködik;
- hogyan oldjuk meg a hangkártya beállítása során felmerülõ problémákat;
- hogyan játsszunk le és kódoljunk MP3-at vagy más egyéb hangformátumot;
- hogyan támogatja a videokat az X szerver;
- hogyan adnak az egyes lejátszók és kódolók még jobb eredményt
- hogyan játsszunk le DVD-ket, .mpg és .avi állományokat;
- hogyan mentsük a CD-k és DVD-k tartalmát állományokba;
- hogyan állítsuk be a TV kártyánkat
- hogyan állítsunk be egy scannert.

A fejezet elolvasásához ajánlott:

- egy új rendszermag beállításának és telepítésének ismerete (A FreeBSD rendszermag testreszabása).

Ha zenei CD-ket próbálunk meg a mount(8) paranccsal csatlakoztatni, akkor az hibával, vagy a legrosszabb esetben akár *teljes rendszerösszeomlással* is járhat. Az ilyen típusú lemezek az ISO szabványú állományrendszerekétõl eltérõ kódolással rendelkeznek.

7.2. A hangkártya beállítása

7.2.1. A rendszer beállítása

A mûvelet megkezdése elõtt ki kell derítenünk, milyen típusú hangkártyánk van, milyen chip van rajta, PCI vagy ISA buszon csatlakozik-e. A FreeBSD rengeteg PCI és ISA buszos kártyát ismer egyaránt. A sajátunk beazonosításához a támogatott hangeszközök listáját a Hardware Notes (Hardverjegyzék) oldalán találhatjuk meg. Ebbõl a jegyzékbõl mellesleg azt is megtudhatjuk, hogy melyik meghajtó kezeli a kártyánkat.

A hangeszközünk használatához be kell töltenünk a neki megfelelõ meghajtót. Ez két módon is megtehetõ. Ezek közül az a legkönnyebb, ha a kldload(8) paranccsal egyszerûen betöltjük a rendszermag hangkártyánkhoz tartozó modulját. Ezt megtehetjük közvetlenül parancssorból:

```
# kldload snd_emu10k1
```

vagy a /boot/loader.conf állományból az alábbihoz hasonló sor hozzáadásával:

```
snd_emu10k1_load="YES"
```

A fenti példák a Creative SoundBlaster® Live! hangkártyára vonatkoznak. A többi betölthetõ hangkártya-modul felsorolása a /boot/defaults/loader.conf állományban található. Ha nem vagyunk benne biztosak, hogy melyik meghajtót is akarjuk pontosan használni, akkor próbálkozzunk az snd_driver modul betöltésével:

```
# kldload snd_driver
```

Ez egy olyan metameghajtó, ami egyszerre betölti az összes érintett eszközmeghajtót, és segítségével felgyorsíthatjuk a megfelelõ meghajtó megtalálását. A /boot/loader.conf használatával is be tudjuk ugyanígy tölteni az összes meghajtót.

Az snd_driver metameghajtó betöltése után úgy kereshetjük meg a ténylegesen használatban levõ meghajtót, ha megnézzük a /dev/sndstat állományt a `cat /dev/sndstat` paranccsal.

A második módszer szerint a hangkártyánk támogatását statikusan beépítjük a rendszermagba. A lentebb található szakaszban olvashatjuk mindazok az információkat, amelyekre szükségünk lehet ennek elvégzése közben. A rendszermag újrafordításával kapcsolatban forduljunk a A FreeBSD rendszermag testreszabásahez.

A hangkártya támogatásával rendelkezõ saját rendszermag összeállítása

Elsõként hozzá kell adnunk a rendszermaghoz a hangeszközök alapmeghajtóját, a sound(4) eszközt. Ezt a rendszermag beállításait tartalmazó állományban az alábbi sor felvételével tehetjük meg:

```
device sound
```

Ezután tegyük még hozzá a hangkártyánkhoz kapcsolódó támogatást is. Ehhez viszont pontosan tudunk kell, melyik meghajtó képes mûködtetni a kártyát. A hangkártyához tartozó meghajtót a Hardware Notes (Hardverjegyzék)-ben található eszközök listájából deríthetjük ki. Például a Creative SoundBlaster® Live! hangkártyát a snd_emu10k1(4) meghajtó kezeli. Ennek a hangkártyának a támogatását az alábbi sorral állíthatjuk be:

```
device snd_emu10k1
```

Az itt használatos formátumot a meghajtó man oldalának átolvasásából tudhatjuk meg. Azonban az összes támogatott hangkártya meghajtó megadásának pontos formátuma megtalálható a /usr/src/sys/conf/NOTES állományban is.

A PnP (Plug n Play)-t nem ismerõ ISA kártyák esetén az összes többi nem PnP-s ISA kártyához hasonlóan szükséges lehet a rendszermag számára megadnunk a kártya hardveres beállításait (IRQ, I/O port stb). Ezt a /boot/device.hints állományon keresztül tehetjük meg. A rendszerindítási folyamat során a loader(8) beolvassa ezt az állományt, majd átadja a benne szereplõ információkat a rendszermagnak. Például a Creative SoundBlaster® 16, nem PnP-s ISA kártya az snd_sb16 meghajtóval együtt az snd_sbc(4) meghajtót használja. A kártya használatához a rendszermag beállításait tartalmazó állományba ezeket a sorokat kell megadni:

```
device snd_sbc
device snd_sb16
```

valamint a /boot/device.hints állományba ezeket:

```
hint.sbc.0.at="isa"
hint.sbc.0.port="0x220"
hint.sbc.0.irq="5"
hint.sbc.0.drq="1"
hint.sbc.0.flags="0x15"
```

Ekkor a kártya a `0x220` I/O portot és `5` IRQ-t használja.

A /boot/device.hints állományban alkalmazott felírási módról bõvebben a sound(4), valamint a kérdéses meghajtó man oldalán tájékozódhatunk.

A fentiekben bemutatott beállítások alapértelmezettek, néhány esetben azonban a kártyánknak megfelelõen meg kell változtatnunk az IRQ és egyéb értékeket. Errõl a kártyáról konkrétan a snd_sbc(4) man oldalon olvashatunk részletesebben.

7.2.2. A hangkártya kipróbálása

Miután újraindítottuk a számítógépünket a módosított rendszermaggal, vagy miután betöltöttük a szükséges modult, a hangkártyának valahogy így kell megjelennie a rendszerünk üzenetpufferében (dmesg(8)):

```
pcm0: <Intel ICH3 (82801CA)> port 0xdc80-0xdcbf,0xd800-0xd8ff irq 5 at device 31.5 on
pci0
pcm0: [GIANT-LOCKED]
pcm0: <Cirrus Logic CS4205 AC97 Codec>
```

A hangkártyánk állapota a /dev/sndstat állományon keresztül ellenõrizhetõ:

```
# cat /dev/sndstat
FreeBSD Audio Driver (newpcm)
Installed devices:
pcm0: <Intel ICH3 (82801CA)> at io 0xd800, 0xdc80 irq 5 bufsz 16384
kld snd_ich (1p/2r/0v channels duplex default)
```

Ez a kiírás rendszerenként eltérhet. Ha nem látunk semmilyen pcm0 eszközt, akkor menjünk vissza és nézzük át újra, pontosan mit is csináltunk. Vizsgáljuk át a rendszermagunk beállításait tartalmazó állományt és gyõzõdjünk meg róla, hogy a megfelelõ meghajtót adtuk meg. Az itt felmerülõ gyakori gondokkal a Gyakori problémák foglalkozik.

Ha azonban minden remekül haladt, akkor most már van egy mûködõ hangkártyánk. Ha rendesen összekapcsoltuk hangkártyánkat a CD- vagy DVD-meghajtónk audio csatlakozásával, akkor tegyünk egy CD-t a meghajtóba és kezdjük el játszani a cdcontrol(1) paranccsal:

```
% cdcontrol -f /dev/acd0 play 1
```

Az olyan alkalmazások, mint például az audio/workman, ehhez egy sokkal barátságosabb felületet nyújtanak. Az MP3 formátumú állományok meghallgatásához pedig minden bizonnyal jól fog jönni egy olyan alkalmazás is, mint például az audio/mpg123.

A kártyát úgy is tesztelhetjük, ha az alábbihoz hasonló módon adatokat küldünk a /dev/dsp állományba:

```
% cat állománynév > /dev/dsp
```

ahol az *állománynév* tetszõleges állomány neve lehet. A parancs hatására valamilyen zajt kell hallanunk, és ez egyben meg is erõsíti, hogy a hangkártyánk mûködik.

A hangkártyánk csatornáinak jellemzõit a mixer(8) paranccsal állíthatjuk. Errõl további részleteket a mixer(8) man oldalon olvashatunk.

Gyakori problémák

Hiba	Megoldás
`sb_dspwr(XX) timed out`	Nem állítottuk be jól az I/O portot.

Hiba	Megoldás
`bad irq XX`	Nem állítottuk be jól az IRQ értékét. Gondoskodjunk róla, hogy a beállított érték megegyezik a hangkártyánkéval.
`xxx: gus pcm not attached, out of memory`	Nincs elég memória az eszköz használatához.
`xxx: can't open /dev/dsp!`	A `fstat | grep dsp` parancs kiadásával ellenõrizzük, hogy valamelyik alkalmazás használja-e már az eszközt. Gyakori bajkeverõ az esound és a KDE hangtámogatása.

7.2.3. Több hangforrás kihasználása

Gyakran szükségünk lehet több hangforrás egyidejû használatára, fõleg olyankor, amikor az esound vagy az artsd bizonyos alkalmazásokkal nem hajlandó megosztani a hangeszközt.

A FreeBSD ezt a *virtuális hangcsatornák* használatával oldja meg, amit a sysctl(8) eszközön keresztül tudunk engedélyezni. Amikor a rendszermagban virtuális csatornák használatával keverünk, akkor lényegében képesek vagyunk a hangkártyánk által egyszerre játszható hangok számát megtöbbszörözni.

A virtuális csatornák számának beállításához a sysctl három változóját kell módosítanunk, amelyet `root` felhasználóként így tehetünk meg:

```
# sysctl dev.pcm.0.play.vchans=4
# sysctl dev.pcm.0.rec.vchans=4
# sysctl hw.snd.maxautovchans=4
```

A fenti példa négy virtuális csatornát hoz létre, ami egészen jellemzõ a mindennapi használatban. A `dev.pcm.0.play.vchans` és `dev.pcm.0.rec.vchans` a pcm0 eszköz lejátszásra és felvételre használt virtuális csatornáinak számát adja meg, amelyet az eszköz csatlakoztatása után tudunk beállítani. A `hw.snd.maxautovchans` az új eszközhöz tartozó virtuális csatornákat adja meg, ami akkor állítódik be, amikor a kldload(8) paranccsal csatlakoztatjuk. Mivel a pcm modul a többi eszközmeghajtótól függetlenül töltõdik be, ezért a `hw.snd.maxautovchans` azt tárolja, hogy a késõbb hozzá csatlakozó eszközök mennyi virtuális csatornát fognak majd kapni. Errõl részletesebben a pcm(4) man oldalon olvashatunk.

A használatban levõ eszközöknél nem tudjuk megváltoztatni a virtuális csatornák számát. Ehhez elõször le kell állítanunk az eszközt használó összes programot, tehát a zenelejátszókat és hangdémonokat.

Amennyiben nem használjuk ki a devfs(5) által nyújtott lehetõségeket, az összes alkalmazásnak a /dev/dsp0.x eszközre kell mutatnia, ahol az *x* értéke 0-tól 3-ig terjedhet attól függõen, hogy a `dev.pcm.0.rec.vchans` értékét a fenti példához hasonlóan 4-re állítottuk-e. A devfs(5) megoldását használó rendszerek esetén ez a folyamat automatikusan lezajlik, tehát az összes /dev/dsp eszközre irányuló kérés magától átirányítódik.

7.2.4. A keverõ alapértelmezett értékeinek beállítása

A keverõben megjelenõ különbözõ csatornák alapértékei a pcm(4) meghajtó forráskódjában huzalozottan találhatóak meg. Számos alkalmazás és démon segít két hívás közt megõrizni a keverõben beállított értékeket, azonban ez nem teljesen tiszta megoldás. A meghajtó szintjén is be tudjuk állítani a keverõ alapértékeit - ezt a /boot/device.hints állomány megfelelõ módosításával érhetjük el, például:

```
hint.pcm.0.vol="50"
```

Ezzel a pcm(4) modul betöltése során a hangerõ (volume) csatorna alapértelmezett értéket 50-re állítjuk.

7.3. MP3

Az MP3 (MPEG Layer 3 Audio) használatával közel CD minõségû hangot lehet elérni, ezért a mi FreeBSD munkaállomásunk sem maradhat ki elõnyeinek élvezetébõl.

7.3.1. MP3 lejátszók

Az XMMS (X Multimedia System) kiemelkedõen a legnépszerûbb X11-es MP3 lejátszó. Mivel az XMMS grafikus felhasználói felülete szinte teljesen megegyezik a Nullsoft Winampjának felületével, ezért még a Winamp skinjeit is használhatjuk vele. Az XMMS-ben ezenkívül még a natív pluginek támogatását is megtalálhatjuk.

Az XMMS a multimedia/xmms portból vagy csomagból telepíthetõ.

Az XMMS használatára könnyû ráérezni: megtaláljuk benne a lejátszandó számok listáját, egy grafikus hangszínszabályzót és még sok minden mást. Akik már ismerik a Winamp mûködését, azok az XMMS-t is egyszerûnek érzik majd.

Mellette az audio/mpg123 port egy másik, parancssoros MP3 lejátszót kínál fel.

Az mpg123 futtatásához paraméterként meg kell adnunk a hangeszközt és lejátszandó MP3 állományt. Ha a hangeszközünk a /dev/dsp1.0 és a *IzéMizé-Sláger.mp3* nevû MP3 állományt akarjuk rajta lejátszatni, akkor a következõt kell begépelnünk:

```
# mpg123 -a /dev/dsp1.0 IzéMizé-Sláger.mp3
High Performance MPEG 1.0/2.0/2.5 Audio Player for Layer 1, 2 and 3.
Version 0.59r (1999/Jun/15). Written and copyrights by Michael Hipp.
Uses code from various people. See 'README' for more!
THIS SOFTWARE COMES WITH ABSOLUTELY NO WARRANTY! USE AT YOUR OWN RISK!

Playing MPEG stream from IzéMizé-Sláger.mp3 ...
MPEG 1.0 layer III, 128 kbit/s, 44100 Hz joint-stereo
```

7.3.2. Sávok lementése CD-rõl

Mielõtt MP3 formátumba tömörítenénk egy CD-t vagy annak egy sávját, a CD-n található audio adatot valahogy le kell tudnunk szedni a merevlemezre. Ezt úgy tehetjük meg, ha a nyers CDDA (CD Digital Audio) adatot WAV formátumú állományokba mentjük.

A sysutils/cdrtools csomag részeként elérhetõ `cdda2wav` segédprogrammal tudjuk a CD-ken levõ audio és a hozzájuk tartozó egyéb információkat leszedni.

A meghajtóban levõ CD teljes tartalmát (`root` felhasználóként) a következõ parancs kiadásával lehet (sávonként) különálló WAV állományokba menteni:

```
# cdda2wav -D 0,1,0 -B
```

A cdda2wav ismeri az ATAPI (IDE) CD-meghajtókat, használatukhoz a SCSI egység sorszáma helyett az eszköz nevét kell megadni. Tehát például így szedjük le egy IDE-meghajtóról a 7. sávot:

```
# cdda2wav -D /dev/acd0 -t 7
```

A `-D 0,1,0` a 0,1,0 sorszámú SCSI eszközre utal, ami megfelel `cdrecord -scanbus` parancs eredményének.

Az egyes sávok lementéséhez a `-t` kapcsoló használható:

```
# cdda2wav -D 0,1,0 -t 7
```

A példa szerint a zenei CD-rõl a hetedik sávot szedjük le. Egyszerre több sávot, például az elsõtõl a hetedikig, egy tartomány megadásával menthetünk le:

```
# cdda2wav -D 0,1,0 -t 1+7
```

A dd(1) segédprogram is használható ATAPI eszközökön levõ hangsávok kimentéséhez. Ennek lehetõségérõl részletesebben a Audio CD-k másolásaban olvashatunk.

7.3.3. MP3 állományok tömörítése

Az MP3 állomány tömörítésére manapság a legtöbben a lame elnevezésû kódolót választják. A portfában a lame az audio/lame helyen található meg.

Az elõbb kimentett WAV állományok felhasználásával az alábbi paranccsal tudjuk átalakítani a audio01.wav állományt audio01.mp3 állománnyá:

```
# lame -h -b 128 \
--tt "Izé dal címe" \
--ta "Izé-mizé előadó" \
```

```
--tl "Izé-mizé album" \
--ty "2001" \
--tc "Leszedte és tömörítette: Izé" \
--tg "Műfaj" \
audio01.wav audio01.mp3
```

A 128 kbites tömörítés a gyakorlatban leginkább használt kódolási arány, sokan azonban a sokkal jobb minõségû 160 vagy 192 kbites tömörítést szeretik. Minél nagyobb a kódolási arány, annál több helyet fog foglalni a keletkezõ MP3 állomány - habár a minõsége is jobb lesz. A `-h` kapcsoló alkalmazásával tudjuk aktivizálni a "jobb minõségû de valamivel lassabb" módot. A `--t` kezdetû paraméterek ID3 tageket adnak meg, amelyek segítségével az MP3 állományokba rájuk vonatkozó információkat tudunk beágyazni. A tömörítés további beállításairól a lame man oldalán tájékozódhatunk.

7.3.4. MP3 állományok kitömörítése

Ha MP3 formátumú állományokat szeretnénk audio CD-re írni, akkor ehhez elõször tömörítetlen WAV formátumba kell ezeket alakítanunk. Az XMMS és az mpg123 is egyaránt lehetõséged ad az MP3 állományok kitömörítésére.

Lemezre írás az XMMS-sel:

1. Indítsuk el az XMMS alkalmazást.
2. Az XMMS menüjének felhozásához kattinsunk jobb gombbal az ablakjára.
3. Válasszuk az `Options` almenüben található `Preference` menüpontot.
4. Változtassuk meg az "Output Plugin" beállítást a "Disk Writer Plugin" értékre.
5. Nyomjunk a `Configure` gombra.
6. Írjuk be (vagy válasszuk ki a `Browse` gombbal) a könyvtárat, ahová majd a kitömörített állományok kerülnek.
7. Az eddig megszokottak szerint töltsük be az XMMS-be az MP3 állományt, állítsuk 100%-ra a hangerõt és kapcsoljuk ki a hangszínszabályzót (EQ, equalizer).
8. Nyomjuk le a `Play` gombot - úgy fog tûnni, mintha az XMMS játszaná az MP3 állományt, de nem hallunk semmit. Ekkor a tartalmát állományba menti.
9. Mikor befejeztük a kitömörítést, ne felejtsük el visszaállítani az "Output Plugin" értékét az alapértelmezettre.

Írás a szabványos kimenetre az mpg123-mal:

1. Futtassuk le a `mpg123 -s audio01.mp3 > audio01.pcm` parancsot.

Az XMMS az állományokat WAV formátumban írja, miközben az mpg123 nyers PCM hangadatokat képez belõlük. A cdrecord használata során mind a két formátumból hozhatóak létre audio CD-k. A nyers PCM a burncd(8) programmal használható. Amikor WAV állományokkal dolgozunk, minden

egyes sáv elején egy apró kattanást hallhatunk: ez a WAV állomány fejléce lesz. A (audio/sox portból vagy csomagból telepíthetõ) SoX segédprogrammal a WAV formátumú állományok fejléce pillanatok alatt eltávolítható:

```
% sox -t wav -r 44100 -s -w -c 2 track.wav track.raw
```

A CD-írók FreeBSD alatti használatával kapcsolatban olvassuk el a Lézeres tárolóeszközök (CD-k) létrehozása és használatat.

7.4. Videók lejátszása

A videolejátszás egy nagyon friss és gyorsan fejlõdõ alkalmazási terület. Legyünk türelmesek, ez nem minden fog annyira könnyen menni, mint a hangok esetében.

A kezdéshez nem árt tudnunk, hogy a videokártyánk milyen gyártmányú és milyen chipet használ. Míg az Xorg és az XFree86™ számos különféle videokártyát ismer, csupán töredékükkel lehet jó lejátszási teljesítményt elõhozni. Az X11 futtatása közben az xdpyinfo(1) parancs kiadásával kérdezhetjük le az X szervertõl a kártyánk használatával elérhetõ kiterjesztéseket.

Érdemes a kezünk ügyében tartani egy rövidke MPEG formátumú állományt, amellyel majd ki tudjuk próbálni a különféle lejátszókat és azok beállításait. Mivel egyes DVD lejátszók alapértelmezés szerint a /dev/dvd helyen keresik a lejátszandó DVD eszközt, vagy egyszerûen csak így írták meg ezeket, mindenképpen hasznos lehet, ha szimbolikus linkeket hozunk létre a megfelelõ eszközökre:

```
# ln -sf /dev/acd0 /dev/dvd
# ln -sf /dev/acd0 /dev/rdvd
```

A devfs(5) mûködése miatt azonban ezek a kézzel létrehozott linkek az újraindítás után már nem maradnak meg. A szimbolikus linkeket a rendszer minden egyes indulásakor úgy tudjuk automatikusan létrehozni, hogyha az /etc/devfs.conf állományba felvesszük az alábbi sort:

```
link acd0 dvd
link acd0 rdvd
```

Emellett a DVD-k titkosításának feloldása, mely a DVD-meghajtók speciális funkcióit igényli, a DVD eszközökön írási jogot is igényel.

Az X11 osztott memóriát kezelõ felületének gyorsításához javasolt néhány sysctl(8) változó értékének megnövelése is:

```
kern.ipc.shmmax=67108864
kern.ipc.shmall=32768
```

7.4.1. A megjelenítõ képességeinek megállapítása

Több különbözõ úton lehet X11 alatt videókat nézni, de ennek tényleges módját igazából a rendelkezésre álló hardver határozza meg. Az itt leírt módszerek által kihozható minõség hardverenként eltérhet. Másodsorban a videók megjelenítése az X11-ben az utóbbi idõben igen nagy hangsúlyt kapott, ezért az Xorg és az XFree86™ minden egyes változatával jelentõsen javulhat a helyzet ezen a téren.

A videók megjelenítésére használt gyakori felületek:

1. X11: az X11 normális kimenete osztott memórián keresztül
2. XVideo: az X11 felületének kiterjesztése, ami tetszõleges X11 által kirajzolható objektum esetén támogat videót
3. SDL: a Simple Directmedia Layer
4. DGA: a Direct Graphics Access (közvetlen grafikus hozzáférés)
5. SVGAlib: alacsonyszintû konzolos grafikus réteg

XVideo

Az Xorg és az XFree86™ 4.X rendelkezik egy *XVideo* (avagy Xvideo, Xv, xv) elnevezésû kiterjesztéssel, amelyen keresztül egy speciális gyorsítás segítségével a kirajzolható objektumokban közvetlenül meg tudunk jeleníteni videókat. Ezzel a kiterjesztéssel még a gyengébb gépeken is nagyon jó minõségû lejátszást tudunk elérni.

A kiterjesztés mûködésérõl az `xvinfo` parancs kiadásával gyõzõdhetünk meg:

```
% xvinfo
```

Ha a parancs eredménye ehhez hasonló, akkor a kártyánk támogatja az XVideót:

```
X-Video Extension version 2.2
screen #0
  Adaptor #0: "Savage Streams Engine"
    number of ports: 1
    port base: 43
    operations supported: PutImage
    supported visuals:
      depth 16, visualID 0x22
      depth 16, visualID 0x23
    number of attributes: 5
      "XV_COLORKEY" (range 0 to 16777215)
              client settable attribute
              client gettable attribute (current value is 2110)
      "XV_BRIGHTNESS" (range -128 to 127)
              client settable attribute
              client gettable attribute (current value is 0)
      "XV_CONTRAST" (range 0 to 255)
```

```
            client settable attribute
            client gettable attribute (current value is 128)
    "XV_SATURATION" (range 0 to 255)
            client settable attribute
            client gettable attribute (current value is 128)
    "XV_HUE" (range -180 to 180)
            client settable attribute
            client gettable attribute (current value is 0)
  maximum XvImage size: 1024 x 1024
  Number of image formats: 7
    id: 0x32595559 (YUY2)
      guid: 59555932-0000-0010-8000-00aa00389b71
      bits per pixel: 16
      number of planes: 1
      type: YUV (packed)
    id: 0x32315659 (YV12)
      guid: 59563132-0000-0010-8000-00aa00389b71
      bits per pixel: 12
      number of planes: 3
      type: YUV (planar)
    id: 0x30323449 (I420)
      guid: 49343230-0000-0010-8000-00aa00389b71
      bits per pixel: 12
      number of planes: 3
      type: YUV (planar)
    id: 0x36315652 (RV16)
      guid: 52563135-0000-0000-0000-000000000000
      bits per pixel: 16
      number of planes: 1
      type: RGB (packed)
      depth: 0
      red, green, blue masks: 0x1f, 0x3e0, 0x7c00
    id: 0x35315652 (RV15)
      guid: 52563136-0000-0000-0000-000000000000
      bits per pixel: 16
      number of planes: 1
      type: RGB (packed)
      depth: 0
      red, green, blue masks: 0x1f, 0x7e0, 0xf800
    id: 0x31313259 (Y211)
      guid: 59323131-0000-0010-8000-00aa00389b71
      bits per pixel: 6
      number of planes: 3
      type: YUV (packed)
    id: 0x0
      guid: 00000000-0000-0000-0000-000000000000
      bits per pixel: 0
      number of planes: 0
      type: RGB (packed)
      depth: 1
```

```
        red, green, blue masks: 0x0, 0x0, 0x0
```

Az XVideo nem mindegyik implementációjában vannak jelen a felsorolt formátumok (YUV2, YUV12 stb.), ami viszont néhány lejátszó számára akadályokat jelenthet.

Amennyiben viszont ezt látjuk:

```
X-Video Extension version 2.2
screen #0
no adaptors present
```

Akkor a kártyánk nem rendelkezik XVideo támogatással.

Ha az XVideo nem támogatott a kártyánk számára, akkor az csupán csak annyit jelent, hogy a gépünknek nehéz dolga lesz a videók megjelenítéséhez szükséges számítási kapacitás kiszolgálásában. Azonban a videokártyánktól és processzorunktól függõen még így is kielégítõ eredményt tudunk elõcsalni. Ekkor viszont minden bizonnyal érdemes lesz átolvasnunk a Ajánlott olvasmányokban, miként tudjuk növelni a teljesítményét.

A Simple Directmedia Layer

A Simple Directmedia Layer, vagy SDL, eredetileg a Microsoft® Windows®, BeOS és UNIX® közti hordozhatóságot szándékozta megvalósítani, aminek segítségével a hangot és grafikát hatékonyan használni tudó alkalmazások hozhatóak létre. Az SDL által nyújtott réteg a hardver olyan alacsonyszintû absztrakcióját öleli fel, amely gyakran még az X11 felületénél is hatékonyabb.

Az SDL a devel/sdl12 helyen található.

Direct Graphics Access (Közvetlen grafikus hozzáférés)

A közvetlen grafikus hozzáférés az X11 egy olyan kiterjesztése, ami lehetõvé teszi a programok számára az X szerver megkerülését és így közvetlenül a videokártya memóriáját képesek elérni. Mivel a megosztás hatékony megvalósításához ez nagyban építkezik alacsonyszintû leképzési mûveletekre, ezért az ilyet használó programokat `root` felhasználóként kell futtatni.

A DGA kiterjesztés a dga(1) segítségével tesztelhetõ és mérhetõ. A `dga` parancs kiadása után minden billentyû lenyomására megváltoztatja a képernyõn látható színeket. A kilépéshez a `q` billentyût kell lenyomni.

7.4.2. A videókkal foglalkozó portok és csomagok

Ebben a szakaszban a FreeBSD Portgyûjteményébõl a videók lejátszására alkalmas programokat vesszük számba. A videolejátszás nagyon gyorsan fejlõdõ terület, ezért az itt említett különbözõ alkalmazások képességei az itt leírtaktól némileg eltérhetnek.

Elõször is fontos tisztában lennünk azzal, hogy számos FreeBSD-n futó videoalkalmazás eredetileg linuxos alkalmazásként indult, és közülük sokan még csak béta minõségûek. Íme a FreeBSD-n is megtalálható videocsomagokkal kapcsolatos néhány olyan gond, amivel esetleg összefuthatunk:

1. Az egyik alkalmazás nem képes visszajátszani olyan állományt, amit egy másik alkalmazás hozott létre.
2. Az alkalmazás nem képes visszajátszani a saját maga által készített állományokat.
3. Ugyanazon alkalmazás két különbözõ gépen, amikor mind a kettõn az adott konfigurációra fordítjuk le, ugyanazt az állományt másképpen játssza vissza.
4. Egy olyan látszólag egyértelmû szûrõ, mint például a kép átméretezése, a hibás átméretezõ rutin miatt nagyon csúnya eredményt produkál.
5. Az alkalmazás gyakran elszáll.
6. A porthoz nem találjuk a dokumentációt, egyedül csak az interneten vagy a port work könyvtárában van.

Sok alkalmazás a "linuxizmus" jeleit is hordozza, vagyis gondok adódhatnak abból, hogy a szerzõk az alkalmazások mûködtetéséhez a Linux rendszermag és a különféle terjesztésekben megtalálható módosított szabványos könyvtárak különlegességeit használják ki. Ezeket a portok karbantartói nem mindig észlelik és javítják ki, ami miatt az alábbiak bármikor bekövetkezhetnek:

1. A processzor jellemzõit a /proc/cpuinfo állományon keresztül állapítják meg.
2. A szálak helytelen használatuk miatt a program befejezõdésekor összeakadnak.
3. Az alkalmazással gyakran együtt használt egyéb alkalmazások még nem nincsenek benne a FreeBSD Portgyûjteményében.

Az ilyen alkalmazások fejlesztõi a hordozhatóság javításával és a problémák megoldásával kapcsolatban eddig mindig igyekeztek együttmûködni a portok karbantartóival.

MPlayer

Az MPlayer az utóbbi idõben felbukkant, gyorsan fejlõdõ videolejátszó. Fejlesztõinek célja a sebesség és rugalmasság a Linux, illetve más UNIX® rendszereken. A kezdeményezés abból fakadt, hogy a fejlesztés mögött álló csapat alapítójának elege lett az akkoriban elérhetõ lejátszók teljesítményébõl. Mondhatnánk, hogy ez a program feláldozta a grafikus felületet az áramvonalas kialakításért, azonban ha hozzászokunk a parancssori beállításokhoz és a billentyûkön keresztüli vezérléshez, remekül mûködik.

Az MPlayer lefordítása

Az MPlayer a multimedia/mplayer helyen található. A program a fordítási folyamat során elvégez számos hardverellenõrzést, aminek eredményeképpen az egyik rendszeren fordított program nem vihetõ a másikra. Ezért különösen fontos portból fordítani és nem pedig bináris csomagot használni. Mindezek mellett a Makefile állományban még számos, a `make` parancsnak a fordítás megkezdésekor átadható beállítást találhatunk:

```
# cd /usr/ports/multimedia/mplayer
# make
N - O - T - E

Take a careful look into the Makefile in order
```

```
to learn how to tune mplayer towards you personal preferences!
For example,
make WITH_GTK1
builds MPlayer with GTK1-GUI support.
If you want to use the GUI, you can either install
/usr/ports/multimedia/mplayer-skins
or download official skin collections from
http://www.mplayerhq.hu/homepage/dload.html
```

Az üzenet fordítása:

```
F - I - G - Y - E - L - E - M

Az mplayert személyes igényeinkhez úgy tudjuk igazítani, ha
figyelmesen átnézzük a Makefile állományt!  Például a WITH_GTK1
megadásával az MPlayer GTK1 alapú grafikus felülettel jön létre.
A grafikus felület használatához telepítenünk kell a
/usr/ports/multimedia/mplayer-skins portot is, vagy letölteni a
hivatalos skingyűjteményt a http://www.mplayerhq.hu/homepage/dload.html
oldalról.
```

A port alapbeállításai a legtöbb felhasználó számára megfelelõek, habár az Xvid kódek használatához meg kell adnunk a `WITH_XVID` beállítást. Rajta kívül még az alapértelmezett DVD eszközt is érdemes megadni a `WITH_DVD_DEVICE` beállítással, amelynek alapértéke a /dev/acd0.

A leírás elkészítésének idõpontjában az MPlayer portja létrehozza a HTML dokumentációt és a két végrehajtható állományt: az `mplayer` lejátszót és a videók újrakódolásáért felelõs `mencoder` segédprogramot.

Az MPlayer HTML dokumentációja nagyon közlékeny, és ha az olvasó nem találná valamelyik videohardver vagy felület leírását ebben a fejezetben, akkor ez a dokumentáció mindenképpen hasznos olvasnivalónak bizonyul. Ha a UNIX®-ok alatt elérhetõ videotámogatás leírását keressük, határozottan megéri idõt szánni az MPlayer dokumentációjának alapos végigolvasására.

Az MPlayer használata

Az MPlayer használatához a felhasználói könyvtárunkban rendelkeznünk kell egy .mplayer elnevezésû könyvtárral. Ezt a következõ paranccsal tudjuk létrehozni:

```
% cd /usr/ports/multimedia/mplayer
% make install-user
```

Az `mplayer` parancssori paraméterei a hozzá tartozó man oldalon találhatóak meg, valamint mindezek a HTML dokumentációban még részletesebben. Ebben a szakaszban csupán néhányukat mutatjuk be.

Egy állomány, mint például a tesztvideo.avi, a `-vo` beállításával játszható le a különbözõ felületeken:

```
% mplayer -vo xv tesztvideo.avi
```

```
% mplayer -vo sdl tesztvideo.avi
```

```
% mplayer -vo x11 tesztvideo.avi
```

```
# mplayer -vo dga tesztvideo.avi
```

```
# mplayer -vo 'sdl:dga' tesztvideo.avi
```

Érdemes az itt felsorolt konfigurációk mindegyikét kipróbálni, mivel az egymáshoz mért teljesítményük rengeteg tényezõn múlik, de közülük talán maga a hardver a legjelentõsebb.

A DVD-k lejátszásához cseréljük ki a tesztvideo.avi paramétert a `dvd://N -dvd-device ESZKÖZ` paraméterekkel, ahol az *N* a lejátszandó fejezet sorszáma, valamint az ESZKÖZ a DVD-hez tartozó eszközleíró. Például így tudjuk elkezdeni /dev/dvd eszközrõl a 3. fejezet lejátszását:

```
# mplayer -vo xv dvd://3 -dvd-device /dev/dvd
```

A port fordítása során a `WITH_DVD_DEVICE` paraméter segítségével megadható az alapértelmezett DVD eszköz, amely alapból a /dev/acd0. Errõl többet a port Makefile állományában találhatunk.

A leállításhoz, szüneteltetéshez, továbblépéshez és többi hasonló funkcióhoz tartozó billentyûket a `mplayer -h` parancs kimenetébõl vagy a man oldal elolvasásából deríthetjük ki.

A lejátszáshoz tartozó néhány viszonylag fontos beállítás: az `-fs -zoom` teljesképernyõs módra vált, valamint a `-framedrop` segít növeli a teljesítményt.

A lejátszáskor kiadandó parancs túlburjánzását el tudjuk kerülni, ha létrehozunk egy .mplayer/config állományt és itt állítjuk be a gyakori opciókat:

```
vo=xv
fs=yes
zoom=yes
```

Végezetül megemlítjük, hogy az `mplayer` segítségével a DVD-n található fejezeteket ki tudjuk menteni .vob állományokba. A DVD második fejezetének kimentéséhez gépeljük be ezt:

```
# mplayer -dumpstream -dumpfile out.vob dvd://2 -dvd-device /dev/dvd
```

A parancs eredményeképpen keletkezõ out.vob állomány formátuma MPEG lesz, amit a fejezetben bemutatott további csomagokkal tudunk feldolgozni.

mencoder

A `mencoder` használatának megkezdése elõtt javasolt alaposan beleásnunk magunkat a HTML dokumentációba és megismerkednünk az alapvetõ beállításaival. Van külön man oldala is, azonban a HTML leírás nélkül önmagában ez nem túl sokat ér. Megszámlálhatatlan úton és módon növelhetõ benne a minõség, csökkenthetõ a kódolási arány, változtatható a formátum, és ezen apró finomságok felelõsek a jó vagy éppen a rossz teljesítményért. A témába néhány példa bemutatásával igyekszünk beavatni az olvasót. Elõször vegyünk egy egyszerû másolást:

```
% mencoder bemenő.avi -oac copy -ovc copy -o eredmény.avi
```

A parancssori paraméterek helytelen kombinációja olyan állományokat eredményezhet, amelyeket még maga az `mplayer` sem képes lejátszani. Ezért ha csak le akarunk szedni egy állományt, akkor maradjunk meg az `mplayer -dumpfile` opciójánál.

A bemenõ.avi állományt MPEG4 video- és MPEG3 hangtömörítéssel (amihez kell majd a audio/lame) így tudjuk lekódolni:

```
% mencoder bemenõ.avi -oac mp3lame -lameopts br=192 \
     -ovc lavc -lavcopts vcodec=mpeg4:vhq -o eredmény.avi
```

Ezzel az `mplayer` és `xine` programok számára is egyaránt lejátszható állomány jön létre.

A DVD fejezeteit úgy tudjuk közvetlenül kódolni, ha a parancssorban kicseréljük a bemenõ.avi állományt az `dvd://1 -dvd-device /dev/dvd` beállításra, illetve ha a programot `root` felhasználóként futtatjuk. De mivel elsõre általában ritkán vagyunk elégedettek a kódolással, érdemes elõször inkább lementeni az egész fejezetet egy állományba, majd azon dolgozni.

A xine videolejátszó

A xine egy széles hatókörû projekt, amelynek nem csak az a célja, hogy egy "mindenes" videolejátszó alkalmazást fejlesszenek, hanem az is, hogy újrahasznosítható függvénykönyvtárakat és egy moduláris felépítésû programot hozzanak létre, amely kiegészítûkkel bûvíthetû. A multimedia/xine helyen portként, valamint csomagként is elérhetõ.

A xine itt-ott még valamelyest durva, de mindenképpen egy dicséretes kezdeményezés. A xine a gyakorlatban erõs processzort és mellé gyors videokártyát kíván, vagy pedig az XVideo kiterjesztés támogatását. A grafikus felhasználói felülete ugyan használható, de még kicsit esetlen.

Az írás pillanatában a xine mellé még nem kapunk olyan modult, amivel le tudnánk játszani a CSS kódolású DVD-ket. Léteznek azonban olyan külsõs modulok, amelyekkel meg lehet valósítani ezt a feladatot, azonban a FreeBSD Portgyûjteményében ezeket még nem találhatjuk meg.

A xine az MPlayerhez képes többet tesz a felhasználóért, azonban ezzel egyidõben el is veszi tõle a finomhangolás lehetõségét. A xine legjobban az XVideót ismerõ felületeken teljesít.

A xine alapértelmezés szerint grafikus felülettel indul, ahol a menük segítségével tudunk megnyitni egy adott állományt:

```
% xine
```

Vagy a grafikus felület használata nélkül kiadhatjuk közvetlenül is az állomány lejátszását:

```
% xine -g -p kedvencmozim.avi
```

A transcode

A transcode nem egy újabb lejátszó, hanem a video- és audio állományok újratömörítésére használható programok gyûjteménye. A transcode segítségével a szabványos be- és kimeneten keresztül parancssoros programokkal képesek vagyunk videoállományokat összefûzni, megjavítani.

A multimedia/transcode port fordítása során temérdek beállítást adhatunk meg, amelyek közül az alábbi parancsban foglaljuk össze az általunk javasolandókat:

```
# make WITH_OPTIMIZED_CFLAGS=yes WITH_LIBA52=yes WITH_LAME=yes WITH_OGG=yes \
WITH_MJPEG=yes -DWITH_XVID=yes
```

Ezek a beállítások a legtöbb felhasználó számára elegendõek.

A `transcode` képességeinek illusztrálásához lássunk egy példát, amiben megmutatjuk, hogyan kell egy DivX állományt PAL szabványú MPEG-1 formátumú (PAL VCD) állománnyá alakítani:

```
% transcode -i bemenõ.avi -V --export_prof vcd-pal -o output_vcd
% mplex -f 1 -o eredmény_vcd.mpg eredmény_vcd.m1v eredmény_vcd.mpa
```

Az eredményül keletkezõ eredmény_vcd.mpg MPEG állomány akár már játszható is MPlayerrel. Ha az állományt kiírjuk egy írható CD-re, akkor ezzel video CD-t is létre tudunk hozni, amihez viszont szükségünk van mind a multimedia/vcdimager és sysutils/cdrdao programokra.

A `transcode` parancsnak van saját man oldala, azonban ehelyett a transcode wikiben érdemes inkább további információkat és példákat keresni.

7.4.3. Ajánlott olvasmányok

A FreeBSD-hez tartozó videoszoftverek nagyon gyorsan fejlõdnek. Könnyen elképzelhetõ, hogy az imént tárgyalt problémák legtöbbje a közeljövõben hamarosan megoldódik. Addig viszont bárkinek, aki a legtöbbet szeretné kihozni a FreeBSD audio- és video lehetõségeibõl, rengeteg leírás és dokumentáció elolvasása alapján kell összecsiszolnia a különbözõ beállításokat, és csak néhány alkalmazás mellett érdemes kitartania. Ebben a szakaszban igyekszünk segíteni az olvasónak megtalálni az ilyen jellegû információkat.

Az MPlayer dokumentációja szakmai szempontból igen közlékeny. Ezt mindenkinek érdemes elolvasnia, aki a késõbbiekben magasabb szakmai szinten akar foglalkozni a UNIX®-os videózással. Az MPlayer levelezési listája viszont alig tolerálja a dokumentációt rendesen el nem olvasó emberek kérdéseit, ezért minden egyes hiba bejelentése elõtt lehetõleg rendesen nézzük át a dokumentáció odavágó részeit.

A xine HOGYAN egyik külön fejezetében az összes lejátszó esetén érvényesíthetõ teljesítménynövelési módszereket mutat be.

Végül íme néhány ígéretes alkalmazás, amelyeket érdemes kipróbálnunk:

- Avifile, ami egyben a multimedia/avifile port
- Ogle, ami a multimedia/ogle port
- Xtheater
- multimedia/dvdauthor, egy nyílt forráskódú DVD-tartalom szerkesztõ

7.5. TV kártyák beállítása

7.5.1. Bevezetés

A TV kártyák segítségével kábeles vagy antennás televízióadásokat tudunk nézni a számítógépünkön. A legtöbbjük RCA vagy S-video bemenettel rendelkezik, valamint néhányukon még FM rádiókészülék is megtalálható.

A FreeBSD a bktr(4) meghajtón keresztül a Brooktree Bt848/849/878/879, illetve a Conexant CN-878/Fusion 878a típusú, PCI-os videorögzítõ chipeket ismeri. Ügyelnünk kell arra, hogy a kártyánkon levõ vevõkészülék is használható legyen, amit pedig a bktr(4) man oldalán megtalálható támogatott eszközök listájából ellenõrizhetünk.

7.5.2. A meghajtó beállítása

A kártyánk használatához be kell töltenünk a bktr(4) meghajtót, ami csupán annyiból áll, hogy a /boot/loader.conf állományhoz hozzáadunk egy ilyen sort:

```
bktr_load="YES"
```

Másik lehetõségünk, ha a TV kártya támogatását statikusan beleépítjük a rendszermagba. Ha ezt a megoldást választjuk, a következõ sorokat kell elhelyeznünk a rendszermag beállításait tartalmazó állományba:

```
device   bktr
device  iicbus
device  iicbb
device  smbus
```

A fentebb látható egyéb eszközök megadása azért szükséges, mert a kártya részegységei egy I2C

buszon csatlakoznak egymáshoz. Miután beillesztettük a szükséges változtatásokat, fordítsuk le és telepítsük az új rendszermagot.

A támogatás hozzáadása után újra kell indítanunk a számítógépünket. A rendszerindítási folyamat során meg kell jelennie a TV kártyánknak is, valahogy így:

```
bktr0: <BrookTree 848A> mem 0xd7000000-0xd7000fff irq 10 at device 10.0 on pci0
iicbb0: <I2C bit-banging driver> on bti2c0
iicbus0: <Philips I2C bus> on iicbb0 master-only
iicbus1: <Philips I2C bus> on iicbb0 master-only
smbus0: <System Management Bus> on bti2c0
bktr0: Pinnacle/Miro TV, Philips SECAM tuner.
```

Természetesen a fenti üzenetek az aktuális hardvereszközünknek megfelelõen némileg eltérhetnek. Ellenõrizzük, hogy a vevõkészüléket helyesen ismerte-e fel a rendszer. Ha nem sikerült volna, akkor a sysctl(8) és a rendszermag beállításai segítségével még mindig van lehetõségünk állítani rajta. Például, ha egy Philips SECAM vevõkészüléket akarunk beállítani, akkor a rendszermag beállításaihoz még hozzá kell adni a következõ sort:

```
options OVERRIDE_TUNER=6
```

vagy erre közvetlenül használhatjuk a sysctl(8) programot is:

```
# sysctl hw.bt848.tuner=6
```

A bktr(4) man oldalán és a /usr/src/sys/conf/NOTES állományban megtalálhatjuk a többi beállítás részletes leírását is.

7.5.3. Hasznos alkalmazások

A TV kártyánk tényleges használatához azonban még a következõ alkalmazások valamelyikét is telepítenünk kell:

- A multimedia/fxtv használatával ablakban tévézhetünk, valamint lehetõségünk van kép/audio/video kimentésére is.
- A multimedia/xawtv az fxtv-hez hasonló lehetõségekkel bíró tévénézõ alkalmazás.
- A misc/alevt dekódolja és megjeleníti a mûsorhoz kapcsolódó Videotex/Teletext üzeneteket.
- Az audio/xmradio segítségével az egyes TV kártyákon megtalálható FM rádiókészülékeket tudjuk használatba venni.
- Az audio/wmtune a rádióvevõkhöz használható hasznos grafikus alkalmazás.

Ebben a témában a FreeBSD Portgyûjteményében további érdekes alkalmazások találhatóak még.

7.5.4. Hibakeresés

Ha bármilyen gond adódna a TV kártyánkkal kapcsolatosan, akkor elõször mindenképpen érdemes megnézni, hogy a rajta levõ videorögzítõ chipet és vevõkészüléket a bktr(4) meghajtó ténylegesen ismeri-e, illetve hogy jól állítottuk-e be. A TV kártyákra irányuló különféle egyéb kérdések és segítség tekintetében érdemes lehet még levelet küldeni a FreeBSD multimedia levelezési lista címére is.

7.6. Lapolvasók

7.6.1. Bevezetés

A FreeBSD lapolvasókhoz a SANE (Scanner Access Now Easy) elnevezésû API (alkalmazásfejlesztõi felület) segítségével képes hozzáférni, amelyet a Portgyûjteményben találhatunk meg. A lapolvasást végzõ hardvereszközök használatához a FreeBSD a SANE mellett még néhány eszközmeghajtóra is támaszkodik.

A FreeBSD egyaránt ismeri az SCSI és USB csatlakoztatású lapolvasókat is. Még mielõtt nekikezdenénk a lapolvasó beállításához, bizonyosodjuk meg róla, hogy a SANE támogatja. A SANE által ismert eszközök felsorolásában ellenõrizhetjük a lapolvasónk támogatottságának állapotát. A FreeBSD 8.*X* elõtti kiadásaiban ezenkívül még a uscanner(4) man oldalon is láthatjuk az ismert USB-s lapolvasók listáját.

7.6.2. A rendszermag beállítása

A korábbiak értelmében tehát mind a SCSI, mind pedig a USB felületen csatlakozó eszközök támogatottak. A lapolvasónknak megfelelõen eltérõ eszközmeghajtók szükségesek.

Beállítás USB felületen

A GENERIC rendszermag alapértelmezés szerint tartalmazza az USB-s lapolvasók használatához szükséges eszközmeghajtókat. Ha valamiért azonban mégis saját rendszermagot akarunk használni, akkor ne felejtsük el ellenõrizni, hogy a rendszermag beállításai között megtalálhatóak a következõ sorok:

```
device usb
device uhci
device ohci
device ehci
```

A FreeBSD 8.*X* elõtti kiadásaiban még a következõ sorra is szükségünk lesz:

```
device uscanner
```

A FreeBSD ezen változataiban a uscanner(4) eszközmeghajtón keresztül tudjuk használni az USB csatolóval rendelkezõ lapolvasókat. A FreeBSD 8.0 változatától kezdõdõen pedig ehhez a libusb(3) függvénykönyvtár nyújt közvetlen támogatást.

A megfelelõen elõkészített rendszermag elindítása után csatlakoztassuk az USB-s lapolvasónkat. Ez a sor fog megjelenni a rendszer üzenetpufferében (dmesg(8)):

```
ugen0.2: <EPSON> at usbus0
```

Vagy FreeBSD 7.*X* rendszerek esetében:

```
uscanner0: EPSON EPSON Scanner, rev 1.10/3.02, addr 2
```

Ezek az üzenetek elárulják nekünk, hogy a lapolvasóhoz mostantól a használt FreeBSD verziótól függõen a /dev/ugen0.2 vagy a /dev/uscanner0 eszközleíró tartozik. A fenti példában egy EPSON Perfection® 1650 típusú USB lapolvasót láthatunk.

Beállítás SCSI felületen

Ha a lapolvasónk SCSI felületen csatlakozik, fontos tisztában lennünk azzal, hogy pontosan milyen SCSI-vezérlõn keresztül is érhetjük el, ugyanis a rajta található SCSI chipkészletnek megfelelõen kell majd hangolnunk a rendszermag beállításait. A GENERIC rendszermag alapból ismeri a leggyakrabban elõforduló SCSI-vezérlõket. Mindenképpen olvassuk át a NOTES nevû állományt és adjuk hozzá a rendszermag beállításaihoz a megfelelõ sort. A SCSI-kártya meghajtóján kívül még az alábbi beállításokat is meg kell adnunk a rendszermagunk számára:

```
device scbus
device pass
```

Ahogy sikerült a rendszermagot sikeresen lefordítani és telepíteni, a rendszer indulása során az üzenetpufferben már láthatjuk is a felismert eszközt:

```
pass2 at aic0 bus 0 target 2 lun 0
pass2: <AGFA SNAPSCAN 600 1.10> Fixed Scanner SCSI-2 device
pass2: 3.300MB/s transfers
```

Ha a rendszer indulásakor még nem kapcsoltuk volna be a lapolvasónkat, a camcontrol(8) parancs segítségével késõbb külön kérhetjük a SCSI buszon található eszközök újbóli felderítését:

```
# camcontrol rescan all
Re-scan of bus 0 was successful
Re-scan of bus 1 was successful
Re-scan of bus 2 was successful
Re-scan of bus 3 was successful
```

Ekkor a lapolvasó megjelenik a SCSI eszközök felsorolásában:

```
# camcontrol devlist
```

```
<IBM DDRS-34560 S97B>              at scbus0 target 5 lun 0 (pass0,da0)
<IBM DDRS-34560 S97B>              at scbus0 target 6 lun 0 (pass1,da1)
<AGFA SNAPSCAN 600 1.10>           at scbus1 target 2 lun 0 (pass3)
<PHILIPS CDD3610 CD-R/RW 1.00>     at scbus2 target 0 lun 0 (pass2,cd0)
```

A SCSI eszközökrõl további leírásokat a scsi(4) és camcontrol(8) man oldalakon találhatunk.

7.6.3. A SANE beállítása

A SANE rendszere két részre oszlik: a backendekre (graphics/sane-backends) és a frontendekre (graphics/sane-frontends). Ezek közül maguk a backendek szolgáltatják a lapolvasó hozzáférhetõségét. A SANE által ismert eszközeinek listájából kifürkészhetjük, hogy lapolvasónkat melyik backenden keresztül érhetjük el. Az eszköz megfelelõ használatához döntõ fontosságú megállapítani a hozzá tartozó backendet. A frontendek között találjuk meg a lapolvasást felügyelõ grafikus felületeket (mint például az xscanimage).

Elsõként telepítsük a graphics/sane-backends portot vagy csomagot. Ezután ellenõrizzük, hogy a SANE felismeri a lapolvasót, és ehhez adjuk ki a `sane-find-scanner` parancsot:

```
# sane-find-scanner -q
found SCSI scanner "AGFA SNAPSCAN 600 1.10" at /dev/pass3
```

A kimenetében jelzi a felületet, amin a lapolvasó csatlakozik, valamint a hozzá tartozó eszközleírót. A gyártó neve és a termék típusa nem minden esetben jelenik meg, de ez nem is annyira fontos.

Némely USB-s lapolvasók esetén még egy firmware-t is be kell töltenünk, amirõl bõvebben a backendhez tartozó man oldalokon olvashatunk. Ajánlott még elolvasni a sane-find-scanner(1) és linprocfs(7) man oldalakat is.

Most pedig nézzük meg, hogy vajon a frontend is be tudja-e azonosítani a lapolvasónkat. Alapértelmezés szerint a SANE backendjéhez tartozik még egy sane(1) nevû segédprogram is, aminek segítségével listázni tudjuk a használható eszközöket és képeket tudunk beolvasni parancssorból. Közülük a `-L` kapcsoló listáz:

```
# scanimage -L
device `snapscan:/dev/pass3' is a AGFA SNAPSCAN 600 flatbed scanner
```

Vagy ha a Beállítás USB felületenban szereplõ USB lapolvasóval nézzük:

```
# scanimage -L
device 'epson2:libusb:/dev/usb:/dev/ugen0.2' is a Epson GT-8200 flatbed scanner
```

Ezt a kimenetet egy FreeBSD 8.*X* rendszeren kaptuk, ahol a `'epson2:libusb:/dev/usb:/dev/ugen0.2'` az eszközhöz tartozó backendet (`epson2`) és eszközleírót (`/dev/ugen0.2`) adja meg.

Ha ennek eredményeképpen semmi sem jelenik meg, vagy a sane(1) látszólag nem talált semmilyen eszközt, akkor a lapolvasó azonosítása nem sikerült. Ilyen esetekben valószínûleg módosítanunk kell a backend beállításait tartalmazó állományt a használni kívánt lapolvasó eszköz szerint. A backendek beállításait a /usr/local/etc/sane.d/ könyvtárban találjuk. Ez a probléma bizonyos USB-s lapolvasók esetében jelentkezik.

Például, ha Beállítás USB felületenban használt USB-s lapolvasónkat FreeBSD 8.*X* alatt tökéletesen felismeri a rendszer, de a FreeBSD korábbi változatai esetén (ahol a uscanner(4) eszközmeghajtót használják) a `sane-find-scanner` parancs a következõket adja vissza:

```
# sane-find-scanner -q
found USB scanner (UNKNOWN vendor and product) at device /dev/uscanner0
```

Akkor a lapolvasót sikerült megtalálni, és láthatjuk, hogy USB-n keresztül csatlakozik és a /dev/uscanner0 eszközleíró tartozik hozzá. Most már ellenõrizhetjük a lapolvasó helyes beazonosítását is:

```
# scanimage -L

No scanners were identified. If you were expecting something different,
check that the scanner is plugged in, turned on and detected by the
sane-find-scanner tool (if appropriate). Please read the documentation
which came with this software (README, FAQ, manpages).
```

Az üzenet fordítása:

```
Nincs azonosítható lapolvasó.  Ha nem erre számítottunk, akkor
ellenőrizzük,
hogy az eszközt tényleg bekapcsoltuk, csatlakoztattuk és észlelte a
sane-find-scanner segédprogram (amennyiben szükséges).  Kérjük, olvassa
el a
szoftverhez tartozó dokumentációt (README, FAQ, man oldalak)!
```

Mivel a lapolvasót nem sikerült azonosítani, át kell írnunk a /usr/local/etc/sane.d/epson2.conf állományt. A használt lapolvasó típusa EPSON Perfection® 1650, ezért hozzá az `epson2` backendet fogjuk használni. Ehhez feltétlenül olvassuk el a konfigurációs állományban található megjegyzéseket is. A sorokat igen könnyû átírni: tegyük megjegyzésbe az összes olyat, ahol a lapolvasónk számára nem megfelelõ felületek találhatóak (a mi esetünkben tehát megjegyzésbe fogjuk tenni az összes `scsi` szóval kezdõdõ sort, hiszen nekünk USB-s eszközünk van), majd az állomány végére írjuk be a használni kívánt felületet és eszközleírót. Ez ebben a konkrét esetben ennyi lenne:

```
usb /dev/uscanner0
```

A megfelelõ formátum és a további részletek leírásához ne felejtsük el azonban elolvasni a backend konfigurációs állományában felbukkanó megjegyzéseket és az ide tartozó man oldalt sem. Most már megpróbálkozhatunk újra a lapolvasó azonosításával:

```
# scanimage -L
device `epson:/dev/uscanner0' is a Epson GT-8200 flatbed scanner
```

Láthatjuk, hogy az USB-s lapolvasónkat sikerült azonosítani. Nem számít, ha esetleg nem egyezne a valósággal a gyártó vagy a típus megjelölése. Itt a valóban lényeges elem az `epson:/dev/uscanner0` mezõ lesz, melynek a backend és az eszközleíró nevét kell helyesen tartalmaznia.

A beállítást akkor zárhatjuk le, miután a `scanimage -L` parancs képes észlelni a lapolvasót. A eszköz ekkor már készen áll a beolvasásra.

Míg a sane(1) parancssorból teszi lehetõvé számunkra a lapolvasást, addig érdemesebb a képek olvasását egy grafikus felületen keresztül végeznünk. A SANE egy egyszerû, ám hatékony grafikus felületet ajánl fel ehhez, ez az xscanimage (graphics/sane-frontends).

Az Xsane (graphics/xsane) egy másik népszerû grafikus frontend. Segítségével speciális lehetõségeket is kihasználhatunk, mint például többféle képolvasási mód (fénymásoló, fax stb.), színkorrekció, kötegelt beolvasás, stb. Mind a két említett alkalmazás elérhetõ a The GIMP bõvítményeként is.

7.6.4. A lapolvasó használatának engedélyezése más felhasználók számára

A korábban tárgyalt mûveletek mindegyikét `root` felhasználóként tudjuk csak végrehajtani. Azonban elõfordulhat, hogy más felhasználók számára is szeretnénk hozzáférést biztosítani a lapolvasóhoz. Ehhez az érintett felhasználóknak a lapolvasóhoz tartozó eszközleíróhoz olvasási és írás joggal kell rendelkezniük. Például az USB-s lapolvasónk a /dev/ugen0.2 eszközleírót használja, amely valójában csak a /dev/usb/0.2.0 eszközleíróra mutató szimbolikus link (ezt gyorsan le tudjuk ellenõrizni, ha megnézzük a /dev könyvtár tartalmát). Az eszközleíró és a rá mutató szimbolikus link rendre a `wheel` és `operator` csoportok birtokában van. Ha a `pgj` nevû felhasználót felvesszük ezekbe a csoportokba, akkor ezáltal hozzá tud majd férni a lapolvasóhoz. Nyilvánvaló biztonsági megfontolásokból azonban kétszer is javasolt meggondolni, mely felhasználókat mely csoportokba vesszük fel, különösen, ha `wheel` csoportról van szó. Ennél valamivel jobb megoldást kínál, ha létrehozunk külön az USB eszközök használatára vonatkozó csoportot és a lapolvasót ezen csoport tagjainak számára elérhetõvé tesszük.

Tehát erre a célra például megalkotjuk a `usb` csoportot. Ehhez elsõ lépésként a pw(8) parancs segítségével hozzuk létre magát a csoportot:

```
# pw groupadd usb
```

Ezután a /dev/usb/0.2.0 eszközleírót és a rá mutató /dev/ugen0.2 szimbolikus linket kell az `usb` csoport részére elérhetõvé tennünk, a megfelelõ írási engedélyekkel (`0660` vagy `0664`) együtt, mivel alapértelmezetten csak a tulajdonosuk (`root`) tudja írni ezeket. Mindezt úgy tudjuk megtenni, ha az /etc/devfs.rules állományhoz hozzáadjuk a megfelelõ sorokat:

```
[system=5]
add path ugen0.2 mode 0660 group usb
add path usb/0.2.0 mode 0660 group usb
```

A FreeBSD 7.*X* változatok esetén valószínûleg a következõ sorokra lesz szükségünk a /dev/uscanner0 eszközleíróhoz:

```
[system=5]
add path uscanner0 mode 0660 group usb
```

Ezt követõen az /etc/rc.conf állományba írjuk be az alábbi sort és utána indítsuk újra a számítógépet:

```
devfs_system_ruleset="system"
```

Az itt szereplõ sorok pontos jelentésérõl a devfs(8) man oldaláról tájékozódhatunk.

Ezután már csak fel kell vennünk azokat a felhasználókat a `usb` csoportba, amelyeknek engedélyezzük a lapolvasó használatát:

```
# pw groupmod usb -m pgj
```

A további részletekrõl a pw(8) man oldalon olvashatunk.

Chapter 8. A FreeBSD rendszermag testreszabása

8.1. Áttekintés

A rendszermag a FreeBSD operációs rendszer lelke. Felelõs a memória kezelésért, a biztonsági szabályozások betartatásáért, a hálózat mûködtetéséért, a lemezhozzáférésért és sok minden másért is. Miközben maga a FreeBSD egyre jobban konfigurálható dinamikusan, addig alkalmanként elegedhetetlen, hogy újrakonfiguráljuk és újrafordítsuk a rendszermagot.

A fejezet elolvasása során megismerjük:

- miért lehet szükségünk egy saját rendszermagra;
- hogyan készítsünk konfigurációs állományt a rendszermaghoz, vagy hogyan módosítsunk egy már létezõt;
- hogyan használjuk a rendszermag konfigurációs állományát egy új rendszermag lefordítására és létrehozására;
- hogyan telepítsük az új rendszermagot;
- hogyan orvosoljuk a felmerülõ problémákat.

A fejezetben az összes példaként bemutatásra kerülõ parancsot `root` felhasználóként kell kiadni a sikeres végrehajtásukhoz.

8.2. Miért készítsünk saját rendszermagot?

A FreeBSD eredetileg ún. "monolitikus" rendszermaggal rendelkezett. Ez azt jelenti, hogy a rendszermag egyetlen nagy program volt, ami elõre rögzített eszközöket ismert, és ha meg akartuk változtatni a rendszermag mûködését, akkor új rendszermagot kellett fordítanunk, majd újra kellett indítanunk vele a számítógépet.

Manapság azonban a FreeBSD már inkább afelé a megközelítés felé halad, ahol a rendszermag funkcionalitásának nagy részét mûködés közben az igények szerint betölthetõ és eltávolítható modulok adják. Ezzel lehetõvé válik, hogy a rendszermag gyorsan illeszkedjen az újonnan megjelenõ hardvereszközökhöz (mint például a laptopok PCMCIA-kártyáihoz), vagy olyan új funkciókat tegyünk a rendszermaghoz, amelyek a fordításánál nem voltak feltétlenül szükségesek. Ezt a modellt nevezik moduláris rendszermagnak.

Ennek ellenére még mindig elkerülhetetlen, hogy esetenként ne legyen szükség a rendszermag statikus testreszabására. Ez a legtöbb esetben azzal magyarázható, hogy vannak olyan funkciók, amelyek túlságosan is mélyen helyezkednek el a rendszermagban, ezáltal nem tölthetõek be dinamikusan. Máskor viszont egyszerûen azért nem lehetséges, mert még senki sem szánt idõt az adott funkcióhoz tartozó, dinamikusan betölthetõ modul elkészítésére.

Egy saját rendszermag készítése azon legfontosabb próbatételek egyike, melyet egy haladó BSD felhasználónak ki kell állnia. Ez a folyamat, habár némileg idõigényes, számos elõnyt tartogat

FreeBSD rendszerünk számára. Eltérõen egy GENERIC (általános) rendszermagtól, amely rengeteg hardvert támogat, egy saját rendszermag csak a *saját* PC-nk hardverét ismeri. Ennek több elõnye is van, például:

- A rendszerünk gyorsabban indul. Mivel a rendszermag csak azokat a hardvereket fogja keresni, melyek a rendszerünkben megtalálhatóak, jelentõs mértékben le tud csökkeni az induláshoz szükséges idõ.
- Kisebb memóriahasználat. Egy saját rendszermag a szükségtelen részek és eszközmeghajtók elhagyása miatt gyakran kevesebb memóriát emészt fel, mint a GENERIC rendszermag. Ez azért is fontos, mert a rendszermag mindig benn van a fizikai memóriában, és ezzel az alkalmazások elõl veszi el a helyet. Emiatt egy saját rendszermag elkészítése különösen hasznos lehet egy kevés fizikai memóriával rendelkezõ rendszeren.
- További hardverek támogatása. A saját rendszermagunkba olyan eszközök támogatását is beletehetjük, amelyek nem szerepelnek a GENERIC rendszermagban, mint például a hangkártyákét.

8.3. A rendszerünkben levõ hardverek összeszedése

Mielõtt belevetnénk magunkat a rendszermag beállításába, érdemes egy leltárt készíteni a gépünkben található különbözõ eszközökrõl. Ahol a FreeBSD nem elsõdlegesen használt operációs rendszer, ott ehhez elegendõ megnézni a jelenlegi rendszerben található elemeket. Például a Microsoft® rendszerek Eszközkezelõjében (Device Manager) általában az összes eszköz fontosabb adatait megtaláljuk. Magát az Eszközkezelõt pedig a Vezérlõpultból (Control Panel) érhetjük el.

A Microsoft® Windows® egyes verzióiban a Rendszer (System) ikonjára kattintva megkapjuk azt a képernyõt, ahonnan közvetlenül el tudjuk érni az Eszközkezelõt.

Ha viszont nincs másik operációs rendszer a gépünkön, akkor magunknak kell mindezeknek utánanéznünk. Erre az egyik alkalmas módszer a dmesg(8) és a man(1) parancsok használata. A FreeBSD-ben található legtöbb meghajtónak van saját man oldala, ami tartalmazza az általuk kezelt eszközök listáját, illetve így a rendszerindítás során észlelt hardvereket nézhetjük vissza. Például az alábbi sorok arra utalnak, hogy a psm meghajtó megtalálta a gépünkhöz tartozó egeret:

```
psm0: <PS/2 Mouse> irq 12 on atkdbc0
psm0: [GIANT-LOCKED]
psm0: [ITHREAD]
psm0: model Generic PS/2 mouse, device ID 0
```

Ezután ezt a meghajtót vagy a rendszermagba kell beépítenünk, vagy pedig a loader.conf(5) állományon keresztül betöltenünk.

Bizonyos esetekben a `dmesg` az eszközök felkutatásának eredményei helyett csak a rendszer üzeneteit mutatja. Ilyen helyzetekben a teljes kimenet a /var/run/dmesg.boot állományban tekinthetõ meg.

A hardverek manuális felderítésének módja a pciconf(8) segédprogram kimenetének böngészése, ami valamivel részletesebb eredményt ad. Mint például:

```
ath0@pci0:3:0:0:        class=0x020000 card=0x058a1014 chip=0x1014168c rev=0x01
hdr=0x00
         vendor     = 'Atheros Communications Inc.'
         device     = 'AR5212 Atheros AR5212 802.11abg wireless'
         class      = network
         subclass   = ethernet
```

A `pciconf -lv` paranccsal kapott kimenet ezen része azt mutatja, hogy az ath meghajtó talált egy vezeték nélküli Ethernet eszközt. Innen a `man ath` paranccsal érhetjük el a ath(4) man oldalát.

A man(1) a `-k` paraméter megadásával további hasznos információkkal is tud szolgálni. A fentiekbõl kiindulva például a következõ paranccsal:

```
# man -k Atheros
```

le tudjuk kérdezni azokat a man oldalakat, amelyek tartalmazzák az adott szót:

```
ath(4)                  - Atheros IEEE 802.11 wireless network driver
ath_hal(4)              - Atheros Hardware Access Layer (HAL)
```

A hardvereszközeink listájával felvértezve most már egy saját rendszermag létrehozása sem lesz annyira ijesztõ.

8.4. Meghajtók, alrendszerek és modulok

Mielõtt új rendszermagot készítenénk, érdemes megfontolnunk, hogy egyáltalán szükségünk lesz-e rá. Ha például valamilyen eszköz támogatásához kell, akkor könnyen elõfordulhat, hogy azt modulként is be tudjuk tölteni.

A rendszermaghoz tartozó modulok a /boot/kernel könyvtárban találhatóak, és a kldload(8) segítségével a rendszer mûködése közben dinamikusan betölthetõek. Ha nem is az összes, de a legtöbb meghajtóhoz tartozik egy modul és egy man oldal. Például az elõzõ szakaszban az ath vezeték nélküli Ethernet meghajtóval foglalkoztunk. A következõ leírást találjuk a hozzá tartozó man oldalon:

```
Vagy ha modulként akarjuk betölteni ezt a meghajtót a rendszer indítása
során, akkor a man:loader.conf[5] állományba vegyük fel a következõ
sort:

    if_ath_load="YES"
```

A fentebb leírtak szerint tehát, ha az `if_ath_load="YES"` sort hozzáadjuk a /boot/loader.conf állományhoz, akkor a rendszer indulásakor ez a modul mindig dinamikusan betöltõdik.

Némely esetben azonban nem áll rendelkezésünkre ilyen modul. Ez különösen igaz bizonyos

alrendszerekre és a fontosabb meghajtókra, például az FFS állományrendszerre vonatkozóan, mivel ezeknek kötelezõen a rendszermagban kell lenniük. Ugyanez elmondható a hálózati támogatásra is (INET). Csak úgy tudjuk megmondani, hogy valamelyik meghajtóra szükség van a rendszermagban, ha elõször megpróbáljuk megkeresni hozzá a megfelelõ modult.

A beépített meghajtók figyelmetlen eltávolításával könnyen lefordíthatatlan állapotba kerülhet a rendszermag. Például, ha az ata(4) meghajtót kivesszük a rendszermag konfigurációs állományából, az ATA alrendszert használó meghajtók csak abban az esetben fognak biztosan mûködni, ha egyúttal felvesszük a loader.conf állományba. Ha nem vagyunk benne biztosak, akkor elõször próbáljuk meg használni a modult, és csak utána hagyjuk el a rendszermagba épített változatát.

8.5. Saját rendszermag készítése és telepítése

Elõször is tegyünk egy rövidke sétát a rendszermag könyvtárában. A továbbiakban említendõ összes könyvtár a /usr/src/sys könyvtáron belül található, amely /sys néven is elérhetõ. Itt rengeteg alkönyvtár található, mindegyikük a rendszermag különbözõ részeit testesíti meg. Ezek közül most számunkra a legfontosabb az architektúra/conf lesz, ahol majd létrehozzuk a saját rendszermagunk konfigurációs állományát, valamint a compile, ahol majd a rendszermagunk fordítása történik. Itt az *architektúra* lehet i386, alpha, amd64, ia64, powerpc, sparc64 vagy pc98 (a PC-k egyik, leginkább Japánban elterjedt változata). Az adott architektúra könyvtárában található összes állomány csak arra az architektúrára vonatkozik, a kód többi része pedig gépfüggetlen és közös az összes többi létezõ és leendõ FreeBSD platformon. Érdemes megfigyelni a könyvtárak logikai elrendezését: minden egyes ismert eszköz, állományrendszer és bõvítmény saját alkönyvtárral rendelkezik.

A példák során ez a fejezet feltételezi, hogy az i386 architektúrát használjuk. Ha ez a mi esetünkben nem így lenne, ne felejtsük el átírni bennük az elérési útvonalakat a rendszerünk architektúrájának megfelelõen.

Ha *nem lenne* /usr/src/sys könyvtár a rendszerünkben, valószínûleg még nem telepítettük a rendszermag forráskódját. Ezt a legkönnyebben úgy tudjuk megtenni, ha `root` felhasználóként elindítjuk a `sysinstall` programot és ott kiválasztjuk a Configure (Beállítások), azon belül Distributions (Terjesztések) menüpontot, amiben válasszuk ki a src, base és sys terjesztéseket. Ha nem szeretnénk erre a célra a sysinstall programot használni, de rendelkezésünkre áll a "hivatalos" FreeBSD CD, akkor a forrásokat akár parancssorból is telepíthetjük:

```
# mount /cdrom
# mkdir -p /usr/src/sys
# ln -s /usr/src/sys /sys
# cat /cdrom/src/ssys.[a-d]* | tar -xzvf -
# cat /cdrom/src/sbase.[a-d]* | tar -xzvf -
```

Ezután lépjünk be az i386/conf könyvtárba és másoljuk le a GENERIC konfigurációs állományt a kedvünk szerinti nevûre. Például:

```
# cd /usr/src/sys/i386/conf
# cp GENERIC SAJÁT
```

Általában a nevet végig nagybetûkkel írjuk, és ha több FreeBSD-s gépet is üzemeltetünk különbözõ hardverekkel, hasznosnak bizonyulhat megemlíteni benne az adott gép rendszerének nevét is. Ebben a példában ez most a SAJÁT lesz.

A rendszermagunk konfigurációs állományát nem éppen a legjobb ötlet a /usr/src könyvtárban tárolni. Ugyanis könnyen elõfordulhat, hogy egy rosszul sikerült fordítás után egyszerûen csak letöröljük az egész /usr/src könyvtárat és onnan kezdjük újra. Azonban csak ezután juthat eszünkbe, hogy vele együtt bizony letöröltük a saját rendszermagunk konfigurációs állományát is! Ehhez hasonlóan, közvetlenül a GENERIC konfigurációs állomány szerkesztése sem ajánlott, mivel a források egy esetleges frissítésénél könnyen felülíródhat és ezzel együtt elvesznek a módosításaink is.

Tehát érdemes inkább valahol máshol tárolnunk a rendszermagunk konfigurációs állományát, majd létrehozni rá egy szimbolikus linket a i386 könyvtárban.

Valahogy így:

```
# cd /usr/src/sys/i386/conf
# mkdir /root/kernel
# cp GENERIC /root/kernel/SAJÁT
# ln -s /root/kernel/SAJÁT
```

Most pedig a kedvenc szövegszerkesztõnkkel lássunk neki a SAJÁT átírásának! Ha nemrég telepítettük csak a rendszerünket, az egyetlen elérhetõ szövegszerkesztõnk minden bizonnyal a vi lesz. Róla most túlságosan is bonyolult lenne leírást adnunk, de az Irodalomjegyzékben található könyvek közül sokban elég jól bemutatják. Ezen kívül a FreeBSD ajánl egy könnyebben megtanulható szövegszerkesztõt is az ee személyében, amely a kezdõk számára az ideális választás. Nyugodtan átírhatjuk az elöl található megjegyzéseket a saját konfigurációnknak megfelelõen, vagy akár azt is rögzíthetjük, hogy miben tértünk el a GENERIC beállításaitól.

Ha fordítottunk már rendszermagot SunOS™ vagy más BSD operációs rendszer alatt, ez az állomány ismerõsnek tûnhet. Ha viszont más operációs rendszerek, mint például a DOS felõl érkezünk, a GENERIC konfigurációs állomány egy kissé terebélyesnek tûnhet számunkra, ezért A konfigurációs állomány címû részt figyelmesen és lassan olvassuk át.

Amennyiben a forrásfánkat a FreeBSD projekt legfrissebb forrásaival szinkronizáljuk, mindig olvassuk el a /usr/src/UPDATING állományt, mielõtt bármilyen frissítéshez is kezdenénk. Itt megtalálhatóak azok a fontos érintett kérdések és területek, amely külön figyelmet igényelnek a frissített forráskód esetén. A /usr/src/UPDATING mindig a FreeBSD forrásának legfrissebb változatához igazodik, és ezért sokkal naprakészebb információkat tartalmaz, mint ez a kézikönyv.

Most pedig le kell lefordítanunk a rendszermag forráskódját.

Procedure: A rendszermag lefordítása

1. Lépjünk be a /usr/src könyvtárba:

```
# cd /usr/src
```

2. Fordítsuk le a rendszermagot:

```
# make buildkernel KERNCONF=SAJÁT
```

3. Telepítsük az új rendszermagot:

```
# make installkernel KERNCONF=SAJÁT
```

A FreeBSD teljes forrásfájára szükség van a rendszermag lefordításához.

Amikor egy saját rendszermagot alapértelmezés szerint fordítunk, vele együtt az *összes* modul is lefordításra kerül. Ha viszont idõt szeretnénk megtakarítani a rendszermag frissítése során, vagy csak a saját moduljainkat akarjuk lefordítani, érdemes átírnunk az /etc/make.conf állományt a rendszermag fordításának megkezdése elõtt:

```
MODULES_OVERRIDE = linux acpi sound/sound sound/driver/ds1 ntfs
```

Ez a változó megadja a ténylegesen lefordítandó modulok listáját.

```
WITHOUT_MODULES = linux acpi sound ntfs
```

Ez a változó a fordításból kihagyandó felsõ szintû modulokat sorolja fel. A rendszermag fordításának folyamatában egyéb hasznosnak tekinthetõ változókról a make.conf(5) man oldalán olvashatunk.

Ezután az új rendszermag a /boot/kernel könyvtárba kerül /boot/kernel/kernel néven, a korábbi rendszermag pedig /boot/kernel.old/kernel néven õrzõdik meg. Most állítsuk le a rendszert és indítsuk újra az új rendszermag aktiválásához. Ha közben valamilyen hiba történt volna, nézzük meg a fejezet végén található, hibakeresésre vonatkozó utasításokat. Mindenképpen olvassuk el azt a részt, amely leírja, hogyan állítsuk helyre a rendszerünket abban az esetben, ha az új rendszermaggal nem indul.

A rendszerindítási folyamathoz tartozó további állományok, mint például a

rendszerbetöltõ (loader(8)) és annak konfigurációs állománya, a /boot könyvtárban találhatóak. A külsõ és saját modulok a /boot/kernel a könyvtárba kerülhetnek, azonban a felhasználóknak nagyon ügyelniük kell rá, hogy az itt található modulok szinkronban legyenek a lefordított rendszermaggal. Ellenkezõ esetben a rendszerben megbízhatatlanságot, hibákat észlelhetünk.

8.6. A konfigurációs állomány

A konfigurációs állomány általános formátuma igen egyszerû. Minden sor tartalmaz egy kulcsszót és egy vagy több paramétert. A további egyszerûsítés kedvéért a legtöbb sor csak egyetlen paramétert tartalmaz. Bármi, ami egy # (kettõskereszt) jelet követ, megjegyzésnek minõsül és nem számít konfigurációs elemnek. A most következõ részek bemutatják az egyes kulcsszavakat abban a sorrendben, ahogy azokat a GENERIC állományban is megtalálhatjuk. Az architektúrafüggõ opciók és eszközök teljes listáját a GENERIC állománnyal egy könyvtárban levõ NOTES állományban találhatjuk meg. Az architektúrától független opciókat a /usr/src/sys/conf/NOTES állományban találjuk.

A FreeBSD 5.0 megjelenése óta a konfigurációs állományokban használható az `include` direktíva. Ennek segítségével egy másik konfigurációs állomány tartalma logikailag beilleszthetõ az aktuálisba, így könnyebbé válik egy már meglevõ állományhoz tartozó kisebb mennyiségû változtatás karbantartása. Például ha csupán pár egyszerû kiegészítést szeretnénk hozzáadni a GENERIC rendszermaghoz, akkor elegendõ a hozzá vett eltéréseket nyilvántartanunk egy külön konfigurációs állományban:

```
include GENERIC
ident SAJAT

options         IPFIREWALL
options         DUMMYNET
options         IPFIREWALL_DEFAULT_TO_ACCEPT
options         IPDIVERT
```

Valószínûleg sok rendszergazda számára jelentõs elõnyt jelent ez a megoldás a konfigurációs állományok korábbról már megszokott újraírásával szemben: a helyi konfigurációs állomány csak a GENERIC rendszermag helyi rendszerre vonatkozó eltéréseit tartalmazza. Így amikor frissítjük a rendszerünket, a GENERIC rendszermag összes újítása elérhetõvé válik, kivéve ha explicit módon le nem tiltottuk ezeket a `noptions` vagy a `nodevice` megadásával. A fejezet további részében egy átlagos konfigurációs állománnyal fogunk foglalkozni, mind a beállítások, mind pedig az eszközök tekintetében.

Ha olyan állományt akarunk készíteni, amely tartalmazza az összes lehetséges opciót, például teszteléshez, futtassuk le `root` felhasználóként az alábbi parancsot:

```
# cd /usr/src/sys/i386/conf && make LINT
```

Itt a GENERIC rendszermag-konfigurációs állomány ismertetése következik, az érthetõség kedvéért

helyenként megjegyzésekkel kibõvítve. A bemutatott állománynak majdnem pontosan meg kell egyeznie a rendszerünkben található /usr/src/sys/i386/conf/GENERIC állománnyal.

```
machine      i386
```

A számítógépünk architektúráját adja meg. A következõk valamelyikének kell lennie: `alpha`, `amd64`, `i386`, `ia64`, `pc98`, `powerpc`, vagy `sparc64`.

```
cpu          I486_CPU
cpu          I586_CPU
cpu          I686_CPU
```

A fenti beállítás segítségével megadhatjuk, milyen típusú processzor található a számítógépünkben. Több ilyen sorunk is lehet (ha például nem lennénk biztosak benne, hogy az `I586_CPU` vagy `I686_CPU` értéket kellene megadnunk), de a saját rendszermagunk összeállításához érdemes csak egyet meghagynunk. Ha nem ismerjük pontosan a processzorunk típusát, vessünk egy pillantást a /var/run/dmesg.boot állományra és keressük ki belõle.

```
ident          GENERIC
```

Ez a rendszermag azonosítója. Változtassuk meg rendszermagunk nevére, legyen például `SAJAT`, ha a korábbi utasításokat követtük. Az `ident` után írt sztring fog megjelenni a rendszermag neve mellett a rendszer indítása során, ezért fontos, hogy az új rendszermagunknak más nevet adjunk, ha meg akarjuk különböztetni az általában használttól (például egy tesztelésre szánt rendszermagot akarunk készíteni).

```
# ha a /boot/device.hints használata helyett statikusan bele akarjuk fordítani
#hints          "GENERIC.hints"         # itt szerepelnek a device hintek
```

A device.hints(5) használható az eszközmeghajtók beállítására. A loader(8) a rendszer indítása során alapértelmezés szerint a /boot/device.hints állományt olvassa be erre a célra. A `hints` beállítás használatával ezeket a "hinteket" statikusan bele tudjuk építeni a rendszermagba. Ebben az esetben nincs szükségünk külön device.hints állomány létrehozására a /boot könyvtárban.

```
makeoptions     DEBUG=-g         # a nyomkövetéshez szükséges gdb(1) szimbólumok
beépítése
```

A FreeBSD hagyományos fordításának folyamata során a rendszermagot a `-g` használatával készítjük el, aminek köszönhetõen hibakeresési információkat tudunk átadni a gcc(1) fordítónak.

```
options          SCHED_ULE          # ULE ütemezõ
```

A FreeBSD alapértelmezett rendszerütemezõje. Ne változtassuk meg!

```
options         PREEMPTION        # a rendszerszálak megszakíthatóságának
engedélyezése
```

Ha engedélyezzük, a rendszermagban futó szálakat meg tudják szakítani más, magasabb prioritású szálak. Ez segít növelni a rendszer válaszadási sebességét és csökkenti a megszakításokat kezelõ szálak várakozását.

```
options         INET              # hálózatkezelés
```

A hálózatkezelés támogatása. Ne töröljük ki, még akkor sem, ha nem tervezzük hálózatra kapcsolni a rendszert. Sok programnak szüksége van legalább az ún. loopback típusú hálózat támogatására (vagyis a számítógépünkön belüli hálózati kapcsolatokra), ezért ez feltétlenül kötelezõ!

```
options         INET6             # IPv6 kommunikációs prokotollok
```

Engedélyezi az IPv6 kommunikációs protokollok használatát.

```
options         FFS               # Berkeley Fast Filesystem
```

Ez a legalapvetõbb merevlemezes állományrendszer. Hagyjuk meg, ha merevlemezrõl akarjuk indítani a rendszerünket.

```
options         SOFTUPDATES       # az FFS Soft Updates támogatása
```

Ez a beállítás engedélyezi a rendszermagban a Soft Updates használatát, amely segít felgyorsítani a lemez írási sebességét. Ha már a rendszermag ezt a funkcionalitást ismeri, akkor még külön az egyes lemezeken is engedélyezni kell. Nézzük meg a mount(8) kimenetét, hogy lássuk, a rendszerünkben levõ lemezek közül melyiken van ténylegesen engedélyezve a Soft Updates használata. Ha nem látjuk benne sehol sem a `soft-updates` opciót, akkor azt (meglevõ állományrendszerek esetén) a tunefs(8) vagy (új állományrendszerek esetén) a newfs(8) parancsokkal tudjuk bekapcsolni.

```
options         UFS_ACL           # a hozzáférés-vezérlési listák (ACL) támogatása
```

Ezzel a beállítással engedélyezhetjük a rendszermagban a hozzáférés-vezérlési listák támogatását. Ez a kiterjesztett attribútumok és az UFS2 használatára támaszkodik. Ezt a lehetõséget részleteiben a Az állományrendszerek hozzáféréseit vezérlő listákban tárgyaljuk. Az ACL alapértelmezés szerint támogatott, és ha korábban már használtuk, akkor semmiképpen se kapcsoljuk ki, mert ezzel az eddig létrehozott hozzáférés-vezérlési listáink érvénytelenné, az állományaink pedig védtelenné válnak.

```
options         UFS_DIRHASH      # nagyobb könyvtárak esetén gyorsulást hoz
```

Ezzel a beállítással némi memória feláldozása árán fel tudjuk gyorsítani a nagyobb könyvtárakon végzett lemezmûveletek sebességét, ezért ezt a beállítást érdemes nagyobb szerverekre vagy interaktivitást igénylõ munkaállomásokra tartogatni, és eltávolítani olyan esetekben, amikor a FreeBSD-t olyan kisebb számítógépeken használjuk, ahol a memória kevés és a lemezmûveletek sebessége kevésbé fontos, például egy tûzfalon.

```
options         MD_ROOT          # tudunk memórialemezrõl is rendszert indítani
```

Ezzel az opcióval engedélyezni tudjuk a rendszer indítását memóriában tárolt virtuális lemezekrõl.

```
options         NFSCLIENT        # hálózati állományrendszer (NFS) kliens
options         NFSSERVER        # NFS szerver
options         NFS_ROOT         # NFS használható gyökérként is, kell hozzá az
NFSCLIENT
```

A hálózati állományrendszer támogatása. Hacsak nem akarunk TCP/IP-n keresztül állományrendszereket csatlakoztatni egy UNIX® állományszerverrõl, kivehetjük.

```
options         MSDOSFS          # MS-DOS állományrendszer
```

Az MS-DOS® állományrendszer. Hacsak nem akarunk DOS-ra formázott merevlemezes partíciót csatlakoztatni a rendszerindítás során, nyugodtan elhagyhatjuk. A fentebb leírtak szerint az elsõ olyan alkalommal automatikusan betöltõdik, amikor egy DOS partíciót csatlakoztatni akarunk. Sõt, a nagyszerû emulators/mtools szoftver segítségével külön csatlakoztatás és leválasztás nélkül tudunk DOS-os floppykat olvasni (és az MSDOSFS-re egyáltalán nincs is szüksége).

```
options         CD9660           # ISO 9660 állományrendszer
```

Az ISO 9660 állományrendszert a CD-k használják. Vegyük ki, ha nincs a számítógépben CD-ROM meghajtó, vagy csak ritkán fogunk CD-ket csatlakoztatni (mivel a hozzá tartozó modul magától betöltõdik az elsõ adat CD csatlakoztatása során). Az audio CD-k nem használják ezt az állományrendszert.

```
options         PROCFS           # a futó programok állományrendszere (szükséges
hozzá a PSEUDOFS)
```

A futó programok állományrendszere. Ez csak a /proc könyvtárra csatlakoztatott "színlelt" állományrendszer, amelynek segítségével a ps(1) és hozzá hasonló programok képesek több információt adni a futó programokról. A PROCFS használata a legtöbb esetben nem indokolt, mivel a különféle nyomkövetõ és felügyeleti eszközök képesek a PROCFS használata nélkül is mûködni:

alapértelmezés szerint a telepített rendszerek sem csatlakoztatják ezt az állományrendszer.

```
options         PSEUDOFS        # pszeudo állományrendszerek támogatása
```

A 6.X verziójú rendszermagokban a PROCFS használatához engedélyeznünk kell a PSEUDOFS használatát is.

```
options         GEOM_GPT        # GUID típusú partíciós táblák használata
```

Ezzel a beállítással engedélyezni tudjuk nagy mennyiségû partíció támogatását egyetlen lemezen.

```
options         COMPAT_43       # kompatibilitás fenntartása a 4.3 BSD-vel [NE
TÖRÖLD!]
```

Kompatibilitás a 4.3BSD-vel. Ne vegyük ki, mert bizonyos programok furcsán fognak viselkedni a hiánya esetén.

```
options         COMPAT_FREEBSD4   # kompatibilitás a FreeBSD4-el
```

Ez a beállítás szükséges a FreeBSD 5.X i386™ és Alpha rendszerein a FreeBSD korábbi verzióihoz fordított alkalmazások támogatásához, melyek régebbi rendszerhívásokat használnak. Az összes i386™ és Alpha típusú rendszeren ajánlott engedélyezni, mivel itt elõfordulhatnak régebbi alkalmazások. A többi platform, mint például az ia64 vagy a sparc64, támogatása csak az 5.X verzióban jelent meg, ezért ott nincs szükség erre.

```
options         COMPAT_FREEBSD5   # kompatibilitás a FreeBSD5-el
```

Ezt a beállítást a FreeBSD 6.X és afeletti verziókban kell használni az olyan FreeBSD 5.X verziókra fordított alkalmazások futtatásának támogatásához, melyek a FreeBSD 5.X rendszerhívásait használják.

```
options         SCSI_DELAY=5000  # a SCSI eszközök keresése elõtt késleltetés
(ezredmásodpercben)
```

Ezzel a beállítással a rendszermag 5 másodpercig várakozni fog a SCSI eszközök keresése elõtt. Ha kizárólag csak IDE típusú merevlemezeink vannak, nyugodtan kihagyhatjuk, máskülönben érdemes a rendszerindítás gyorsítása érdekében csökkenteni ezt az értéket. Természetesen, ha így teszünk és a FreeBSD nem tudja felismerni a SCSI eszközeinket, akkor növeljük meg valamennyivel.

```
options         KTRACE          # a ktrace(1) támogatása
```

Engedélyezi a rendszermagban futó rutinok nyomonkövetését, ami hasznos lehet a hibák keresése

során.

```
options         SYSVSHM         # SYSV-szerû osztott memória
```

Ezzel a beállítással engedélyezni tudjuk a rendszerben a System V típusú osztott memória használatát. Leggyakrabban az X rendszer XSHM kiterjesztése használja, amelyen keresztül számos mûveletigényes grafikus program mûködését fel lehet gyorsítani. Ha X-et használunk, mindenképpen szükségünk lehet erre.

```
options         SYSVMSG         # SYSV-szerû üzenetsorok
```

A System V üzenetek támogatása. Ez a beállítás csupán néhány száz byte-tal növeli a rendszermagot.

```
options         SYSVSEM         # SYSV-szerû szemaforok
```

A System V szemaforok támogatása. Nem túl gyakran alkalmazzák ezeket, de ez csak néhány száz byte-ot tesz hozzá a rendszermaghoz.

A ipcs(1) parancs -p paraméterével ki tudjuk listáztatni azokat a futó programokat, amelyek ezen System V eszközöket használják.

```
options         _KPOSIX_PRIORITY_SCHEDULING # POSIX P1003_1B valósidejû
kiterjesztések
```

A POSIX® 1993-as változatában megjelent valósidejû bõvítések. A Portgyûjteményben megjelenõ egyes alkalmazások használják ezeket (mint például a StarOffice™).

```
options         KBD_INSTALL_CDEV  # CDEV bejegyzés létrehozása a /dev könyvtárban
```

Ez a beállítás kell ahhoz, hogy a /dev könyvtárban létre tudjunk hozni eszközleírókat a billentyûzethez.

```
options         ADAPTIVE_GIANT    # adaptív Giant mutexek
```

A Giant annak a kölcsönös kizárási mechanizmusnak (blokkolt mutexnek) a neve, amely a rendszermag erõforrásainak jelentõs részét védi. Manapság ez már egy elfogadhatatlanul szûk keresztmetszetet képez a teljesítményben, ezért a fejlesztésben fokozatosan felváltják az egyes erõforrásokat külön-külön védõ zárolások. Az ADAPTIVE_GIANT beállítás hatására a Giant a helyzethez igazodóan forgó (spin) mutexek közé kerül. Ez azt jelenti, hogy amikor egy szál zárolni akarja a Giant mutexet, de ezt már megtette elõtte egy másik processzorról futó szál, a szál tovább fut és várakozni fog a zárolás feloldására. Normális esetben ugyanis egy szál továbbra is blokkolt

állapotban marad, várakozva a futásra. Ha nem tudunk dönteni, hagyjuk változatlanul.

Hozzátesszük, hogy a FreeBSD 8.0-CURRENT és későbbi változataiban az össszes mutex alapértelmezés szerint adaptív, hacsak meg nem adjuk a `NO_ADAPTIVE_MUTEXES` beállítást. Ennek eredményeképpen a Giant most már alapból adaptív, ezért esetükben az `ADAPTIVE_GIANT` nem szerepel a rendszermag beállításai között.

```
device          apic              # I/O APIC
```

Az apic nevû eszköz engedélyezésével használhatjuk a hardveres APIC-ot a megszakítások vezérlésére. Az apic alkalmazható egy- és többprocesszoros rendszerek esetén is egyaránt, de az SMP rendszermagoknál szükséges. Több processzor támogatásánál mindenképpen tegyük hozzá az `options SMP` beállítást is.

Az apic eszköz csak az i386 architektúrán létezik, ezért a többi architektúrán nem szabad használnunk ezt a beállítást.

```
device          eisa
```

Abban az esetben engedélyezzük, ha EISA-s alaplapunk van, ezzel aktiváljuk az EISA buszra csatlakoztatott eszközök automatikus felismerését és beállíthatóságát.

```
device          pci
```

Tegyük hozzá a konfigurációs állományhoz, ha PCI-os alaplapuk van. Ezzel engedélyezhetjük a PCI kártyák automatikus felismerését és a PCI és ISA buszok közti átirányítást.

```
# Hajlékonylemezes meghajtók
device          fdc
```

Ez a hajlékonylemezes meghajtó vezérlõje.

```
# ATA és ATAPI eszközök
device          ata
```

Ez az eszközmeghajtó felelõs az összes ATA és ATAPI eszközért. A modern számítógépeken csak egyszer kell megadnunk a `device ata` sort a beállítások között az összes PCI-os ATA/ATAPI eszköz felismeréséhez.

```
device          atadisk               # ATA lemezmeghajtók
```

Az ATA lemezmeghajtók támogatásához erre van még szükség a `device ata` mellett.

```
device          ataraid                 # ATA RAID-meghajtók
```

Az ATA RAID-meghajtók kezeléséhez erre a sorra van szükség a `device ata` mellett.

```
device          atapicd                 # ATAPI CD-meghajtók
```

Az ATAPI CD-meghajtók használatához ezt is tegyük a konfigurációba a `device ata` mellé.

```
device          atapifd                 # ATAPI floppy meghajtók
```

A `device ata` használata mellett erre van még szükségünk az ATAPI floppy meghajtók kezeléséhez.

```
device          atapist                 # ATAPI szalagos meghajtók
```

Az ATAPI szalagos egységek használatához ezt a sort is tegyük a konfigurációba a `device ata` mellé.

```
options         ATA_STATIC_ID           # statikus eszközszámozás
```

Ezzel a beállítással a vezérlõk számozása állandó lesz. Nélküle az eszközszámok dinamikusan kerülnek kiosztásra.

```
# SCSI vezérlõk
device          ahb         # EISA AHA1742 család
device          ahc         # AHA2940 és integrált AIC7xxx eszközök
options         AHC_REG_PRETTY_PRINT    # a hibák kereséséhez kiíratja a regiszterek
                                        # bitmezõit.  Kb. 128 KB-al növeli a méretét.
device          ahd         # AHA39320/29320 és integrált AIC79xx eszközök
options         AHD_REG_PRETTY_PRINT    # a hibák kereséséhez kiíratja a regiszterek
                                        # bitmezõit.  Kb. 215 KB-al növeli a méretét.
device          amd         # AMD 53C974 (Teckram DC-390(T))
device          isp         # Qlogic család
#device         ispfw       # a QLogic HBA firmware-e, többnyire modul
device          mpt         # LSI-Logic MPT-Fusion
#device         ncr         # NCR/Symbios Logic
device          sym         # NCR/Symbios Logic (újabb chipsetek, illetve az `ncr'
típusúak)
device          trm         # Tekram DC395U/UW/F DC315U csatolók

device          adv         # Advansys SCSI-csatolók
device          adw         # Advansys wide SCSI-csatolók
device          aha         # Adaptec 154x SCSI-csatolók
device          aic         # Adaptec 15[012]x SCSI-csatolók, AIC-6[23]60.
```

```
device          bt          # Buslogic/Mylex MultiMaster SCSI-csatolók

device          ncv         # NCR 53C500
device          nsp         # Workbit Ninja SCSI-3
device          stg         # TMC 18C30/18C50
```

SCSI-vezérlõk. Vegyük ki azokat, amelyekkel ténylegesen nem rendelkezünk. Ha csak IDE eszközeink vannak a rendszerünkben, az összeset eltávolíthatjuk. A `_REG_PRETTY_PRINT` végzõdésû sorok a megfelelõ meghajtók hibakerési beállításait takarják.

```
# SCSI-perifériák
device          scbus       # SCSI-busz (kell a SCSI-hoz)
device          ch          # SCSI médiumváltók (media changer)
device          da          # közvetlen hozzáférés (lemezek)
device          sa          # soros hozzáférés (szalag stb.)
device          cd          # CD
device          pass        # áteresztõ eszköz (közvetlen SCSI hozzáférés)
device          ses         # SCSI környezeti szolgáltatások (és SAF-TE)
```

SCSI-perifériák. Itt is érvényes, hogy kivehetjük azokat az eszközöket, amelyekkel nem rendelkezünk. De ha csak IDE hardvereink vannak, teljesen eltávolíthatjuk ezeket.

Annak ellenére, hogy valójában nem igazi SCSI-eszközök, az USB-s umass(4) és még néhány más egyéb meghajtó is használja a SCSI alrendszert. Emiatt semmiképpen se távolítsuk el a SCSI támogatást a rendszerünkõl abban az esetben, ha ilyen meghajtókat is használni szándékozunk.

```
# a SCSI alrendszerhez kapcsolódó RAID-vezérlõk
device          amr         # AMI MegaRAID
device          arcmsr      # Areca SATA II RAID
device          asr         # DPT SmartRAID V, VI és Adaptec SCSI RAID
device          ciss        # Compaq Smart RAID 5*
device          dpt         # DPT Smartcache III, IV - lásd a NOTES állományt
device          hptmv       # Highpoint RocketRAID 182x
device          rr232x      # Highpoint RocketRAID 232x
device          iir         # Intel Integrated RAID
device          ips         # IBM (Adaptec) ServeRAID
device          mly         # Mylex AcceleRAID/eXtremeRAID
device          twa         # 3ware 9000 series PATA/SATA RAID

# RAID vezérlõk
device          aac         # Adaptec FSA RAID
device          aacp        # SCSI áteresztõ az aac-hez (kell hozzá a CAM)
device          ida         # Compaq Smart RAID
device          mfi         # LSI MegaRAID SAS
device          mlx         # Mylex DAC960 család
device          pst         # Promise Supertrak SX6000
```

```
device          twe         # 3ware ATA RAID
```

Az ismert RAID-vezérlõk. Ha közülük egyikkel sem rendelkezünk, távolítsuk el ezeket a konfigurációból.

```
# az atkbdc0 vezérli a billentyûzetet és a PS/2-es egeret
device          atkbdc      # AT billentyûzet vezérlõ
```

A billentyûzet vezérlõje (atkbdc) az AT-s billentyûzet és a PS/2 stílusú pozícionáló eszközök vezérléséhez szükséges I/O szolgáltatásokat biztosítja. Erre a vezérlõre a billentyûzet meghajtójának (atkbd) és a PS/2 pozícionáló eszközök eszközmeghajtójának (psm) is szüksége van.

```
device          atkbd       # AT billentyûzet
```

Az atkbd meghajtó, a atkbdc vezérlõvel együtt, adja a hozzáférést az AT billentyûzet vezérlõre csatlakoztatott AT 84 és a fejlettebb AT billentyûzetek felé.

```
device          psm         # PS/2 egér
```

Használjuk ezt az eszközt, ha az egerünk a PS/2 portra csatlakozik.

```
device          kbdmux         # billentyûzet multiplexer
```

A billentyûzet multiplexer alapszintû támogatása. Ha nem kívánunk a jövõben egynél több billentyûzetet csatlakoztatni a rendszerünkre, nyugodt szívvel kivehetjük ezt a sort.

```
device          vga         # VGA videokártya meghajtó
```

Videokártya meghajtó.

```
device          splash      # üdvözlõképernyõk és képernyõkímélõk támogatása
```

Nyissunk egy üdvözlõképernyõvel! A képernyõkímélõknek is szükségük van erre az eszközre.

```
# a syscons az alapértelmezett konzolmeghajtó, hasonlít a SCO konzolra
device          sc
```

Az sc az alapértelmezett meghajtó a konzolok számára, és sokban hasonlít a SCO konzolra. Mivel a legtöbb teljesképernyõs program a termcap terminációadatbázis könyvtáron keresztül éri el a konzolt, nem igazán számít, hogy ezt vagy a VT220-kompatibilis vt konzolmeghajtót használjuk. Ha bármilyen gondunk lenne a teljesképernyõs programok futtatásával ezen a konzolon, a

bejelentkezéskor állítsuk a TERM környezeti változónkat a scoansi értékre.

```
# ezzel tudjuk engedélyezni a pcvt (VT220-kompatibilis) konzolmeghajtót
#device          vt
#options         XSERVER          # az X szerver támogatása vt konzolon
#options         FAT_CURSOR       # telt kurzor használata
```

Ez a VT220-kompatibilis konzolmeghajtó, amely visszafelé kompatibilis a VT100/102-vel is. Remekül mûködik olyan laptopokon, ahol a hardver nem használható az sc konzollal. Itt ugyanúgy érdemes egyébként a vt100 értékre vagy a vt220 értékre állítani a TERM környezeti változónkat. Hasznosnak bizonyulhat abban az esetben is, amikor hálózaton keresztül nagy mennyiségû és eltérõ típusú számítógépekhez csatlakozunk, és ahol a termcap és terminfo adatbázisokban az sc bejegyzései gyakran nem is érhetõek el - a vt100 viszont virtuálisan az összes platformon elérhetõ.

```
device          agp
```

Írjuk bele a konfigurációba, ha van AGP kártya a rendszerünkben. Ezzel engedélyezzük az AGP és az AGP GART támogatását az ezeket ismerõ kártyák számára.

```
# energiagazdálkodás támogatása (bõvebben lásd: NOTES)
#device          apm
```

A fejlett energiagazdálkodás támogatása. Laptopok esetén hasznos, habár ez alapértelmezés szerint nincs engedélyezve a GENERIC konfigurációban.

```
# az i8254 készenléti módjának támogatása
device          pmtimer
```

Az energiagazdálkodási események, mint például APM és ACPI idõzítõjének eszközmeghajtója.

```
# PCCARD (PCMCIA) támogatás
# PCMCIA és cardbus támogatás
device          cbb             # cardbus (yenta) bridge
device          pccard          # PC Card (16 bites) busz
device          cardbus         # CardBus (32 bites) busz
```

A PCMCIA támogatása. Mindenképpen szükségünk lesz rá, ha laptopunk van.

```
# soros (COM) portok
device          sio             # 8250, 16[45]50 alapú soros portok
```

Ezek azok a soros portok, amelyek az MS-DOS®/Windows® világban csak COM portokként ismernek.

Ha van egy belsõ modemünk a COM4-en és egy soros portunk a COM2-n, a modem IRQ-ját meg kell változtatnunk 2-re (valamilyen homályos mûszaki okból kifolyólag a COM2 = IRQ9), hogy hozzá tudjunk férni FreeBSD-bõl. Ha többportos soros kártyánk lenne, lapozzuk fel a sio(4) man oldalát, és ott hozzá megtaláljuk a /boot/device.hints állományba írandó megfelelõ értékeket. Egyes videokártyák (különösen az S3 chipekre épülõk) az I/O címeket `0x*2e8` alakban használják, és mivel rengeteg olcsó soros kártya nem kódolja vissza egészében a 16 bites I/O címteret, ütközni fognak ezekkel a kártyákkal, és ezáltal a COM4 port gyakorlatilag elérhetetlenné válik.

Minden egyes soros portnak egyedi IRQ-ja kell legyen (hacsak nem használunk olyan többportos kártyát, amely támogatja a megosztott megszakításokat), ezért a COM3 és COM4 esetén alapértelmezett IRQ-k nem használhatóak.

```
# párhuzamos port
device          ppc
```

Ez az ISA busz párhuzamos portjának felülete.

```
device          ppbus      # a párhuzamos port busza (kell)
```

A párhuzamos porthoz tartozó busz támogatása.

```
device          lpt        # nyomtató
```

A párhuzamos portra csatlakozó nyomtatók támogatása.

A fentiek közül mind a három szükséges a párhuzamos porton csatlakozó nyomtatók használatához.

```
device          plip       # TCP/IP párhuzamos porton keresztül
```

Ez a párhuzamos port hálózati felületének meghajtója.

```
device          ppi        # a párhuzamos port felületének eszköze
```

Általános célú ("geek port") és IEEE1284 I/O.

```
#device         vpo        # az scbus és a da kell a használatához
```

Ez az Iomega Zip meghajtóihoz tartozó eszköz. A mûködéséhez szükség van az `scbus` és `da` engedélyezésére. A legjobb teljesítményt EPP 1.9 módban mûködõ portokkal lehet kihozni belõle.

```
#device         puc
```

Tegyük bele a konfigurációba ezt az eszközt, ha egy olyan "buta" soros vagy párhuzamos PCI kártyánk van, amelyet a puc(4) segédmeghajtó ismer.

```
# PCI Ethernet kártyák
device          de          # DEC/Intel DC21x4x (Tulip)
device          em          # Intel PRO/1000 Gigabit Ethernet kártya
device          ixgb        # Intel PRO/10GbE Ethernet kártya
device          txp         # 3Com 3cR990 (Typhoon)
device          vx          # 3Com 3c590, 3c595 (Vortex)
```

Különféle PCI hálózati kártyák meghajtói. Vegyük ki azokat, amelyek nem találhatóak meg a rendszerünkben.

```
# PCI Ethernet kártyák, melyek az MII busz vezérlõkódját használják
# FIGYELEM: Ne töröljük ki a 'device miibus' sort, ha ilyen kártyánk van!
device          miibus      # az MII busz támogatása
```

Az MII busz engedélyezése elengedhetetlen bizonyos 10/100-as PCI Ethernet kártyák használatához, konkrétan azokéhoz, amelyek az MII-vel együttmûködni képes adó-vevõt használnak vagy az MII-höz hasonló adó-vevõ vezérlõ felületet valósítanak meg. A `device miibus` hozzáadása a rendszermaghoz magával vonja az általános miibus API és az összes PHY meghajtó támogatását, beleértve azt az általános PHY eszközt is, amelyet az egyes eszközmeghajtók külön nem támogatnak.

```
device          bce         # Broadcom BCM5706/BCM5708 Gigabit Ethernet
device          bfe         # Broadcom BCM440x 10/100 Ethernet
device          bge         # Broadcom BCM570xx Gigabit Ethernet
device          dc          # DEC/Intel 21143 és egyéb hasonlóak
device          fxp         # Intel EtherExpress PRO/100B (82557, 82558)
device          lge         # Level 1 LXT1001 gigabit ethernet
device          msk         # Marvell/SysKonnect Yukon II Gigabit Ethernet
device          nge         # NatSemi DP83820 gigabit ethernet
device          nve         # nVidia nForce MCP integrált Ethernet hálózat
device          pcn         # AMD Am79C97x PCI 10/100 (az 'lnc' elõtt)
device          re          # RealTek 8139C+/8169/8169S/8110S
device          rl          # RealTek 8129/8139
device          sf          # Adaptec AIC-6915 (Starfire)
device          sis         # Silicon Integrated Systems SiS 900/SiS 7016
device          sk          # SysKonnect SK-984x & SK-982x gigabit Ethernet
device          ste         # Sundance ST201 (D-Link DFE-550TX)
device          stge        # Sundance/Tamarack TC9021 gigabit Ethernet
device          ti          # Alteon Networks Tigon I/II gigabit Ethernet
device          tl          # Texas Instruments ThunderLAN
device          tx          # SMC EtherPower II (83c170 EPIC)
```

```
device          vge        # VIA VT612x gigabit ethernet
device          vr         # VIA Rhine, Rhine II
device          wb         # Winbond W89C840F
device          xl         # 3Com 3c90x (Boomerang, Cyclone)
```

Meghajtók, melyek az MII busz vezérlõkódját használják.

```
# ISA Ethernet és pccard hálózati kártyák.
device          cs         # Crystal Semiconductor CS89x0 NIC
# az 'device ed' eszközhöz kell a 'device miibus'
device          ed         # NE[12]000, SMC Ultra, 3c503, DS8390 cards
device          ex         # Intel EtherExpress Pro/10 és Pro/10+
device          ep         # Etherlink III alapú kártyák
device          fe         # Fujitsu MB8696x alapú kártyák
device          ie         # EtherExpress 8/16, 3C507, StarLAN 10 stb.
device          lnc        # NE2100, NE32-VL Lance Ethernet kártyák
device          sn         # az SMC 9000-res sorozatú Ethernet chipjei
device          xe         # Xircom pccard Ethernet

# ISA eszközök, melyek a régi ISA betétet használják
#device         le
```

ISA Ethernet meghajtók. A konkrétan támogatott kártyák teljes felsorolását lásd a /usr/src/sys/i386/conf/NOTES állományban.

```
# vezeték nélküli hálózati kártyák
device          wlan            # 802.11 támogatás
```

Általános 802.11 támogatás. Erre a sorra mindenképpen szükség van a vezeték nélküli hálózatok használatához.

```
device          wlan_wep        # 802.11 WEP támogatás
device          wlan_ccmp       # 802.11 CCMP támogatás
device          wlan_tkip       # 802.11 TKIP támogatás
```

A 802.11 eszközök esetén a titkosítás támogatása. Ezeket a sorokat akkor adjuk meg, ha titkosítást akarunk használni vagy a 802.11i biztonsági protokolljait.

```
device          an         # Aironet 4500/4800 802.11 vezeték nélküli hálózati kártyák
device          ath             # Atheros pci/cardbus hálózati kártyák
device          ath_hal         # Atheros HAL (Hardware Access Layer)
device          ath_rate_sample # küldési mintavételi vezérlés az ath-hoz
device          awi        # BayStack 660 és mások
device          ral        # Ralink Technology RT2500 vezeték nélküli hálózati kártyák
device          wi         # WaveLAN/Intersil/Symbol 802.11 vezeték nélküli hálózati
kártyák
```

```
#device         wl         # régebbi, nem 802.11 Wavelan vezeték nélküli hálózati
kártyák
```

A különbözõ vezeték nélküli kártyák támogatása.

```
# Pszeudo eszközök
device   loop          # hálózati loopback
```

Ez a TCP/IP általános loopback eszköze. Ha telnettel vagy FTP-vel rácsatlakozunk a `localhost` címére (vagyis a `127.0.0.1`-re), akkor rajta keresztül saját magunkhoz jutunk vissza. Ennek a megléte *kötelezõ*!

```
device   random        # álvéletlenszám eszköz
```

Kriptográfiai szempontból biztonságos álvéletlenszám generátor.

```
device   ether         # Ethernet támogatás
```

Az `ether` eszközre csak abban az esetben van szükség, ha Ethernet kártyánk van. Ez magában foglalja az általános Ethernet protokoll kódját.

```
device   sl            # belsõ SLIP
```

Az `sl` a SLIP használatát engedélyezi. Ez egy régi protokoll, amelyet azóta már szinte teljesen kiszorított a PPP, mivel azt könnyebb beállítani és sokkal jobban is illik a modem-modem kapcsolatokhoz, illetve sokkal erõteljesebb.

```
device   ppp           # belsõ PPP
```

Ez a tárcsázós kapcsolatok rendszermagon belüli PPP támogatását adja meg. Van a PPP-nek egy külsõ, a felhasználói programként megvalósított változata is, amely a `tun` eszközt használja és sokkal nagyobb rugalmasságot kínál fel, illetve olyan lehetõségeket, mint például az igény szerinti tárcsázás.

```
device   tun           # csomag alagút
```

Ezt a felhasználói PPP szoftver használja. A könyv PPP-rõl szóló részében többet is megtudhatunk róla.

```
device   pty           # Pszeudo terminálok (telnet stb.)
```

Ezek a "pszeudo terminálok", vagy más néven szimulált bejelentkezési portok. A bejövõ `telnet` és `rlogin` munkamenetek használják, valamint az xterm és a hozzá hasonló alkalmazások, mint például az Emacs.

```
device   md            # memórialemezek
```

A memóriában levõ pszeudo lemezes meghajtók.

```
device   gif           # IPv6 és IPv4 tunnelek használata
```

Megvalósítja az IPv6 IPv4 feletti, az IPv4 IPv6 feletti, az IPv4 IPv4 feletti és az IPv6 IPv6 feletti közvetítését. A `gif` eszköz "magától másolódik", vagyis szükség szerint hozza létre a megfelelõ eszközleírókat.

```
device   faith         # IPv6-IPv4 közti továbbítás (fordítás)
```

Ez a pszeudo eszköz elfogja a hozzá küldött csomagokat és átadja ezeket az IPv4/IPv6 fordítással foglalkozó démonnak.

```
# a `bpf' eszköz használatával a Berkeley csomagszûrõt (Berkeley Packet Filter)
engedélyezzük
# Legyünk rá tekintettel, hogy ennek komoly következményei lehetnek
# rendszeradminisztrációs szempontból!
# A 'bpf'-re szükség van a DHCP-hez.
device   bpf           # Berkeley csomagszûrõ
```

A Berkeley csomagszûrõje. Ez egy olyan pszeudo eszköz, amely lehetõvé teszi, hogy a hálózati csatolók forgalmát megfigyeljük, mivel a (pl. Ethernet) hálózatunkon minden csomagot elkap. Ezek a csomagok lemezre is menthetõek vagy kielemezhetõek a tcpdump(1) program segítségével.

A bpf(4) eszközt a dhclient(8) is használja többek közt az alapértelmezett átjáró IP-címének megszerzéséhez. Ha DHCP-t akarunk használni, hagyjuk így.

```
# USB támogatás
device          uhci          # UHCI PCI->USB felület
device          ohci          # OHCI PCI->USB felület
device          ehci          # EHCI PCI->USB felület (USB 2.0)
device          usb           # USB busz (kell)
#device         udbp          # USB Double Bulk Pipe eszközök
device          ugen          # általános
device          uhid          # Human Interface Devices
device          ukbd          # billentyûzet
device          ulpt          # nyomtató
device          umass         # lemez/háttértároló - kell hozzá az scbus és a da
device          ums           # egér
```

```
device          ural          # Ralink Technology RT2500USB vezeték nélküli hálózati
kártyák
device          urio          # Diamond Rio 500 MP3 lejátszó
device          uscanner      # lapolvasók
# USB Ethernet, kell hozzá az mii
device          aue           # ADMtek USB Ethernet
device          axe           # ASIX Electronics USB Ethernet
device          cdce          # általános USB, Etherneten keresztül
device          cue           # CATC USB Ethernet
device          kue           # Kawasaki LSI USB Ethernet
device          rue           # RealTek RTL8150 USB Ethernet
```

A különféle USB eszközök támogatása.

```
# FireWire támogatás
device          firewire      # FireWire buszkód
device          sbp           # SCSI FireWire-ön keresztül (kell hozzá az scbus és a
da)
device          fwe           # Ethernet FireWire-ön keresztül (nem szabványos!)
```

A különféle Firewire eszközök támogatása.

A FreeBSD által ismert további eszközökrõl a /usr/src/sys/i386/conf/NOTES állományból tájékozódhatunk.

8.6.1. Sok memória kezelése (PAE)

A sok memóriával rendelkezõ számítógépek esetén szükség lehet a felhasználói és rendszerszintû virtuális címek (Kernel Virtual Address, KVA) 4 gigabyte feletti használatára. Ennek a korlátozásnak a kiküszöbölésére az Intel® külön támogatást épített be a Pentium® Pro és az azt követõ processzorok 36 bites fizikai címzésének kialakításához.

A Fizikai Címkiterjesztés (Physical Address Extension, PAE) az Intel® Pentium® Pro és késõbbi processzoraiban található meg, és lehetõvé teszi egészen 64 gigabyte-ig a memóriahasználatot. A FreeBSD is támogatja ezt a tulajdonságot a PAE rendszermag beállítás használatával, és megtalálható a FreeBSD összes jelenlegi verziójában. Az Intel® architektúrájú processzorok memóriaszervezésének korlátai miatt nem különböztethetõ meg a 4 gigabyte alatti és feletti memória. A 4 gigabyte felett található memóriaterületek egyszerûen hozzáadódnak a rendelkezésre álló memóriához.

A rendszermagban a PAE támogatását egyszerûen az alábbi sor hozzáadásával tudjuk engedélyezni:

```
options         PAE
```

A FreeBSD-ben a PAE támogatása csak az Intel® IA-32 architektúrájú processzoraihoz érhetõ el. Emellett meg kell említenünk, hogy a FreeBSD-ben

található PAE támogatás nem lett szélesebb körben próbára téve, ezért a FreeBSD többi megbízható elemeihez képest csak béta állapotúnak tekinthetõ.

A FreeBSD PAE támogatásának van néhány hiányossága:

- Egy futó program a virtuális memóriában nem képes 4 gigabyte-nál többet elérni.
- A bus_dma(9) felületet nem használó eszközmeghajtók adathibákat okozhatnak a PAE-t támogató rendszermagokban, és emiatt nem ajánljuk a használatukat. Ebbõl a megfontolásból készítettünk egy PAE nevû konfigurációs állományt a FreeBSD-hez, amelyben nem szerepel egyetlen olyan meghajtó sem, amely ismereteink szerint nem mûködik együtt a PAE-t támogató rendszermagokkal.
- Bizonyos finomhangolási beállítások a memóriahasználatot a rendelkezésre álló fizikai memória mennyiségébõl számítják ki. A PAE támogatással mûködõ rendszerek esetében megjelenõ sok memória miatt azonban az ilyen eszközök szükségtelenül több területet foglalhatnak le. Erre példa lehet a `kern.maxvnodes` sysctl változó, amely a rendszermag által maximálisan felhasználható virtuális csomópontok számát korlátozza. Ajánlott tehát az ilyen és ehhez hasonló beállítások értelmes értékre történõ visszaállítása.
- Szükséges lehet a rendszermag virtuális címterének (KVA) növelése vagy a rendszermag által túlságosan nagy méretûre foglalt címterû különféle erõforrások (lásd fentebb) csökkentése a KVA kifogyásának elkerülésére. A KVA területének növelését a `KVA_PAGES` beállításával tehetjük meg.

Ha gondjaink lennének a teljesítménnyel vagy a megbízhatósággal, keressük fel a tuning(7) man oldalt. A pae(4) man oldalon pedig a FreeBSD PAE támogatásáról találhatunk naprakész információkat.

8.7. Ha valamilyen hiba történne

Négyféle probléma jelentkezhet egy saját rendszermag készítése során. Ezek:

A `config` hibát jelez

Amikor a config(8) parancs hibát jelez vissza a rendszermagunk konfigurációs beállításainak feldolgozása során, akkor minden bizonnyal csak egy apró hibát vétettünk valahol. Szerencsére a config(8) kiírja a hibás sor számát, ezért gyorsan fel tudjuk kutatni a hibát tartalmazó sort. Például, ha ezt látjuk:

```
config: line 17: syntax error
```

Akkor gyõzõdjünk meg róla, hogy helyesen írtuk be az adott sorban szereplõ kulcsszót. Ebben segítségünkre lehet, ha összevetjük a GENERIC konfigurációs állománnyal vagy más hivatkozásokkal.

A `make` hibát jelez

Ha a `make` jelez hibát, az általában arra utal, hogy az általunk korábban megadott rendszermag konfigurációs állományt a config(8) nem értette meg rendesen. Megint azt tudjuk csak javasolni, hogy nézzük át a konfigurációs beállításainkat, és ha ezután sem sikerül megoldani a problémát,

akkor mellékeljük egy levélben a rendszermagunk konfigurációs beállításait és küldjük el a FreeBSD general questions levelezési lista címére, ahol a hozzáértõk gyorsan átnézik.

A rendszermag nem indul:

Ha az új rendszermagunk nem indul vagy nem képes felismerni az eszközeinket, ne essünk kétségbe! Szerencsére a FreeBSD tökéletes megoldással tud szolgálni az összeférhetetlen rendszermagok esetére: a FreeBSD rendszerbetöltõjében egyszerûen válasszuk ki az indítandó rendszermagot. Ezt akkor tudjuk elõhívni, amikor a rendszerindító menü megjelenik. Válasszuk ki a hatos, vagyis az "Escape to a loader prompt" (a betöltõ parancssorának elõhívása) menüpontot. Mikor megjelenik a parancssor, írjuk be, hogy `unload kernel`, majd adjuk ki a `boot /boot/kernel.old/kernel`, parancsot, amiben bármilyen más olyan rendszermagot is megnevezhetünk, ami korábban már mûködött. Ezért amikor beállítunk egy új rendszermagot, mindig érdemes a kezünk ügyében tartani legalább egy olyan rendszermagot, amely mûködik.

Miután sikerült elindítanunk az egyik használható rendszermagot, nézzük át még egyszer a konfigurációs állományt és próbáljuk újra lefordítani a rendszermagot. A probléma megoldását segítheti a /var/log/messages állomány áttanulmányozása is, ami többek közt rögzíti a rendszermag sikeres indulása során keletkezõ üzeneteket. Ezenkívül a dmesg(8) parancs is meg tudja jeleníteni az aktuális rendszerindítás üzeneteit.

Ha gondok merülnének fel a rendszermag elkészítése során, mindenképpen tartsuk meg a GENERIC, vagy bármilyen másik olyan rendszermagot, amelyrõl tudjuk, hogy mûködik. Nevezzük át, így nem fog felülíródni a következõ fordítás és telepítés során. A kernel.old állományra ugyanis nem minden esetben számíthatunk, mivel az új rendszermagok telepítésénél a kernel.old mindig felülíródik a legutóbb telepített rendszermaggal, amely azonban nem feltétlenül lesz mûködõképes. Sõt, amint csak lehetséges, rakjuk a mûködõ rendszermagot a /boot/kernel könyvtárba vagy különben a ps(1) és a hozzá hasonló parancsok nem fognak rendesen mûködni. Mindezek elvégzéséhez egyszerûen nevezzük át a jó rendszermagot tartalmazó könyvtárt:

```
# mv /boot/kernel /boot/kernel.rossz
# mv /boot/kernel.jó /boot/kernel
```

A rendszermag mûködik, a ps(1) viszont nem

Ha olyan rendszermagot telepítettünk, aminek a verziója nem egyezik meg a hozzá tartozó segédprogramokéval, tehát például -CURRENT rendszermagot raktunk egy -RELEASE rendszerhez, egyes rendszerállapotjelzõ parancsok, mint például a ps(1) vagy a vmstat(8) nem fognak mûködni. Ebben az esetben az egész rendszert újra kell fordítanunk és telepítenünk a rendszermagunkkal megegyezõ verziójú forrásból. Részben ezért sem különösen ajánlott, hogy az operációs rendszer többi részétõl eltérõ verziójú rendszermagot használjunk.

Chapter 9. Nyomtatás

9.1. Áttekintés

A FreeBSD a nyomtatók széles skálájával képes együttmûködni, a legrégebbi vegyszeres nyomtatótól kezdve egészen napjaink lézernyomtatójáig, aminek köszönhetõen alkalmazásainkkal nagyon jó minõségû nyomtatásokat tudunk készíteni.

A FreeBSD a helyi hálózaton nyomtatószervernek is beállítható. Ekkor a vele közös hálózatra csatlakozó többi, FreeBSD, Windows® vagy Mac OS® rendszerû számítógéptõl képes nyomtatási kéréseket elfogadni. A FreeBSD gondoskodik róla, hogy egyszerre csak egy nyomtatás készüljön el, számon tartja, hogy mely felhasználók és számítógépek nyomtatnak a legtöbbet, és minden feladathoz "munkalapot" (banner page) készít, amiben többek közt megtalálhatjuk, hogy kihez tartozik.

A fejezet elolvasása során megismerjük:

- hogyan állítsuk be a FreeBSD nyomtatási sorát;
- hogyan telepítsünk nyomtatási szûrõket, hogyan kezeljünk különbözõ speciális nyomtatási feladatokat, tehát például miként alakítsuk át a beérkezõ dokumentumokat olyan nyomtatási formátumra, amelyet a nyomtatónk is megért;
- hogyan engedélyezzük a fejléc- vagy nyomtatási információk kinyomtatását;
- hogyan nyomtassunk más számítógépekhez csatlakoztatott nyomtatókkal;
- hogyan nyomtassunk a hálózatra közvetlenül kapcsolt nyomtatókkal;
- hogyan állítsuk be a nyomtatási korlátozásokat, például a nyomtatási feladatok méretét, amivel egyes felhasználók nyomtatását visszafoghatjuk;
- hogyan készítsünk nyomtatási kimutatásokat és nyilvántartást a nyomtató használatáról;
- hogyan keressük meg a nyomtatás során felmerülõ problémák okait.

A fejezet elolvasásához ajánlott:

- egy új rendszermag beállításának és telepítésének ismerete (A FreeBSD rendszermag testreszabása).

9.2. Bevezetés

A FreeBSD-ben a nyomtatók mûködéséhez be kell állítani az LPD nyomtatási rendszert. Ez a Berkeley sornyomtatási rendszere, amelyet ezentúl röviden csak LPD-nek fogunk hívni. Ez a FreeBSD alapértelmezett szabványos nyomtatásvezérlõ rendszere. Ebben a fejezetben az LPD és annak konfigurációja kerül bemutatásra.

Ha már találkoztunk az LPD-vel vagy hozzá hasonló rendszerekkel, akkor innen nyugodtan ugorhatunk a Kezdeti beállítások címû szakaszra.

Az LPD vezérli a számítógéphez csatlakoztatott nyomtató összes funkcióját. Számos feladata van:

- Felügyeli a lokálisan és hálózaton keresztül csatlakoztatott nyomtatók hozzáféréseit.
- Lehetõvé teszi az átküldött állományok kinyomtatását, amelyeket *nyomtatási feladatoknak* nevezünk.
- Minden nyomtatóhoz fenntart egy nyomtatási *sort*, amivel meg tudja akadályozni, hogy egyszerre több felhasználó is hozzá tudjon férni az egyes nyomtatókhoz.
- A *fejléceket* (vagy más néven *munka-* vagy *elválasztó lapokat*) nyomtat, így a felhasználók könnyen megtalálják a saját nyomtatásaikat a többi közt.
- Felügyeli a soros portokon csatlakozó nyomtatók kommunikációs beállításait.
- A hálózaton keresztül átküldi a nyomtatási feladatokat egy másik számítógép LPD sorába.
- A nyomtatási feladatok formázásához lefuttatja az adott nyomtató nyelvéhez és képességeihez illeszkedõ speciális szûrõket.
- Nyilvántartja a nyomtató kihasználtságát.

A beállításait tartalmazó állomány (/etc/printcap) és a speciális szûrõprogramok segítségével az LPD sokféle nyomtatón képes az összes említett feladatot vagy annak egy részét megvalósítani.

9.2.1. Amiért nyomtatási sort érdemes használni

Amikor csak egyedül vagyunk a rendszerben, felmerülhet bennünk a kérdés, hogy minek is kellene nekünk veszõdni a nyomtatási sor beállításával, hiszen nincs szükségünk sem a hozzáférések vezérlésére, sem fejlécekre, sem pedig nyilvántartásra. Noha akár közvetlenül is el tudjuk érni a nyomtatót, néhány okból azért mégis érdemes nyomtatási sort használni:

- Az LPD a háttérben nyomtat, ezért ilyenkor nem kell megvárni, amíg az adat átmásolódik a nyomtatóra.
- Az LPD tetszõlegesen tudja alakítani a nyomtatási feladatokat: hozzájuk tud tenni különbözõ adatokat (dátum és idõ), vagy a speciális állományokat (például a TeX DVI formátumát) képes megértetni a nyomtatóval, és nem nekünk kell mindezeket a lépéseket elvégeznünk.
- Számos nyomtatási lehetõséggel rendelkezõ szabad és kereskedelmi program arra számít, hogy a rendszerünkben nyomtatási sor található, ezért annak beállításával sokkal könnyebb használni ezeket a szoftvereket.

9.3. Kezdeti beállítások

Úgy tudjuk használni a nyomtatókat az LPD nyomtatási rendszerével, ha egyaránt beállítjuk a nyomtatót és magát az LPD-t is. Itt a beállítás két szintjét tárgyaljuk:

- Az Alacsonyszintû nyomtatóbeállítás címû szakaszból megtudhatjuk, hogyan tudunk csatlakoztatni egy nyomtatót, hogyan adjuk meg az LPD-nek, miként kommunikáljon vele, hogyan nyomtassunk ki egyszerû szöveges állományokat a nyomtatón.
- A Magasszintû nyomtatóbeállítás szakaszban bemutatjuk, hogyan nyomtassunk ki különféle speciális állományokat, hogyan készíttessünk fejléceket, hogyan nyomtassunk hálózaton keresztül, hogyan vezéreljük a nyomtatók hozzáférését és hogyan tartsuk nyilván a nyomtató használatát.

9.3.1. Alacsonyszintû nyomtatóbeállítás

Ebben a szakaszban láthatjuk, miképpen kell beállítani a nyomtatónkat és az LPD hogyan lesz képes azt használatba venni. Az alapoktól kezdünk:

- A Hardveres beállítás címû szakaszban abban kapunk segítséget, hogyan kell a nyomtatót a számítógéphez csatlakoztatni.
- A Szoftveres beállítás címû szakaszban az LPD nyomtatási rendszer beállítását tartalmazó állományt (/etc/printcap) vesszük sorra.

Amennyiben olyan nyomtatót akarunk beállítani, amely nem helyileg, hanem valamilyen hálózati protokollon keresztül csatlakozik, nézzük meg a Nyomtatók hálózati adatcsatlakozással címû szakaszt.

Habár ez a szakasz nevében csupán "Alacsonyszintû nyomtatóbeállításról" szól, meglehetõsen szerteágazó tud lenni. A nyomtató hardveres és szoftveres életre keltése az egyik legnehezebb feladat. Ha van egy mûködõ nyomtatónk, a fejlécek és a nyilvántartás beállítása tulajdonképpen már gyerekjáték.

Hardveres beállítás

Ebben a szakaszban a nyomtatók csatlakoztatásának lehetséges módozatairól esik szó. Beszélni fogunk mindenféle portokról és kábelekrõl, és a FreeBSD rendszermagjának az egyes nyomtatók használatához szükséges beállításairól is.

Ha korábban tudtuk csatlakoztatni a nyomtatónkat, és más operációs rendszerekkel már sikeresen nyomtattunk is vele, akkor rögtön ugorhatunk is a Szoftveres beállításokat tartalmazó szakaszra.

Portok és kábelek

A személyi számítógépekhez kapható nyomtatók általában a következõ három csatolófelület egyikével rendelkeznek:

- A *soros*, más néven RS-232-es vagy COM porton keresztül kommunikáló felületek a számítógép soros portján küldenek adatot a nyomtatónak. A soros csatolófelületek igen elterjedtek a számítógépiparban, könnyen tudunk ilyen kábelt szerezni, gyorsan is gyártható. Elõfordulhat, hogy a soros csatolófelületek használatához valamilyen különleges kábelre, valamint bonyolult kommunikációs beállítások megadására van szükség. A legtöbb soros port által elérhetõ legnagyobb adatátviteli sebesség másodpercenként 115 200 bit, ami miatt azonban a komolyabb grafikai tartalmak nyomtatása szinte lehetetlen.
- A *párhuzamos* csatolófelületek a számítógépünk párhuzamos portjával küldenek adatokat a nyomtatónak. A párhuzamos felületek gyorsabbak az RS-232 soros felületnél, és a számítógéppiacon is gyakran megtalálhatóak. Könnyen tudunk ilyen kábelt szerezni, azonban kézileg nehezebb elkészíteni. A párhuzamos csatolófelületekhez általában nem tartoznak kommunikációs beállítások, ezért rendkívül egyszerûen el lehet boldogulni velük.

  A párhuzamos felületekre olykor "Centronics" csatolófelületként is hivatkoznak, amelyet egy nyomtatótípus után neveztek el.

- A Universal Serial Bus (Univerzális soros busz) rövidítéseként használt USB elnevezésû csatolófelület a párhuzamos és a soros felületeknél jóval nagyobb sebességre képes. A hozzá tartozó kábelek felépítése egyszerû és az áruk olcsó. Habár a nyomtatás terén az USB hivatott leváltani az RS-232-es soros és a párhuzamos felületeket, nem mindegyik UNIX® rendszer támogatja kellõképpen. Ezt a problémát például úgy kerülhetjük el, ha olyan nyomtatót vásárolunk, amelyen a legtöbbhöz hasonlóan a párhuzamos és az USB csatlakozás is megtalálható.

A párhuzamos felületeken általában csak egy irányban tudunk üzeneteket küldeni (a számítógéptõl a nyomtatóhoz), miközben az USB és a soros felület használatával mind a két irányban is. FreeBSD alatt viszont már az újabb (EPP és ECP) párhuzamos portok egy IEEE 1284 szabványú kábellel képesek oda-vissza kommunikálni.

A párhuzamos nyomtatók kétirányú kommunikációját általában két mód közül az egyiken szokták megvalósítani. Az elsõ esetben a FreeBSD a nyomtatóhoz egy speciális meghajtót használ, amely ismeri az általa beszélt nyelvet. Ilyenek a tintasugaras nyomtatók, amelyek más egyéb állapotinformációk mellett ezen keresztül képesek jelezni a tintapatronokban levõ tinta mennyiségét. A második esetben a nyomtató ismeri a PostScript® nyelvet.

A PostScript® nyelvû nyomtatási feladatok valójában a nyomtatónak küldött programok. Használatukhoz még papírra sincs feltétlenül szükség, és elõfordulhat, hogy közvetlenül a számítógépnek válaszolnak. A PostScript® is kétirányú kommunikáción keresztül értesíti a számítógépet az olyan gondokról, mint például a PostScript® programokban levõ hibák vagy a papír beakadása, amely információnak a felhasználók szoktak örülni. Hovatovább ez a kétirányú kommunikáció a kulcsa a PostScript® nyomtatók hatékony nyilvántartásának is: egyszerûen lekérdezzük a nyomtatótól a lapszámlálót (ami megadja, hogy a nyomtató eddig mennyi lapot nyomtatott ki), kiküldjük a felhasználóhoz tartozó feladatot és ismét lekérdezzük a lapszámlálót. A két érték kivonásából tájékozódhatunk a felhasználó által igényelt lapok mennyiségérõl.

Párhuzamos portok

A párhuzamos csatolófelületen érintkezõ nyomtató használatához kapcsoljunk össze számítógépünket és nyomtatónkat egy párhuzamos kábellel. Az erre vonatkozó konkrét utasítások a nyomtató és/vagy a számítógép kézikönyvében olvashatóak.

Jegyezzük meg, hogy a számítógép melyik párhuzamos portjára csatlakoztattuk a kábelt. FreeBSD alatt az elsõ ilyen port a ppc0 eszköz, a második pedig a ppc1 eszköz lesz és így tovább. A nyomtatóeszköz elnevezése ugyanezt a sémát követi: a /dev/lpt0 lesz az elsõ párhuzamos porton levõ nyomtató stb.

Soros portok

A soros csatolófelületet használó nyomtatók beüzemeléséhez elõször egy soros kábel segítségével kapcsoljuk össze a számítógépünkkel. Ennek pontos részleteit a nyomtató és/vagy a számítógépünk kézikönyvében találhatjuk meg.

Ha nem vagyunk benne biztosak, hogy milyen a "megfelelõ soros kábel", próbáljunk az alábbiak alapján dönteni:

- A *modem* kábele a két oldalán levõ, egymásnak megfelelõ tüskéket közvetlenül összeköti. Ezt a

típust nevezik "DTE-DCE" kábelnek.

- A *null-modem* kábel bizonyos érintkezõket rendesen, másokat pedig fordítva köt össze (például a küldõt a fogadóval), illetve némelyeket rövidre zár közvetlenül a csatlakozón belül. Ez a típus a "DTE-DTE" kábel.
- Néhány speciális nyomtató esetén elõfordul még a *soros nyomtató*kábel, amely leginkább a null-modem kábelekhez hasonlít, azonban az ott rövidre zárt csatornák itt a nekik megfelelõ érintkezõknek továbbítanak jeleket.

Emellett még a nyomtató elõlapján vagy az alján található kapcsolók segítségével be kell állítanunk a nyomtatóhoz tartozó kommunikációs paramétereket is. Itt válasszuk azt a `bps` (a bitek száma másodpercenként) értéket, amelyet még a számítógépünk és a nyomtatónk is egyaránt képes támogatni. Válasszunk 7 vagy 8 adatbitet, páros, páratlan vagy kikapcsolt paritásbitet és 1 vagy 2 stopbitet. Ekkor tudjuk megadni a forgalomirányítási protokollt is: lehet kikapcsolt, XON/XOFF (ez az ún. "sávon belüli" vagy "szoftveres") forgalomirányítás. Ne felejtsük el ezeket a beállításokat a most következõ szoftveres beállítások elvégzése során sem.

Szoftveres beállítás

Ebben a fejezetben tárgyaljuk a FreeBSD-ben található LPD nyomtatási rendszer mûködéséhez és a nyomtatáshoz szükséges szoftveres beállításokat.

Íme az elvégzendõ lépések rövid vázlata:

1. Amennyiben szükséges, állítsuk be a rendszermagunkat a nyomtató által használt portra. Ehhez A rendszermag beállítása szakaszban olvashatjuk el, mit is kell pontosan tenni.
2. Ha párhuzamos portot használunk, akkor állítsuk be, hogy a párhuzamos port miként fog kommunikálni. A párhuzamos port kommunikációs módjának beállítása címû szakasz tárja fel ennek részleteit.
3. Próbáljuk ki, hogy ezek után az operációs rendszer képes-e adatot küldeni a nyomtatónak. A nyomtató kommunikációjának ellenõrzése szakaszban kapunk erre pár javaslatot.
4. Az /etc/printcap állomány felhasználásával állítsuk be a nyomtatónkhoz az LPD-t. Errõl a fejezet további részei adnak majd felvilágosítást.

A rendszermag beállítása

Az operációs rendszer magja eszközök egy adott csoportjával képes együttmûködni, amelyben a soros és párhuzamos felületen csatlakozó nyomtatók is megtalálhatóak. Azonban ha a rendszermag nem ismeri fel még valamelyiket, akkor a soros vagy párhuzamos portok használatához külön támogatásra van szükség.

Így tudjuk megnézni, hogy a jelenleg használt rendszermag támogatja-e a soros csatolófelületet:

```
# grep sioN /var/run/dmesg.boot
```

Itt az *N* nullától kezdõdõen adja meg a soros port sorszámát. Amennyiben látunk valami ilyesmit:

```
sio2 at port 0x3e8-0x3ef irq 5 on isa
sio2: type 16550A
```

Ez azt jelenti, hogy a rendszermag sikeresen észlelte a portot.

A párhuzamos csatolófelület támogatásáról így gyõzõdhetünk meg:

```
# grep ppcN /var/run/dmesg.boot
```

Itt az *N* nullától kezdõdõen sorszámozza a párhuzamos portot. Ha eredményül valami hasonlót kapunk:

```
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
ppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode
ppc0: FIFO with 16/16/8 bytes threshold
```

Ez arra utal, hogy a rendszermagunk tud a portról.

Elõfordulhat azonban, hogy az operációs rendszer csak akkor fogja észrevenni a nyomtatásra használt soros vagy párhuzamos portot, ha átállítjuk a rendszermagunkat.

A soros port támogatásának beállításához olvassuk el a rendszermag beállításáról szóló szakaszt. A párhuzamos port támogatásához szintén olvassuk el ugyanazt a szakaszt *és* a most következõt.

A párhuzamos port kommunikációs módjának beállítása

A párhuzamos csatolófelület használata esetén választhatunk, hogy a FreeBSD milyen módon tartsa a kapcsolatot a nyomtatóval: megszakításokkal vezérelje (interrupt-driven), vagy esetleg folyamatosan kérdezgesse (polled). A FreeBSD általános meghajtója (lpt(4)) a ppbus(4) alrendszert használja, ami a portot a ppc(4) meghajtón keresztül vezérli.

- A *megszakítás alapú* módszer a GENERIC rendszermagban alapértelmezés. Ilyenkor az operációs rendszer egy megszakításkérés felhasználásával értesül arról, hogy a nyomtató mikor áll készen adatok fogadására.
- A *lekérdezéses* módszer használata során az operációs rendszer folyamatosan érdeklõdik a nyomtató rendelkezésre állásáról. Amikor erre pozitív megerõsítést kap, akkor a rendszermag újabb adatokat küld.

A megszakításos módszer valamivel gyorsabb, azonban cserébe lefoglal egy értékes IRQ vonalat. A HP újabb nyomtatói állítólag nem mûködnek megfelelõen ilyen módban, valamilyen (pillanatnyilag még nem teljesen tisztázott) idõzítési probléma miatt. Ezért az ilyen nyomtatóknak is valószínûleg a lekérdezéses módszert kell használniuk. Más nyomtatók pedig, habár mûködnek mind a két módszerrel, hihetetlenül lassúak a megszakításokkal.

Kétféleképpen állíthatjuk be a kommunikációs módot: a rendszermagon keresztül, vagy az lptcontrol(8) segédprogrammal.

A rendszermagban így állíthatjuk be a kommunikációt:

1. Írjuk át a rendszermag beállításait tartalmazó állományt. Keressük meg benne a használt párhuzamos portnak megfelelõen a `ppc0`, `ppc1` (második párhuzamos port) vagy `ppc2` (harmadik párhuzamos port) bejegyzést, és engedélyezzük.
   - A megszakításos mód használatához nyissuk meg a /boot/device.hints állományt, és az *N* helyére írjuk be a

     ```
     hint.ppc.0.irq="N"
     ```

     sorba a megfelelõ IRQ számát. A rendszermag beállításait tartalmazó állománynak tartalmaznia kell a ppc(4) meghajtót is:

     ```
     device ppc
     ```

   - A lekérdezéses mód használatához a /boot/device.hints állományból távolítsuk el a következõ sort:

     ```
     hint.ppc.0.irq="N"
     ```

     Némely esetben azonban ennyi még nem lesz elég a port lekérdezéses beállításához. Ugyanis ha a hozzá tartozó meghajtó az acpi(4), akkor ez fogja felismerni, kezelni és a nyomtatóhoz tartozó portok hozzáférési módját vezérelni. A problémát ezért gyakran érdemes az acpi(4) beállításai között is keresni.
2. Mentsük el az állományt. Konfiguráljuk be, fordítsuk le és telepítsük az új rendszermagot. Ennek pontos részleteit a rendszermag beállításáról szóló fejezetben olvashatjuk.

A kommunikáció módjának beállítása az lptcontrol(8) programmal:

1. A megszakításos mód beállításához írjuk be:

   ```
   # lptcontrol -i -d /dev/lptN
   ```

   ahol az `lptN` a nyomtatóhoz tartozó eszköz neve.
2. A lekérdezéses mód beállításához írjuk be:

   ```
   # lptcontrol -p -d /dev/lptN
   ```

   ahol az `lptN` a nyomtatóhoz tartozó eszköz neve.

Ha ezeket a parancsokat berakjuk az /etc/rc.local állományunkba, akkor azzal a rendszer minden egyes indítása során beállítjuk a számunkra megfelelõ módot. Errõl többet az lptcontrol(8) man oldaláról tudhatunk meg.

A kommunikáció ellenõrzése

Még mielõtt nekilátnánk a nyomtatási rendszer beállításának, bizonyosodjuk meg róla, hogy az operációs rendszer képes adatokat továbbítani a nyomtatónak. Sokkal könnyebb egymástól függetlenül megvizsgálni a kommunikáció és a nyomtatási rendszer mûködését.

A nyomtatót úgy tudjuk kipróbálni, ha küldünk neki valamilyen szöveget. Az lptest(1) tökéletesen megfelelõ akkor, ha olyan nyomtatónk van, amely azonnal kinyomtatja a kapott szöveget. Ez a program 96 sorban létrehozza mind a 96 kinyomtatható ASCII karaktert.

A PostScript® (vagy más egyéb nyelvet ismerõ) nyomtatóknak azonban ennél kifinomultabb próbára van szüksége. Erre a célra tökéletesen megfelel egy olyan kisebb PostScript® programocska, mint például ez:

```
%!PS
100 100 moveto 300 300 lineto stroke
310 310 moveto /Helvetica findfont 12 scalefont setfont
(Remek! Ez mukodik!) show
showpage
```

Ezt a PostScript® kódot nyugodtan elmenthetjük egy állományba, amelyet aztán a késõbbi szakaszokban megjelenõ példák szerint használni is tudunk majd.

A kézikönyvben a nyomtató nyelve alatt leginkább egy PostScript®-szerû nyelvet értünk, nem pedig a Hewlett Packard PCL típusú nyelvét. Habár a PCL nagyon sokra képes, hiszen keverhetjük még benne akár a programokat és a nyers szövegeket is. Ezzel szemben a PostScript® nem képes nyers szöveget kinyomtatni, ezért az ilyen típusú nyomtatók mûködtetéséhez külön támogatásra van szükségünk.

A párhuzamos nyomtató ellenõrzése

Ebben a szakaszban megtudhatjuk, hogy FreeBSD alatt miként ellenõrizzük a párhuzamos portra csatlakozó nyomtatók mûködését.

A párhuzamos porton levõ nyomtató kipróbálásához:

1. A su(1) segítségével váljunk `root` felhasználóvá.
2. Küldjünk a nyomtatónak valamilyen adatot.
   - Ha a nyomtató képes nyers szöveget fogadni, akkor használjuk az lptest(1) programot. Ehhez gépeljük be:

```
# lptest > /dev/lptN
```

ahol az *N* nullától kezdõdõen a párhuzamos port sorszáma.

- Ha a nyomtató PostScript® vagy más nyomtatási nyelvet ismer, akkor egy apró programot kell küldenünk neki. Ehhez írjuk be:

```
# cat > /dev/lptN
```

Ezután soronként írjuk be a programot, de *vigyázzunk*, mert az `Enter` vagy a `Return` lenyomása után már nem tudjuk kijavítani! A program begépelése után nyomjuk meg a `Ctrl` + `D` vagy bármely más olyan billentyûkombinációt, amivel ki tudunk lépni.

Ezt a programot belerakhatjuk egy állományba is, amire aztán adjuk ki az alábbi parancsot:

```
# cat állomány > /dev/lptN
```

ahol az *állomány* a nyomtatóra küldendõ program neve lesz.

Ezután a nyomtató megkezdi a nyomtatást. Ne aggódjunk, ha netalán valami furcsán nézne ki, mert a késõbbiekben ezt még úgyis rendbetesszük.

A soros nyomtató ellenõrzése

Ebben a szakaszban megtudhatjuk, hogyan ellenõrizzük a FreeBSD és soros portra kötött nyomtató kapcsolódását.

Így tudjuk kipróbálni a soros porton csatlakozó nyomtatónkat:

1. A su(1) paranccsal váljunk `root` felhasználóvá.
2. Nyissuk meg az /etc/remote állományt. Tegyük hozzá a következõ sort:

```
printer:dv=/dev/port:br#bps:pa=paritás
```

ahol a *port* a soros porthoz tartozó eszközleíró neve (`ttyd0`, `ttyd1`, stb.), a *bps* a nyomtató által használt adatátviteli sebesség, végül a *paritás* a nyomtatóhoz használt paritás (ami lehet `even` (páros), `odd` (páratlan), `none` (nincs), vagy `zero` (nulla)).

Íme egy olyan soros nyomtató beállítása (`printer` néven), amely sebessége 19 200 bps, a harmadik portra csatlakozik és nem használ paritást:

```
printer:dv=/dev/ttyd2:br#19200:pa=none
```

3. Kapcsolódjunk a nyomtatóhoz a tip(1) segítségével. Ennek parancsa:

   ```
   # tip printer
   ```

   Ha az iménti lépés nem mûködne, próbálkozzunk az /etc/remote állomány újbóli módosításával, és a /dev/cuaaN eszköz helyett használjuk a /dev/ttydN eszközt!

4. Küldjünk adatot a nyomtatónak.
   - Ha a nyomtató képes nyers szöveget nyomtatni, akkor használjuk az lptest(1) segédprogramot. Gépeljük be:

     ```
     % $lptest
     ```

   - Ha a nyomtató a PostScript® vagy egy hozzá hasonló nyomtatási nyelven kommunikál, akkor a nyomtatónak egy rövid programot kell küldenünk. Soronként gépeljük be a programot, azonban *vigyázzunk* arra, hogy a törlés és minden más szerkesztésre használt billentyû a nyomtató számára is értelmes lehet. Az is elõfordulhat, hogy a program küldését egy speciális jelsorozattal tudjuk csak lezárni. A PostScript® nyomtatók esetén ilyenkor elegendõ a `Ctrl` + `D` billentyûk együttes lenyomása.

     Vagy tehetjük az egész programot egy állományba, amihez aztán írjuk be ezt:

     ```
     % >állomány
     ```

     ahol az *állomány* a programot tartalmazó állomány neve. Miután a tip(1) elküldte az állományt, nyomjuk le a lezáráshoz szükséges billentyûkombinációt.

Most már meg kellene jelennie valaminek a nyomtatón. Az még nem számít, pontosan mi is lesz az - késõbb még majd úgyis beállítjuk.

A nyomtatási rendszer aktiválása: a /etc/printcap állomány

Csatlakoztattuk a nyomtatónkat, a mûködtetéséhez beállítottuk a rendszermagot (amennyiben erre szükségünk volt), és tudtunk neki adatokat küldeni. Most már készen állunk arra, hogy LDP alkalmazáson keresztül beállítsuk a nyomtató hozzáférésének vezérlését.

Az LPD beállításait az /etc/printcap állományban találjuk. Az LPD nyomtatási rendszer minden egyes mûvelet elõtt beolvassa ezt az állományt, ezért a benne végzett módosítások szinte azonnal életbe is lépnek.

A printcap(5) tartalma könnyen érthetõ, a /etc/printcap állományt egyszerûen módosíthatjuk a kedvenc szövegszerkesztõnkkel. A felépítése teljesen megegyezik a többi hozzá hasonló állományéval: ilyenek például a /usr/shared/misc/termcap és a /etc/remote. Az itt alkalmazott formátum teljes leírását a cgetent(3) man oldalon találjuk.

A nyomtatási rendszer egyszerû beállítása az alábbi lépésekbõl áll:

1. Adjunk nevet (és még néhány álnevet) a nyomtatónak, írjuk ezeket az /etc/printcap állományba. A nevekről A nyomtató elnevezése címû szakaszban kapunk felvilágosítást.
2. A(z alapból bekapcsolt) fejléclapokat az `sh` tulajdonság megadásával kapcsolhatjuk ki. A részleteket A fejléclapok letiltása címû szakaszban találjuk.
3. Hozzunk létre egy nyomtatási könyvtárat, és adjuk meg a helyét az `sd` tulajdonság beállításával. A nyomtatási könyvtár létrehozása címû szakaszban fogunk erről többet mondani.
4. Állítsunk be egy nyomtató által használt /dev könyvtárbeli leírót, és az `lp` tulajdonsággal adjuk meg az /etc/printcap állományban. Erről részletesebben A nyomtatóeszköz azonosítása címû szakaszban olvashatunk. Ha a nyomtató soros porton keresztül csatlakozik, az `ms#` tulajdonsággal még meg kell adnunk A nyomtatási rendszer kommunikációs paraméterei szakaszban tárgyaltakat is.
5. Helyezzünk el egy szûrõt a beérkezõ nyers szövegek számára. Erről A szövegszûrõ telepítése címû szakasz értekezik.
6. Az lpr(1) parancs segítségével próbáljuk ki a nyomtatást. Ennek pontos részleteit a Próbáljuk ki! és a Hibakeresés címû fejezetekben találhatjuk meg.

A magasabb szintû nyomtatók, mint például a PostScript® nyomtatók nem képesek közvetlenül nyers szöveget nyomtatni. Az imént felvázolt egyszerû beállítási séma feltételezi, hogy csak olyan állományokat fogunk nyomtatni a nyomtatón, amelyeket meg is ért.

A felhasználók gyakran arra számítanak, hogy bármelyik általuk elérhetõ nyomtatón képesek nyers szöveget kinyomtatni. Az LPD alkalmazással kapcsolatban álló programok is általában ugyanezt az elgondolást követik. Ha egy saját nyelvvel rendelkezõ nyomtatót akarunk telepíteni, de a nyomtató saját nyelvén *és* a nyers szöveg formájában érkezõ nyomtatási feladatok is rendesen ki akarjuk nyomtatni, akkor mindenképpen javasoljuk, hogy illeszünk még egy további lépést is ebbe a sorba: illesszünk a rendszerbe egy nyers szövegről automatikusan PostScript® (vagy más egyéb) nyelvre tolmácsoló programot. Erről a Szöveges nyomtatási feladatok PostScript® nyomtatókon címû fejezetben olvashatunk.

A nyomtató elnevezése

Az elsõ (egyszerû) lépés a nyomtatónk nevének kiválasztása. Igazából nem számít, mennyire kifejezõ vagy éppen hóbortos nevet adunk neki, hiszen emellett még számos álnévvel is illethetjük.

Az /etc/printcap állományban megtalálható nyomtatók egyikének legalább az `lp` álnévvel rendelkeznie kell, mivel ez lesz az alapértelmezett nyomtató neve. Tehát ha a felhasználó nem adja meg sem a `PRINTER` környezeti változót, sem pedig az LPD-vel kapcsolatban álló aktuális parancsban a használni kívánt nyomtató nevét, akkor a rendszer az `lp` nevût fogja keresni.

Ezenkívül általában még gyakran adnak egy olyan álnevet is a nyomtatónak, ahol annak teljes leírása, többek közt a gyártmánya és a típusa szerepel.

Ahogy sikerült nevet és álneveket adni a nyomtatónak, írjuk is be ezeket az /etc/printcap

állományba. Itt a nyomtató neveit balról kezdjük felsorolni, mindegyik álnevet egy függõleges vonallal válasszunk el, és az utolsó után tegyünk pontosvesszõt.

A most következõ példában egy olyan vázt mutatunk be az /etc/printcap állományhoz, amelyben két nyomtatót (egy Diablo 630 márkájú sornyomtatót és egy Panasonic KX-P4455 típusú PostScript® lézernyomtatót) adunk meg:

```
#
#  /etc/printcap (rose)
#
rattan|line|diablo|lp|Diablo 630 Line Printer:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:
```

Ebben a példában az elsõ nyomtató neve `rattan`, és ehhez tartozik még a `line`, `diablo`, `lp`, és `Diablo 630 Line Printer` álnév. Mivel itt soroltuk fel az `lp` álnevet is, ezért a rendszerben ez lesz az alapértelmezett nyomtató. A második nyomtató neve `bamboo`, és álnevei többek közt a `ps`, `PS`, `S`, `panasonic`, valamint a `Panasonic KX-P4455 PostScript v51.4`.

A fejléclapok letiltása

Az LPD nyomtatási rendszer alapértelmezés szerint minden egyes feladathoz *fejléclapot* készít. Ez a lap szép nagy betûkkel tartalmazza a nyomtatási feladatot kiadó felhasználó nevét, a gépet, amirõl küldték, és a feladat nevét. Sajnálatos módon ez azonban inkább akadályozza a hibakeresést a nyomtató beállításában, ezért most inkább kapcsoljuk ki ezeket.

Ha le akarjuk tiltani a fejléclapokat, az /etc/printcap állományban adjuk meg az `sh` (úgymint "suppress header pages") tulajdonságot. Íme egy példa az `sh` tulajdonsággal bõvített /etc/printcap állományra:

```
#
#  /etc/printcap (rose) - sehol sem lesznek fejléclapok
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:
```

Ebben a példában megfigyelhetjük a helyes felírási módot: az elsõ sor a bal szélsõ oszlopban kezdõdik, az azt követõ sorok pedig bentebb. Minden bejegyzésben az utolsó kivételével mindegyik sor egy visszaper (backslash) karakterrel zárul.

A nyomtatási könyvtár létrehozása

A nyomtatási rendszerünk beállításának következõ lépése a *nyomtatási könyvtár* létrehozása. Ez egy olyan könyvtár, ahová a különbözõ nyomtatási feladatok kerülnek a feldolgozásuk elõtt, valamint ahol a nyomtatási rendszer többi állománya lakozik.

A nyomtatási rendszer adatait tároló könyvtárakat tartalmuk gyakori változása miatt általában a /var/spool könyvtárba szokás tenni. Ezen könyvtárak tartalmát nem szükséges menteni sem. Az mkdir(1) parancs futtatásával egyszerûen újra létre tudjuk hozni.

Általában minden nyomtatóhoz külön létre szoktak hozni egy könyvtárat az adott nyomtató nevén. Erre példa:

```
# mkdir /var/spool/nyomtatónév
```

Azonban ha a hálózatunkon rengeteg nyomtató található, akkor érdemes inkább egyetlen könyvtárat használni, amelyet az LPD számára tartunk fenn.

```
# mkdir /var/spool/lpd
# mkdir /var/spool/lpd/rattan
# mkdir /var/spool/lpd/bamboo
```

Amennyiben fontos nekünk a felhasználói nyomtatások titkosságának megóvása, érdemes levédenünk a nyomtatási könyvtárat, így az nem lesz mindenki által elérhetõ. A nyomtatási könyvtárak tulajdonosa egyedül és kizárólag a daemon felhasználó és a daemon csoport legyen, és hozzá olvasási, írási és keresési engedélyekkel rendelkezzen. Ezt fogjuk most beállítani a példáinkban szereplõ nyomtatóinkhoz is:

```
# chown daemon:daemon /var/spool/lpd/rattan
# chown daemon:daemon /var/spool/lpd/bamboo
# chmod 770 /var/spool/lpd/rattan
# chmod 770 /var/spool/lpd/bamboo
```

Végezetül az /etc/printcap állományban ezeket a könyvtárakat se felejtsük el megadni az LPD-nek. Itt a nyomtatási könyvtár nevét az sd tulajdonsággal írjuk le:

```
#
#  /etc/printcap (rose) - a nyomtatási könyvtárak hozzáadása
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:
```

Vegyük észre, hogy a nyomtató neve ugyan a sor elején kezdõdik, azonban a hozzá tartozó összes többi sor mind bentebb kezdõdik és egy visszaper (backslash) karakterrel választjuk le.

Ha az sd tulajdonsággal nem adunk meg semmilyen nyomtatási könyvtárat, akkor ennek az értéke alapértelmezés szerint a /var/spool/lpd lesz.

A nyomtatóeszköz azonosítása

A Hardveres beállítás címû szakaszban már beazonosítottuk, hogy a FreeBSD a /dev könyvtárban melyik eszközleírón keresztül fogja megszólítani a nyomtatót. Most ideje ugyanezt tudatni az LPD démonnal is. Így amikor a nyomtatási rendszer végre szeretne hajtani egy nyomtatási feladatot, a szûrõprogram nevében ezt az eszközt nyitja meg (ahol a szûrõn keresztül továbbítjuk az adatokat a nyomtató felé).

Az `lp` tulajdonság segítségével a /etc/printcap állományban soroljuk fel a nyomtatók /dev könyvtárban található leíróit.

Az eddig használt példánkban most tételezzük fel, hogy a `rattan` nevû nyomtató az elsõ párhuzamos porton található, míg a `bamboo` nevû a hatodik soros porton. Ebben a helyzetben így kellene kiegészítenünk az /etc/printcap állományunkat:

```
#
#  /etc/printcap (rose) - a használni kívánt eszközök
#  beazonosítása
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:\
        :lp=/dev/lpt0:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:\
        :lp=/dev/ttyd5:
```

Az LPD alapértelmezés szerint a /dev/lp eszközt fogja használni, ha nem adjuk meg az `lp` tulajdonságot az /etc/printcap állományban. A /dev/lp azonban a FreeBSD-ben jelenleg nem létezik.

Ha a telepítendõ nyomtatónk valamelyik párhuzamos portra csatlakozik, akkor innen akár tovább is léphetünk A szövegszûrõ telepítése címû szakaszra. Ha viszont nem, kövessük a most következõ szakaszban szereplõ utasításokat.

A nyomtatási rendszer kommunikációs paraméterei

A soros portra csatlakozó nyomtatóknál az LPD képes beállítani az adatátviteli sebességet, a paritást, valamint más egyéb olyan kommunikációs paramétereket, amelyekkel a szûrõprogram adatokat tud továbbítani a nyomtató felé. Ez több szempontból is elõnyös, mivel:

- Egyszerûen az /etc/printcap állomány átírásával ki tudunk próbálni több kommunikációs beállítást, nem kell magát a szûrõprogramot újrafordítanunk.
- A nyomtatási rendszer képes ugyanazt a szûrõt több, különbözõ kommunikációs beállítást alkalmazó nyomtatóhoz is használni.

Az /etc/printcap állományban az `lp` tulajdonsággal megadott eszközök soros kommunikációjának beállításait az alábbi tulajdonságok határozzák meg:

`br#sebesség`

Beállítja az eszköz adatátviteli sebességét a *sebesség* értékre, ahol a *sebesség* lehet 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19 200, 38 400, 57 600 vagy 115 200 bit másodpercenként (bps).

`ms#stty-mód`

Beállítja az eszköz megnyitása után használt termináleszköz mûködésének paramétereit. Az stty(1) man oldalon többet is megtudhatunk róluk.

Miután az LPD megnyitja az `lp` tulajdonsággal megadott eszközt, beállítja az `ms#` tulajdonság értéke szerint annak jellemzõit. Itt a `parenb`, `parodd`, `cs5`, `cs6`, `cs7`, `cs8`, `cstopb`, `crtscts`, és `ixon` módok lehetnek lényegesek, melyekrõl az stty(1) man oldalon többet is megtudhatunk.

Állítsuk most be az egyik képzeletbeli nyomtatónkat a hatodik soros portra. Az adatátviteli sebessége 38 400 bps lesz. A kommunikáció módjánál kapcsoljuk ki a paritást (`-parenb`), 8 bites karakterek legyenek (`cs8`), ne legyen modemes vezérlés (`clocal`) és a hardveres forgalomirányítás legyen `crtscts`:

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:\
        :lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:
```

A szövegszûrõ telepítése

Most már utasíthatjuk az LPD-t, hogy milyen szövegszûrõt használjon a nyomtatási feladatok eszközre küldéséhez. A *szövegszûrõ* (text filter), vagy más néven *bemeneti szûrõ* (input filter) egy olyan program, amelyet az LPD egy nyomtatási feladat elvégzésekor lefuttat. Amikor az LPD lefuttatja a nyomtatóhoz tartozó szövegszûrõt, a szûrõ szabványos bemenetére elküldi az elvégzendõ nyomtatási feladatot, és a szabványos kimenetét pedig átirányítja az `lp` tulajdonság által megadott nyomtatóeszközre. Ennek megfelelõen a szûrõnek a szabványos bemenetrõl kell olvasnia az elvégzendõ feladatot, a szabványos kimenetre pedig a ténylegesen nyomtatandót kell kiírnia. A szövegszûrõk részleteirõl a Hogyan mûködnek a szûrõk? szakasz szól.

A mi esetünkben most szövegszûrõnek tökéletesen megfelel egy olyan rövid szkript, ami a nyomtatóra a nyomtatási feladatot a `/bin/cat` paranccsal küldi ki. A FreeBSD-ben még találhatunk egy másik szûrõt is, amelynek a neve lpf. Ez képes a törlést és aláhúzást jelzõ karaktereket érthetõvé tenni bizonyos nyomtatók számára. Természetesen itt használhatunk kedvünk szerinti szûrõt is. Az `lpf` szûrõ mûködésének részleteit Az lpf szövegszûrõ címû szakaszban fejtjük ki bõvebben.

Elõször is készítsünk egy /usr/local/libexec/if-simple nevû egyszerû szövegszûrõ szkriptet. A kedvenc szövegszerkesztõnkkel írjuk bele a következõ sorokat:

```
#!/bin/sh
#
# if-simple - egyszerû szövegszûrõ szkript az lpd-hez
# Helye: /usr/local/libexec/if-simple
#
```

```
# Egyszerûen átmásolja a kimenetére a bemenetérõl érkezõ adatokat; nem
# fogad el semmilyen paramétert.

/bin/cat && exit 0
exit 2
```

Tegyük indíthatóvá:

```
# chmod 555 /usr/local/libexec/if-simple
```

Ezután tájékoztassuk róla az LPD-t az /etc/printcap állományban található `if` tulajdonság megadásával. Itt most a példánkban szereplõ mind a két nyomtatóhoz beillesztjük:

```
#
#  /etc/printcap (rose) - a szövegszûrõ hozzáadása
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:\
        :lp=/dev/lpt0:\
        :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:\
        :lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:\
        :if=/usr/local/libexec/if-simple:
```

Az if-simple szkript megtalálható a /usr/shared/examples/printing könyvtárban.

Az LPD elindítása

Az lpd(8) az /etc/rc szkriptbõl, az `lpd_enable` változó értékének megfelelõen indul el. Ennek értéke alapból `NO`, vagyis nem. Ha eddig még nem tettük volna meg, akkor az /etc/rc.conf állományba most vegyük fel a következõ sort:

```
lpd_enable="YES"
```

Ezután vagy indítsuk újra a számítógépünket, vagy pedig adjuk ki az lpd(8) parancsot:

```
# lpd
```

Próbáljuk ki!

Elérkeztünk az LPD egyszerû beállításának utolsó lépéséhez. Sajnos azonban még nem gratulálhatunk, hiszen hátra van még a nyomtató kipróbálása és az esetlegesen elõforduló hibák kijavítása. A beállítást úgy tudjuk a legegyszerûbben letesztelni, ha megpróbálunk valamit

kinyomtatni. Az LPD rendszerben az lpr(1) parancs használatával tudunk nyomtatási feladatokat kiadni.

A kommunikáció ellenõrzése címû szakaszban megtalálhatjuk, hogy hozzunk létre tesztelésre alkalmas szövegeket az lpr(1) és az lptest(1) programok segítségével.

Az LPD beállításainak egyszerû tesztelése:

Írjuk be:

```
# lptest 20 5 | lpr -Pnyomtatónév
```

ahol a *nyomtatónév* az /etc/printcap állományban megadott egyik nyomtató neve (vagy álneve) lehet. Az alapértelmezett nyomtató kipróbálásához ne adjunk meg az lpr(1) parancsnak semmilyen `-P` paramétert. Még egyszer megemlítenénk, hogy amennyiben PostScript® nyomtatót tesztelünk, az elõbbi helyett az lptest(1) paranccsal küldjünk ki egy PostScript® programot. Ehhez tegyük a tesztelõ programunkat egy állományba, majd írjuk be az `lpr állománynév` parancsot.

A PostScript® nyomtató esetén a kiküldött program eredményét kell látnunk. Amennyiben az lptest(1) parancsot használjuk, valami ilyesmire kell számítanunk:

```
!"#$%&'()*+,-./01234
"#$%&'()*+,-./012345
#$%&'()*+,-./0123456
$%&'()*+,-./01234567
%&'()*+,-./012345678
```

A nyomtató kimerítõbb teszteléséhez próbáljunk meg nagyobb programokat keríteni valahonnan (ha a nyomtatónk valamilyen nyelven kommunikál) vagy adjunk meg az lptest(1) parancsnak más paramétereket. Például az `lptest 80 60` soronként 80 karaktert írat ki 60 sorban.

Amennyiben a nyomtató nem mûködne, nézzük meg a Hibakeresés hez tartozó szakaszt.

9.4. Magasszintû nyomtatóbeállítás

Ebben a szakaszban olyan szûrõket mutatunk be, amelyek speciálisan formázott állományok, fejléclapok, hálózati nyomtatás, nyomtatási nyilvántartás vagy szabályozás esetén használhatóak.

9.4.1. Szûrõk

Noha az LPD képes hálózati protokollokat, nyomtatási sorokat, hozzáférést és sok minden más nyomtatási feladatot kezelni, a *tényleges* munka legnagyobb része a *szûrõkben* (filter) történik. A szûrõk olyan programok, amelyek tartják a kapcsolatot a nyomtatóval és megbirkóznak annak eszközfüggõségeivel és különleges igényeivel. Az egyszerû beállítás során egy primitív szövegszûrõt állítottunk be (lásd A szövegszûrõ telepítése) - ami annyira egyszerû, hogy szinte minden nyomtatón mûködnie kell.

Azonban mindahhoz, hogy ki tudjuk használni a különbözõ átalakítási, nyilvántartási lehetõségeket, valamint a nyomtatók különlegességeit és egyebeit, meg kell értenünk a szûrõk pontos mûködését. Az elõbb említett feladatok ugyanis teljesen a szûrõ kezében vannak. Ezzel kapcsolatban azonban rossz hír, hogy ezeket a szûrõket *nekünk* kell megírnunk. A jó hír ellenben az, hogy könnyen találunk ilyen szûrõket, vagy ha éppen nem lelnénk valamelyiket, akkor is gyorsan meg tudjuk ezeket írni.

Sõt, a FreeBSD alapból tartalmaz is egyet, amit a /usr/libexec/lpr/lpf helyen találunk meg, és sok olyan nyomtatóval képes együttmûködni, amelyek nyers szöveget tudnak nyomtatni. (Kezeli az állományokban felbukkanó törléseket és tabulálásokat, valamint képes nyilvántartást vezetni, de semmi többet.) Rajta kívül még számos szûrõt és szûrõelemet is találhatunk a FreeBSD Portgyûjteményében.

Lássuk, mit tartogat számunkra ez a rész:

- A Hogyan mûködnek a szûrõk? címû szakaszban megpróbálunk egyfajta áttekintést adni a szûrõk nyomtatási folyamatban betöltött szerepérõl. Mindenképpen érdemes elolvasnunk ezt a szakaszt, mivel ebben derül ki, hogy valójában mi is történik a "függöny mögött", vagyis amikor az LPD használja ezeket a szûrõket. Ezzel a tudással el tudjuk kerülni vagy éppen nyakon tudjuk csípni azokat a problémákat, amelyek a nyomtatóinkhoz telepített szûrõk hozzáadása során adódhatnak.
- Az LPD alapból arra számít, hogy minden nyomtató képes nyers szöveget nyomtatni. Ez gondot okoz a PostScript® (és minden más nyelv alapú) nyomtatók esetén, mivel azok nem képesek nyers szöveget nyomtatni. Szöveges nyomtatási feladatok PostScript® nyomtatókon címû szakaszban viszont fény derül rá, hogyan kerekedjünk felül ezen. Feltétlenül olvassuk el, ha PostScript® nyomtatónk van.
- A PostScript® számos program közkedvelt kimeneti formátuma, sõt gyakran maguk a felhasználók is szeretnek ilyen programokat írni. Sajnos azonban a PostScript® nyomtatók egyáltalán nem olcsók. A PostScript® szimulációja nem PostScript® nyomtatókon címû szakaszban megtudhatjuk, miképp tudjuk úgy módosítani a szûrõt, hogy *nem PostScript*® nyomtatókon is tudjunk PostScript® programokkal nyomtatni. Ezt a szakaszt akkor érdemes elolvasni, ha nincs PostScript® nyomtatónk.
- A Konverziós szûrõk címû szakaszban elárulljuk, miként lehetséges automatizálni a különbözõ állományformátumok és a nyomtatók által érthetõ formátumok közti konverziókat, legyen az grafikus vagy betûszedésre vonatkozó adat. A szakasz elolvasása során megismerjük, hogyan tudjuk a nyomtatónkat képessé tenni az `lpr -t` paranccsal troff adatok, vagy a `lpr -d` paranccsal a TeX DVI állományainak, esetleg az `lpr -v` paranccsal raszteres képek nyomtatására és így tovább. Csak ajánlani tudjuk ennek elolvasását.
- A Kimeneti szûrõk címû szakaszban kivesézzük az LPD egyik kevésbé használt lehetõségét is, a kimeneti szûrõket. Hacsak nem fejléclapokat akarunk készíteni (lásd Fejléclapok), akkor ezt a szakaszt nyugodtan kihagyhatjuk.
- Az lpf szövegszûrõ szakaszban bemutatásra kerül a FreeBSD-ben alapból megtalálható `lpf` szûrõ, amely egy sornyomtatóknál (vagy az így viselkedõ lézernyomtatóknál) használható egyszerû szövegszûrõ. Ha nyers szövegek nyomtatásánál meg akarjuk oldani a nyomtatási feladatok nyilvántartását, vagy a törlés karakter láttán a nyomtatónk füstölni kezdene, akkor mindenképpen érdemes belemerülnünk az `lpf` titkaiba.

A most következõ szkriptek mindegyike megtalálható a /usr/shared/examples/printing könyvtárban.

Hogyan mûködnek a szûrõk?

Ahogy már korábban is jeleztük, a szûrõ egy olyan végrehajtható program, amelyet az LPD indít el, amikor a nyomtatóval eszközfüggetlen módon kommunikál.

Amikor az LPD egy feladat elvégzése során ki akar nyomtatni egy állományt, akkor elindít egy ilyen szûrõprogramot. A szûrõ szabványos bemenetére elküldi a kinyomtatandó állományt, a szabványos kimenetét a nyomtatóra, a szabványos hibajelzéseit pedig egy naplóállományba irányítja (ez utóbbit az /etc/printcap állományban az `lf` tulajdonsággal adhatjuk meg, vagy alapértelmezés szerinti a /dev/console állományba kerül).

Az LPD a használni kívánt szûrõt és annak paramétereit az /etc/printcap állományban felsoroltak vagy az lpr(1) parancssorában megadottak szerint választja ki. Például, ha a felhasználó a `lpr -t` parancsot adja ki, akkor az LPD a célként megadott nyomtatónál szereplõ `tf` tulajdonság által megadott troff szûrõt kezdi el használni. Amennyiben a felhasználó egyszerûen csak nyers szöveget akar nyomtatni, akkor az `if` szûrõnek kellene elindulnia (ez viszont csak részben igaz: lásd Kimeneti szûrõk).

Háromfajta szûrõ jelenhet meg az /etc/printcap állományban:

- A *szövegszûrõ* (text filter), ami a hagyományos szöveges nyomtatásért felelõs, és amit az LPD dokumentációjában érdekes módon *bemeneti szûrõnek* (input filter) hívnak. Mivel az LPD arra számít, hogy minden nyomtató alapból képes kinyomtatni bármilyen nyers szöveget, ezért a szövegszûrõ feladata, hogy a nyomtató számára gondoskodjon a tabulátorok, törlések és más egyéb speciális karakterek megfelelõ kezeléséről. Emellett ha olyan helyen vagyunk, ahol szükség van a nyomtatási feladatok nyilvántartására is, a szövegszûrõ ennek megoldására is képes, méghozzá úgy, hogy összeszámolja a kinyomtatott sorokat, és elosztja ezeket a nyomtató által oldalanként nyomtatott sorok számával. Egy szövegszûrõ a következõ paraméterekkel indulhat:

  ```
  szûrõnév [ -c ] -w szélesség -l hossz -i behúzás -n hozzáférés -h gépnév nyilvántartás
  ```

  ahol a

  `-c`
  : akkor jelenik meg, ha egy nyomtatási feladatot az `lpr -l` paranccsal adunk át

  szélesség
  : az /etc/printcap állományban definiált `pw` (page width, avagy oldalszélesség) tulajdonság értéke, ami alapbeállítás szerint 132

  hossz
  : a `pl` (page length, avagy oldalhossz) tulajdonság értéke, amely az alapbeállítás szerint 66

  behúzás
  : az `lpr -i` parancs megadása során használt behúzás mértéke, ami alapból 0

hozzáférés

a nyomtatást végzõ felhasználó hozzáférésének megnevezése

gépnév

a gép neve, amirõl a nyomtatási feladat érkezett

nyilvántartás

ez a nyilvántartást tároló állomány `af` tulajdonsággal definiált neve

- A *konverziós szûrõk* (conversion filter) egy adott állományformátumot hoznak a nyomtató számára értelmes formára. Például ditroff adatok közvetlenül ugyan nem nyomtathatóak, azonban a ditroff állományokhoz tudunk telepíteni egy olyan szûrõt, amely a ditroff adatokat a nyomtató számára is emészthetõ és nyomtatható formájúvá teszi. A Konverziós szûrõk címû szakasz tud ezekrõl többet mondani. Ilyen esetekben kérhetünk nyilvántartást. A konverziós szûrõk az alábbi paraméterekkel indulhatnak:

  ```
  szûrõnév -x pixelszélesség -y pixelmagasság -n hozzáférés -h gépnév nyilvántartás
  ```

  ahol a *pixelszélesség* a `px` tulajdonság értékébõl (ami alapból 0), a *pixelmagasság* a `py` tulajdonság értékébõl (ami alapból szintén 0) származik.

- A *kimeneti szûrõ* (output filter), ami csak akkor aktív, ha a szövegszûrõ nem, vagy ha engedélyeztük fejléclapok nyomtatását. Tapasztalatom szerint az ilyen szûrõket ritkán használják. A Kimeneti szûrõk címû szakasz mutatja be a mûködésüket. Ekkor csupán két paraméterünk van:

  ```
  szûrõnév -w szélesség -l hosszúság
  ```

  amik rendre megegyeznek a szövegszûrõk `-w` és `-l` paramétereivel.

A szûrõk *ki is tudnak lépni* a következõ kódokkal (exit status):

0

A szûrõ sikeresen kinyomtatta az állományt.

1

A szûrõnek nem sikerült kinyomtatnia az állományt, azonban szeretné, ha az LPD újból megpróbálkozna vele. Az LPD tehát ebben az esetben újraindítja a szûrõt.

2

A szûrõnek nem sikerült kinyomtatnia az állományt, és nem is kívánja újra megpróbálni. Ekkor az LPD eldobja az állományt.

A FreeBSD kiadásokban megtalálható /usr/libexec/lpr/lpf szövegszûrõ képes a kapott szélesség és hossz paraméterekkel megállapítani az oldaltöréseket és a nyomtató használatát nyilvántartani, amihez a hozzáférés, gépnév és nyilvántartás adatait használja fel.

Amikor majd igyekszünk mellé újabb szûrõket beszerezni, ne felejtsük el ellenõrizni, hogy együtt tudnak-e mûködni az LPD-vel. Ha a válasz igen, akkor a fentebb említett paraméterek mindegyikét ismerniük kell. Az általános használatra készült szûrõk készítése során mi magunknak is be kell

tartanunk ezeket az elvárásokat.

Szöveges nyomtatási feladatok PostScript® nyomtatókon

Ha csak egyedül dolgozunk a számítógépen és PostScript® (vagy bármilyen más nyelvet ismerõ) nyomtatónk van, valamint megígérjük, hogy soha nem küldünk sem mi, sem pedig nem küldetünk semmilyen más programmal nyers szöveget a nyomtatóra, akkor átléphetjük ezt a szakaszt.

Ha viszont egyaránt akarunk küldeni PostScript® programot és nyers szöveget tartalmazó nyomtatási feladatot a nyomtatónak, akkor ehhez kénytelenek vagyunk a rendszerünket beállítani. Elõször is szükségünk van szövegszûrõre, ami megállapítja, hogy a frissen érkezett nyomtatási feladat nyers szöveget vagy PostScript® programot tartalmaz-e. Minden PostScript®-alapú feladat a `%!` karaktersorozattal kezdõdik (a többi esetben olvassuk a nyomtató leírását). Szóval, ha a nyomtatandó állomány elsõ két karaktere ilyen, akkor egy PostScript® programmal van dolgunk és közvetlenül továbbküldhetjük a nyomtatási feladatot a nyomtatónak. Minden más esetben a szûrõnek elõbb át kell alakítania a szöveget PostScript® nyelvre.

Hogyan érhetjük el mindezt?

Ha soros nyomtatónk van, akkor erre a feladatra az `lprps` parancs tökéletes. Az `lprps` egy olyan PostScript® szûrõ, amely mind a két irányban képes közvetíteni. Folyamatosan rögzíti egy állományba a nyomtató állapotát, így a felhasználók és rendszergazdák pontosan látják a nyomtató jelenlegi állapotát (például `toner low` (a toner hamarosan kifogy) vagy `paper jam` (a papír beragadt)). Ami viszont sokkal lényegesebb, hogy a `psif` nevû program képes megmondani az érkezõ nyomtatási feladat valódi típusát, és ennek megfelelõen meg tudja hívni nyers szöveg átalakítására a `textps` (egy másik program, amit a `lprps` mellé kapunk) parancsot. Ezután az `lprps` elküldi a feladatot a nyomtatónak.

Az `lprps` a FreeBSD Portgyûjteményének része (lásd A Portgyûjtemény), ezért a használni kívánt papír méretétõl függõen pillanatok alatt magunk is letölhetjük, fordíthatjuk és telepíthetjük a print/lprps-a4 és print/lprps-letter csomagok valamelyikét. Az `lprps` telepítése után egyszerûen csak adjuk meg a `psif` elérési útvonalát. Ha tehát telepítettük a Portgyûjteményból az `lprps` csomagot, akkor egy soros portra csatlakozó PostScript® nyomtató esetén ezt kell beírnunk az /etc/printcap állományba:

```
:if=/usr/local/libexec/psif:
```

Ezenkívül még az `rw` tulajdonsággal meg kell mondanunk az LPD-nek, hogy a nyomtatót írásra és olvasásra nyissa meg.

Amennyiben a PostScript® nyomtatónk a párhuzamos porton csatlakozik (és amiért a nyomtatónk nem képes az `lprps` által igényelt kétirányú kommunikációra), szövegszûrõként a következõ szkriptet fogjuk használni:

```
#!/bin/sh
#
#  psif - PostScript vagy nyers szöveg nyomtatása PostScript nyomtatón
#  Ez a szkriptes változat, NEM pedig az lprps-hez mellékelt szûrõ
```

```
#  (a /usr/local/libexec/psif állomány)!
#

IFS="" read -r first_line
first_two_chars=`expr "$first_line" : '\(..\)'`

if [ "$first_two_chars" = "%!" ]; then
    #
    #  PostScript: nyomtassuk ki.
    #
    echo "$first_line" && cat && printf "\004" && exit 0
    exit 2
else
    #
    #  Nyers szöveg: alakítsuk át, majd nyomtassuk ki.
    #
    ( echo "$first_line"; cat ) | /usr/local/bin/textps && printf "\004" && exit 0
    exit 2
fi
```

A fentebb szereplõ szkriptben a `textps` programot használjuk a nyers szövegek PostScript® programokra alakításához, de helyette bármilyen más konvertáló programot is igénybe vehetünk. A FreeBSD Portgyûjteményében (lásd A Portgyûjtemény) találhatunk erre a célra egy `a2ps` nevû programot is, amit esetleg érdemes lehet közelebbrõl megnéznünk.

PostScript® szimulációja nem PostScript® nyomtatókon

A PostScript® a magas színvonalú betûszedés és nyomtatás *de facto* szabványa. Emellett azonban a PostScript® egy *költséges* szabvány is. Az Aladdin Enterprises-nak hála azonban létezik egy hozzá hasonló szabad szoftver, a Ghostscript, amely képes FreeBSD-n is futni. A Ghostscript képes a legtöbb PostScript® állomány olvasására, megjelenítésére mindenféle eszközökön, beleértve a PostScript®et nem ismerõ nyomtatókat is. A Ghostscript és egy speciális szövegszûrõ telepítésével el tudjuk érni, hogy egy nem PostScript® nyomtató valódi PostScript® nyomtatóként viselkedjen.

Ha telepíteni szeretnénk, a Ghostscript megtalálható a FreeBSD Portgyûjteményében. Innen tehát magunk is könnyedén le tudjuk tölteni, fordítani és telepíteni.

A PostScript® nyomtatás szimulációjához elõször egy szûrõ segítségével észre kell vennünk, hogy egy PostScript® formátumú állományt készülünk kinyomtatni. Ha nem ilyen a nyomtatási feladat, akkor egyenesen a nyomtatóra küldjük, azonban minden más esetben elõször a Ghostscript segítségével átalakítjuk egy olyan formátumba, amit a nyomtató is képes feldolgozni.

Nézzünk erre egy példát: a most következõ szövegszûrõ a Hewlett Packard DeskJet 500-as nyomtatóihoz használható. Más nyomtató esetén cseréljük ki a `gs` (Ghostscript) parancs `-sDEVICE` paraméterét a neki megfelelõre. (A telepített Ghostscript által ismert nyomtatók listáját a `gs -h` paranccsal kérdezhetjük le.)

```
#!/bin/sh
#
```

```
#  ifhp - Ghostscripttel szimulált Postscript nyomtatás DeskJet 500-on
#  Helye: /usr/local/libexec/ifhp

#
#  LF karaktereket CR+LF-ként kezeljük (elkerülve ezzel a HP/PCL
#  nyomtatókon a "lépcsõzést"):
#
printf "\033&k2G" || exit 2

#
#  Az állomány elsõ két karakterének beolvasása
#
IFS="" read -r first_line
first_two_chars=`expr "$first_line" : '\(..\)'`

if [ "$first_two_chars" = "%!" ]; then
    #
    #  Ez PostScript: küldjük át a Ghostscripten és nyomtassuk ki.
    #
    /usr/local/bin/gs -dSAFER -dNOPAUSE -q -sDEVICE=djet500 \
      -sOutputFile=- - && exit 0
else
    #
    #  Nyers szöveg vagy HP/PCL, ezért küldjük át közvetlenül. Az utolsó
    #  lap kidobásához küldünk még egy lapdobást is.
    #
    echo "$first_line" && cat && printf "\033&l0H" &&
exit 0
fi

exit 2
```

Befejezésül az `if` tulajdonságon keresztül értesítenünk kell errõl a szûrõrõl az LPD-t is:

```
:if=/usr/local/libexec/ifhp:
```

Készen is vagyunk! Most már nyugodtan beírhatjuk, hogy `lpr sima.szöveg` vagy `lpr akármi.ps`, mind a kettõnek ki kell tudnia nyomtatódnia.

Konverziós szûrõk

Miután elvégeztük az Alacsonyszintû nyomtatóbeállítás címû szakaszban leírt beállításokat, a (nyers ASCII szöveg mellett) kedvenc állományformátumainkhoz is minden bizonnyal szeretnénk telepíteni néhány konverziós szûrõt.

Miért használjunk konverziós szûrõket?

A konverziós szûrõk segítségével állományok mindenféle formátumait könnyen ki tudjuk nyomtatni. Például tegyük fel, hogy sokat dolgozunk a TeX betûszedõ rendszerrel és egy

PostScript® nyomtatónk van. Minden alkalommal, amikor egy DVI állományt hozunk létre a TeX forrásból, azt közvetlenül még nem tudjuk a nyomtatóra küldeni. Ehhez a következõ parancsokat kell kiadnunk:

```
% dvips hínár-elemzés.dvi
% lpr hínár-elemzés.ps
```

Ha telepítünk egy konverziós szûrõt a DVI állományokhoz, meg tudjuk spórolni ezt a manuális átalakítási lépést azzal, hogy átadjuk ezt a feladatot az LPD-nek. Így ezután mindig, amikor egy DVI állományt akarunk kinyomtatni, csupán egyetlen lépésre lesz szükségünk:

```
% lpr -d hínár-elemzés.dvi
```

Az LPD-nek a `-d` paraméterrel adjuk meg, hogy a nyomtatás elõtt hajtsa végre a DVI átalakítását. A Formázási és konverziós beállítások címû szakaszban találjuk meg a többi konverziós opciót.

Minden olyan konverziós beállításhoz, amit használni szeretnénk a nyomtatóval, telepítenünk kell egy *konverziós szûrõt* (conversion filter) és meg kell adnunk a nevét az /etc/printcap állományban. A konverziós szûrõk az egyszerû nyomtatóbeállításnál szereplõ szövegszûrõkhöz hasonlítanak (lásd A szövegszûrõ telepítése szakasz) azzal a kivétellel, hogy a nyers szövegek kinyomtatása helyett ezek a szûrõk a nyomtató számára értelmes formátumra alakítják az állományokat.

Milyen konverziós szûrõket érdemes telepíteni?

Olyan konverziós szûrõket telepítsünk, amelyekre gyakran szükségünk lehet. Ha például sok DVI adatot szeretnénk nyomtatni a jövõben, akkor használjunk DVI konverziós szûrõt, vagy ha sok troff formátumú adatot nyomtatunk, akkor minden bizonnyal jól fog jönni egy troff szûrõ.

A következõ táblázat foglalja össze azokat a szûrõket, amelyekkel az LPD képes együttmûködni. Megtudhatjuk, hogy az /etc/printcap állományban melyik tulajdonság tartozik hozzájuk és hogyan hívjuk meg ezeket az `lpr` paranccsal:

Állománytípus	Tulajdonság az /etc/printcap állományban	Az lpr kapcsolója
cifplot	`cf`	`-c`
DVI	`df`	`-d`
plot	`gf`	`-g`
ditroff	`nf`	`-n`
FORTRAN forrás	`rf`	`-f`
troff	`tf`	`-f`
raster	`vf`	`-v`
nyers szöveg	`if`	nincs, `-p`, vagy `-l`

A példánkban tehát a `lpr -d` parancs használata arra utal, hogy a nyomtatónak az /etc/printcap

állományból a df tulajdonságára van szüksége.

Minden hadakozás ellenére állíthatjuk, hogy a FORTRAN források és a plot által használt szövegek formátuma napjainkra már elavultnak tekinthetõ. Ezért ezekhez az opciókhoz a saját szûrõinkkel tetszõleges formázási lehetõségeket rendelhetünk. Például, ha Printerleaf (az Interleaf asztali kiadványszerkesztõ formátuma) állományokat szeretnénk közvetlenül nyomtatni, akkor valószínûleg nem lesz szükségünk plot állományokra. Ezért a gf tulajdonságnak megadhatunk egy Printerleaf konverziós szûrõt, amelyen keresztül aztán a felhasználók az lpr -g paranccsal Printerleaf állományokat tudnak nyomtatni.

Konverziós szûrõk telepítése

Mivel a konverziós szûrõk az alap FreeBSD rendszeren kívülre kerülnek, ezért ezeket minden valószínûség szerint valahol a /usr/local könyvtárban találjuk meg. Ezen belül is általában a /usr/local/libexec könyvtárban fordulnak elõ, mivel ezeket csak az LPD futtatja, senki másnak nincs rájuk szüksége.

A konverziós szûrõk aktiválásához az /etc/printcap állományban egyszerûen adjuk meg az alkalmas tulajdonságoknak megfelelõ szûrõk elérési útvonalait.

A példánkban most felveszünk egy DVI konverziós szûrõt a bamboo nevû nyomtatóhoz. Itt ismét láthatjuk a korábban használt /etc/printcap állományt, ahol most azonban a bamboo nevû nyomtatónál hozzáadtunk egy df tulajdonságot:

```
#
#  /etc/printcap (rose) -  egy df szûrõ hozzáadása a bamboo
#  nevû nyomtatóhoz
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:\
        :lp=/dev/lpt0:\
        :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:\
        :lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:rw:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

A DVI szûrõ ebben az esetben a /usr/local/libexec/psdf néven elérhetõ aprócska szkript. Ezt találhatjuk benne:

```
#!/bin/sh
#
#  psdf - DVI szûrõ PostScript nyomtatóhoz
#  Helye: /usr/local/libexec/psdf
#
# Az lpr -d parancs hatására hívódik meg
#
```

```
exec /usr/local/bin/dvips -f | /usr/local/libexec/lprps "$@"
```

A szkript a `dvips` parancsot szûrõként futtatja (az `-f` paraméterrel) a szabványos bemenetrõl, ahova a nyomtatási feladatot is kapja. Ezután elindítja az `lprps` PostScript® szûrõt (lásd a Szöveges nyomtatási feladatok PostScript® nyomtatókon címû szakaszt) az LPD által átadott paraméterekkel. Az `lprps` parancs ezekkel a paraméterekkel tartja nyilván az így kinyomtatott lapokat.

További példák konverziós szûrõkre

A konverziós szûrõk telepítésének nincs bevált receptje, ezért ebben a szakaszban bemutatunk rájuk néhány mûködõ illusztrációt. Ezeket tudjuk felhasználni saját szûrõk elkészítésére. Vagy ha megtehetjük, használjuk közvetlenül ezeket.

Ebben a példa szkriptben Hewlett Packard LaserJet III-Si nyomtatókhoz hozunk létre raszteres (pontosabban GIF formátumú) konverziós szûrõt:

```
#!/bin/sh
#
#  hpvf - GIF állományokat konvertál át HP/PCL-be, majd kinyomtatja
#  Helye: /usr/local/libexec/hpvf

PATH=/usr/X11R6/bin:$PATH; export PATH
giftopnm | ppmtopgm | pgmtopbm | pbmtolj -resolution 300 \
    && exit 0 \
    || exit 2
```

Úgy mûködik, hogy a GIF állományt elõször PNM (portable anymap), utána PGM (portable graymap), majd PBM (portable bitmap) formátumúra alakítja, amibõl végül LaserJet/PCL-kompatibilis adat lesz.

Ez lesz a hozzá tartozó /etc/printcap állomány:

```
#
#  /etc/printcap (orchid)
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
        :lp=/dev/lpt0:sh:sd=/var/spool/lpd/teak:mx#0:\
        :if=/usr/local/libexec/hpif:\
        :vf=/usr/local/libexec/hpvf:
```

A most következõ szkript a groff betûszedû rendszerbõl érkezõ troff adatokat alakítja át a `bamboo` nevû PostScript® nyomtató számára:

```
#!/bin/sh
#
#  pstf - a groff troff adait alakítja PS-re, majd kinyomtatja
#  Helye: /usr/local/libexec/pstf
```

```
#
exec grops | /usr/local/libexec/lprps "$@"
```

A szkript az `lprps` parancs segítségével kommunikál a nyomtatóval. Ha a nyomtatónk párhuzamos porton csatlakozik, akkor helyette ezt a szkriptet használjuk:

```
#!/bin/sh
#
#  pstf - a groff troff adatait alakítja PS-re, majd kinyomtatja
#  Helye: /usr/local/libexec/pstf
#
exec grops
```

Kész is! A szûrõ életrekeltéséhez mindössze ennyit kell beillesztenünk az /etc/printcap állományba:

```
:tf=/usr/local/libexec/pstf:
```

Most pedig jöjjön a FORTRAN szerelmeseinek szívét megmelengetõ szkript. Ez egy olyan szövegszûrõ, amely bármelyik nyers szöveget közvetlenül kezelni tudó nyomtató esetén mûködik. A `teak` nevû nyomtatóhoz helyezzük be:

```
#!/bin/sh
#
# hprf - FORTRAN szövegszûrõ LaserJet 3si-hez
# Helye: /usr/local/libexec/hprf
#

printf "\033&k2G" && fpr && printf "\033&l0H" &&
 exit 0
exit 2
```

Az /etc/printcap állományban a `teak` nyomtatóhoz a következõ sor beírásával tudjuk engedélyezni ezt a szûrõt:

```
:rf=/usr/local/libexec/hprf:
```

Most pedig következzen egy utolsó, de az eddigieknél valamivel összetettebb példa. Ebben a korábban bemutatott `teak` nevû LaserJet nyomtatóhoz fogunk hozzáadni egy DVI szûrõt. Elõször is következzen a mûvelet egyszerûbb része: bõvítsük ki az /etc/printcap állományt a DVI szûrõ helyének megadásával:

```
:df=/usr/local/libexec/hpdf:
```

Ezután következzék a nehezebb rész: a szûrõ elkészítése. Ehhez szükségünk lesz egy DVI-rõl

LaserJet/PCL-re alakító programra. A FreeBSD Portgyûjteményében (lásd A Portgyûjtemény) találunk is egyet: a csomag neve print/dvi2xx. A csomag telepítésével megkapjunk a nekünk kellõ dvilj2p programot, ami képes DVI-t LaserJet IIp, LaserJet III és a LaserJet 2000 típusok által ismert kódokra fordítani.

A dvilj2p felhasználásától függetlenül a hpdf néven létrehozni kívánt szûrõnk még így is bonyolult lesz, hiszen a dvilj2p nem tud olvasni a szabványos bemenetrõl, hanem mindenáron egy állománnyal akar dolgozni. Sõt, olyan állománnyal, amelynek .dvi kiterjesztése van, ezért még a /dev/fd/0 (vagyis a szabványos bemenethez tartozó eszközleíró) használata is akadályokba ütközik.

Üröm még az örömünkben, hogy a /tmp könyvtárat sem tudjuk felhasználni ideiglenes link létrehozására: a szimbolikus linkeket a bin felhasználó és csoport birtokolja, a szûrõt pedig a daemon felhasználó futtatja. A /tmp könyvtárban rááadásul csak a tulajdonosaik képesek állományokat átnevezni vagy törölni (sticky bit). Ezért a szûrõ ugyan létre tudna hozni egy linket, azonban ezt a feladata végeztével nem lesz majd képes törölni, mivel a link egy másik felhasználóhoz tartozik.

Ezért a szûrõ az aktuális könyvtárban fogja létrehozni ezt a szimbolikus linket, ami jelen esetünkben a nyomtatási rendszer által használt könyvtár lesz (ezt az /etc/printcap állomány sd tulajdonságával adjuk meg). Itt remekül el tudják végezni a feladataikat a szûrõk, különösen mivel (néha) több hely van itt, mint a /tmp könyvtárban.

Végül lássuk magát a szûrõt:

```
#!/bin/sh
#
#  hpdf - DVI adat nyomtatása HP/PCL nyomtatón
#  Helye: /usr/local/libexec/hpdf

PATH=/usr/local/bin:$PATH; export PATH

#
#  Létrehozunk egy függvényt az átmeneti állományok törlésére. Ezek
#  az aktuális könyvtárban jönnek létre, ami pedig a nyomtatási
#  rendszer adott nyomtatóhoz tartozó könyvtára lesz.
#
cleanup() {
   rm -f hpdf$$.dvi
}

#
#  Létrehozunk egy függvényt a súlyos hibák kezelésére: írassunk ki
#  egy adott üzenetet és lépjünk ki a 2-es hibakóddal.  Ezzel üzenünk
#  az LPD-nek, hogy ne hajtsa végre újra a nyomtatási feladatot.
#
fatal() {
    echo "$@" 1>&2
    cleanup
    exit 2
}
```

```
#
#  Ha a felhasználó eltávolítja a nyomtatási feladatot a sorból, akkor az
#  LPD egy SIGINT jelzést fog küldeni, ezért próbáljuk meg azt elkapni
#  (néhány más egyéb jelzéssel együtt), így még tudjuk törölni az
#  ideiglenesen # létrehozott állományokat.
#
trap cleanup 1 2 15

#
#  Gondoskodjunk róla, hogy a feladat megkezdésekor még egyetlen
#  használt állomány sem létezik.
#
cleanup

#
#  Kössük össze a szabványos bemenetet egy DVI állománnyal (amit
#  majd nyomtatni akarunk).
#
ln -s /dev/fd/0 hpdf$$.dvi || fatal "Cannot symlink /dev/fd/0"

#
#  LF = CR+LF
#
printf "\033&k2G" || fatal "Cannot initialize printer"

#
#  Alakítsuk át az adatot és nyomtassunk. A dvilj2p által visszaadott érték
#  nem túlságosan megbízható, ezért ne is foglalkozzunk vele.
#
dvilj2p -M1 -q -e- dfhp$$.dvi

#
#  Takarítsunk el magunk után és lépjünk ki szabályosan
#
cleanup
exit 0
```

Automatikus konverziók: a konverziós szûrõk helyett

A konverziós szûrõk sokat segítenek egy kényelmes nyomtatási környezet kialakításában, azonban a használatukhoz a felhasználóknak (az lpr(1) parancson keresztül) egyenként hivatkozniuk kell rájuk. Ha a rendszerünk felhasználói nem eléggé mûveltek számítástechnikai téren, akkor még egy szûrõ megadása is zavaró lehet számukra. Ami még ennél is rosszabb, hogy egy rosszul megadott szûrõ hatására a nyomtató sem fogja jól kezelni az adott állomány formátumát és erre válaszul akár többszáz lapot is pillanatok alatt kiköphet magából.

A konverziós szûrõk telepítése helyett gyakran csak egy (alapértelmezett) szövegszûrõre van szükségünk, amely kideríti a nyomtatandó állomány pontos formátumát és magától elindítja a neki megfelelõ konverziós szûrõt. Ilyen esetekben például a `file` parancs pont a hasznunkra válhat. Persze *bizonyos* állománytípusok közt nagyon nehéz különbséget tenni - de ezekre továbbra is

adhatunk még külön konverziós szûrõket.

A FreeBSD Portgyûjteményében találhatunk egy `apsfilter` elnevezésû szövegszûrõt (print/apsfilter), ami képes ilyen automatikus konverzióra. Képes felismerni a nyers szöveget, PostScript® programokat, DVI és szinte bármilyen formátumú állományokat, lefuttatni rájuk a megfelelõ átalakításokat, majd kinyomtatni ezeket.

Kimeneti szûrõk

Az LPD nyomtatási rendszer kezel egy eddig még nem tárgyalt szûrõtípust is: ez a kimeneti szûrõ. A kimeneti szûrõ a szövegszûrõhöz hasonlóan csak nyers szöveg nyomtatására használatos, de tartalmaz néhány egyszerûsítést. Ha kizárólag csak kimeneti szûrõket alkalmazunk, akkor:

- Az LPD az egész nyomtatási feladathoz egyetlen kimeneti szûrõt fog használni, nem pedig minden állományhoz külön.
- Az LPD a kimeneti szûrõ számára nem nyújt semmilyen segítséget a nyomtatási feladaton belül szereplõ állományok kezdetének vagy végének megállapításában.
- Az LPD a szûrõnek nem adja át sem a felhasználó hozzáférését, sem pedig gépnevét, ezért nyilvántartásra nem alkalmas. Mindent összegezve lényegében csak két paramétert kap meg:

  ```
  szûrõnév -w_szélesség_ -l_hossz_
  ```

  ahol a *szélesség* a kérdéses nyomtató `pw` tulajdonságából, a *hossz* pedig a `pl` tulajdonságából származik.

Ne bûvöljön el minket a szûrõ egyszerûsége! Ha például a nyomtatási feladatban minden állományt újabb lapon szeretnénk kezdeni, akkor azt kimeneti szûrõvel *nem tudjuk megoldani*. Erre a célra használjunk szövegszûrõt (másik nevén bemeneti szûrõt), lásd A szövegszûrõ telepítése szakaszt. Továbbá, a kimeneti szûrõ valójában *sokkal bonyolultabb* abban a tekintetben, hogy a beérkezõ adatok közül neki kell kikeresnie a speciális jelentéssel bíró karaktereket ugyanúgy, ahogy az LPD helyett saját magának kell küldenie a jelzéseket.

Azonban a kimeneti szûrõk használata *elkerülhetetlen*, ha például fejléclapokat akarunk nyomtatni, és esetleg még különbözõ inicializálásra használatos speciális kódokat vagy karakterláncokat akarunk ez elõtt kiküldeni. (Ellenben *badarság* a fejléclapoktól követelni a felhasználó adatait, hiszen az LPD a kimeneti szûrõnek nem ad semmilyen erre vonatkozó információt.)

Egyetlen nyomtató esetén az LPD egyaránt lehetõvé teszi kimeneti, szöveg- és más egyéb szûrõk használatát. Ilyenkor az LPD a kimeneti szûrõn keresztül csak a fejlécet tartalmazó oldal (lásd a Fejléclapok szakaszt) nyomtatását indítja el. Ezt követõen az LPD arra számít, hogy a kimeneti szûrõ két karakter, az ASCII 031 és az ezt követõ ASCII 001, hatására *leállítja magát*. Amikor tehát a kimeneti szûrõ érzékeli ezt a két karaktert (031, 001), akkor a `SIGSTOP` jelzéssel le kell állnia. Miután az LPD lefuttatta a többi szûrõt, a `SIGCONT` jelzéssel újraindítja a kimeneti szûrõt.

Ha van kimeneti szûrõnk, de *nincs* szövegszûrõnk, akkor az LPD minden további feldolgozás nélkül továbbadja a nyomtatási feladatot a kimeneti szûrõnek. Ahogy már korábban is említettük, a kimeneti szûrõ a nyomtatási feladatban levõ összes állományt egymás után nyomtatja ki, lapdobások vagy bármilyen más papírmozgatás nélkül, ezért valószínûleg *nem* ez kell nekünk. Az esetek túlnyomó részében ehhez elég egy szövegszûrõ.

A korábban szövegszûrõként beharangozott `lpf` program kimeneti szûrõként is képes funkcionálni. Ha szükségünk lenne egy gyorsan összecsapható kimeneti szûrõre, és nem akarunk a speciális karakterek, valamint a jelzések küldésével elidõzni, akkor próbálkozzunk az `lpf` használatával. Az `lpf` parancsot mellesleg becsomagolhatjuk egy olyan szkriptbe is, amely elvégzi a nyomtató számára szükséges inicializálást.

Az `lpf` szövegszûrõ

A FreeBSD bináris terjesztéséhez mellékelt /usr/libexec/lpr/lpf program egy szövegszûrõ (bemeneti szûrõ), amely képes (az `lpr -i` paranccsal hozzáadott nyomtatási feladatokat) tabulálni, (az `lpr -l` paranccsal felvett nyomtatási feladatokban) a vezérlõkaraktereket figyelemen kívül hagyni, a nyomtatási feladatban elõforduló törlések és behúzások nyomtatási pozícióját igazítani és nyilvántartani a kinyomtatott lapokat. Kimeneti szûrõként is tud viselkedni.

Az `lpf` szûrõ rengeteg nyomtatási környezetben felhasználható. Habár nem képes a nyomtatónak inicializáló jelsorozatokat küldeni, mégis könnyû olyan szkriptet írni, amely elvégzi ezeket a hiányzó kezdeti beállításokat, majd lefuttatja az `lpf` szûrõt.

Az `lpf` akkor lesz képes helyesen számolni a kinyomtatott lapokat, ha ehhez az /etc/printcap állományban jól töltjük ki a `pw` és `pl` tulajdonságokat. Ezen értékek segítségével határozható meg ugyanis, hogy mennyi szöveg fért rá egy lapra és így mennyi lapot emésztett fel az adott felhasználó által küldött nyomtatási feladat. A nyomtatás nyilvántartásával kapcsolatban A nyomtató használatának nyilvántartása címû szakaszt érdemes elolvasni.

9.4.2. Fejléclapok

Ha *nagyon sok* felhasználónk van, és sok különbözõ nyomtatót is használnak, akkor elõbb vagy utóbb minden bizonnyal elkerülhetetlenné fog válni a *fejléclapok* használata.

A fejléc-, vagy más néven *munka-* vagy *elválasztó lapok* segítik elõ az elvégzett nyomtatási feladatok azonosítását. A többi dokumentumtól eltérõ módon, általában dekoratív keretben, nagy, vastag betûkkel nyomtatódnak ki, hogy a halomnyi papír között a felhasználók könnyedén megtalálhassák az elküldött nyomtatási feladataik eredményét. Természetesen a fejléclapok nyilvánvaló hátulütõje, hogy így minden nyomtatási feladathoz még egy lappal többet kell elhasználni és mivel gyakorlatilag néhány percnél tovább nincs is rájuk szükség, meglehetõsen hamar a kukába kerülnek. (A fejléclapok nyomtatási feladatonként jönnek létre, nem pedig a nyomtatási feladatokban levõ állományokhoz egyenként, ezért nem is akkora pazarlás ez.)

Az LPD rendszer képes magától fejléclapokat készíteni a nyomtatásokhoz, *amennyiben* a nyomtatónk képes közvetlenül nyers szöveget nyomtatni. Ha PostScript® nyomtatónk van, akkor ennek legyártásához egy külsõ programra van szükségünk, lásd a Fejléclapok PostScript® nyomtatókon szakaszt.

A fejléclapok engedélyezése

Az Alacsonyszintû nyomtatóbeállítás címû szakaszban az /etc/printcap állományban a `sh` (úgy mint "suppress header") tulajdonsággal kikapcsoltuk a fejléclapokat. A fejléclapok engedélyezéséhez mindössze el kell távolítanunk ezt az `sh` tulajdonságot.

Ez túl egyszerû, nemde?

Igen, ez így van. *Elõfordulhat*, hogy szükségünk van még egy olyan kimeneti szûrõre is, amely inicializáló karaktereket küld a nyomtatónak. Íme egy példa ehhez a Hewlett Packard PCL-kompatibilis nyomtatói esetére:

```
#!/bin/sh
#
#  hpof - Kimeneti szûrõ Hewlett Packard PCL-kompatibilis nyomtatókhoz
#  Helye: /usr/local/libexec/hpof

printf "\033&k2G" || exit 2
exec /usr/libexec/lpr/lpf
```

Az `of` tulajdonsággal adjuk meg a kimeneti szûrõt. A Kimeneti szûrõk szakaszban errõl részletesebben is olvashatunk.

A korábban ismertetett `teak` nevû nyomtatóhoz most az alábbi minta /etc/printcap állományt mellékeljük. Itt engedélyeztük a fejléclapokat és hozzátettük az iménti kimeneti szûrõt:

```
#
#  /etc/printcap (orchid)
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
        :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:\
        :if=/usr/local/libexec/hpif:\
        :vf=/usr/local/libexec/hpvf:\
        :of=/usr/local/libexec/hpof:
```

Mostantól kezdve, amikor a felhasználók a `teak` nyomtatón akarnak nyomtatni, minden nyomtatási feladathoz kapni fognak egy fejléclapot. Amennyiben a kedves felhasználók mégis keresgetni akarják a nyomtatásaikat, az `lpr -h` paranccsal tetszõleges módon letilthatják azokat. Az lpr(1) többi hasonló opcióját A fejléclapokhoz tartozó beállítások szakaszban találjuk.

Az LPD minden fejléclap után egy lapdobást küld. Ha erre a célra a nyomtatónk egy eltérõ karaktert vagy karaktersorozatot használ, akkor azt az /etc/printcap állomány `ff` tulajdonságával határozhatjuk meg.

A fejléclapok vezérlése

A fejléclapok engedélyezésével az LPD egy ún. *hosszú fejlécet* fog készíteni, vagyis a felhasználót, a gépet és a nyomtatási feladatot jól azonosító, egész lapot kitöltõ óriási betûket. Erre egy példa (amiben a `rose` nevû géprõl `kelly` küldte az "outline" elnevezésû nyomtatási feladatot):

```
     k                 ll       ll
     k                  l        l
     k                  l        l
     k   k     eeee     l        l     y    y
     k  k     e    e    l        l     y    y
```

```
      k k      eeeeee      l        l     y    y
      kk k     e           l        l     y    y
      k   k    e    e      l        l     y   yy
      k    k    eeee      lll      lll     yyy y
                                              y
                                          y    y
                                           yyyy

                                  ll
                          t         l        i
                          t         l
       oooo    u    u   ttttt       l       ii     n nnn     eeee
      o    o   u    u     t         l        i     nn   n   e    e
      o    o   u    u     t         l        i     n    n   eeeeee
      o    o   u    u     t         l        i     n    n   e
      o    o   u   uu     t  t      l        i     n    n   e    e
       oooo     uuu u      tt      lll      iii    n    n    eeee

      r rrr     oooo     ssss     eeee
      rr   r   o    o   s    s   e    e
      r        o    o    ss      eeeeee
      r        o    o      ss    e
      r        o    o   s    s   e    e
      r         oooo     ssss     eeee

                                             Job:  outline
                                             Date: Sun Sep 17 11:04:58 1995
```

Ezt követõen az LPD elküld még egy lapdobást is, ezért maga a nyomtatási feladat eredménye egy új oldalon fog kezdõdni (kivéve, ha az /etc/printcap állományban az adott nyomtatóhoz tartozó bejegyzésben megadtuk az `sf` (úgy mint "suppress form feeds", vagyis a lapdobások letiltása) tulajdonságot.

Ha úgy jobban tetszik, akkor az /etc/printcap állományban a `sb` tulajdonsággal az LPD utasítható *rövid fejlécek* készítésére is. Ilyenkor a fejléclap tartalma mindössze ennyi lesz:

```
rose:kelly  Job: outline  Date: Sun Sep 17 11:07:51 1995
```

Alapértelmezés szerint az LPD elõször a fejléclapot, majd a nyomtatási feladatot végzi el. Ezt a sorrendet az /etc/printcap állományban a `hl` (header last) tulajdonsággal meg tudjuk fordítani.

A nyomtató használatának nyilvántartása

Az LPD által felkínált fejléclapok használata során egyetlen irányelv érvényesül a nyilvántartásukban: a fejléclapok *költségmentesek*.

De miért?

Azért, mert kizárólag csak a kimeneti szûrõ képes a fejléclapok viselkedését irányítani, ami viszont

nem képes semmiféle nyilvántartásra, hiszen nem kapja meg az ehhez szükséges *felhasználói- vagy gépnév* információkat, illetve nyilvántartásokat. Emiatt fogalma sincs róla, hogy kit terhel az adott nyomtató használata. Úgy sem tudjuk megoldani a problémát, ha a szöveg- vagy konverziós szûrõkben (ahol már rendelkezésünkre állnak a felhasználó és a gépének adatai) "növeljük a lapok számát eggyel" a nyomtatási feladatban, mivel a felhasználók az `lpr -h` parancs használatával kedvük szerint letilthatják a fejléclapokat. Ezt ugyan alapvetõen a természetet óvni kívánó felhasználók részesítik elõnyben, de ettõl függetlenül sem erõszakolhatjuk rá mindenkire.

Az sem elég, ha minden szûrõ létrehozza a saját fejlécét (amiért aztán pénzt kérhetnénk). Mivel ha a felhasználók az `lpr -h` paranccsal le akarják tiltani a fejlécek használatát, attól a szûrõkhöz még mindig létrejönnek, hiszen az LPD a `-h` opcióról semmilyen értesítést nem küld át a szûrõknek.

Nos, ilyenkor mitévõk legyünk?

A lehetõségeink:

- Elfogadjuk az LPD elvét, és nem számítunk fel költséget a fejléclapokra.
- Az LPD helyett egy másik nyomtatási rendszert használunk, például az LPRng rendszert. A Más nyomtatási rendszerek címû szakaszban kiderül, milyen alternatívák érhetõek el az LPD kiváltására.
- Írjunk mi magunk egy *intelligens* kimeneti szûrõt. Normális esetben a kimeneti szûrõk nem valók másra, csupán a nyomtató alaphelyzetbe hozására vagy egyszerûbb karakterkonverziók elvégzésére. Fejléclapokhoz és nyers szöveget tartalmazó nyomtatási feladathoz remekül használható (ahol nincs szöveg- (avagy bemeneti) szûrõ). Azonban ha a nyers szövegekhez van szövegszûrõnk, akkor az LPD a kimeneti szûrõt csak a fejléclapokhoz indítja el. Emellett a kimeneti szûrõ az LPD által generált fejléc szövegébõl képes megmondani, melyik felhasználóhoz és géphez tartozik a szóbanforgó fejléc. A módszer egyetlen bökkenõje, hogy a nyilvántartásokat tároló állományról viszont még így se tudunk semmilyen információt szerezni (mivel nem kapjuk meg az `af` tulajdonsággal beállított állomány nevét). Ha azonban egy rendszerszinten elérhetõ állományba mentjük ezeket az adatokat, akkor akár bele is drótozhatjuk ezt a kimeneti szûrõbe. A kimeneti szûrõnek az adatok megtalálásában ilyenkor úgy tudunk segíteni, ha az /etc/printcap állományban az `sh` (rövid fejléc) tulajdonságot állítjuk be. De ez igazából sok hûhó semmiért, és a felhasználók is jobban megbecsülik az olyan nagylelkû rendszergazdát, aki nem számítja fel nekik a fejléclapokat.

Fejléclapok PostScript® nyomtatókon

Ahogy arról már korábban is szó esett, az LPD képes többféle nyomtató számára is megfelelõ, nyers szövegû fejléclapokat készíteni. Persze a PostScript® közvetlenül nem képes nyers szövegek nyomtatására, ezért az LPD ezen lehetõsége lényegében használhatatlan - többnyire.

Ilyen helyzetben a fejléclapok használatának nyilvánvaló módja, hogy minden szövegszûrõt fejlécek gyártására utasítunk. Ezek a szûrõk a felhasználóról és a gépérõl kapott információkból össze tudják állítani a megfelelõ fejléclapot. A megoldás hátránya, hogy ez még olyankor is megtörténik, amikor a felhasználók az `lpr -h` paranccsal küldik a nyomtatási feladataikat.

Kísérletezzünk egy kicsit ezzel a módszerrel! A most következõ szkript három paramétert fogad el (a felhasználó hozzáférést, a gép és a nyomtatási feladat nevét), majd ezekbõl létrehoz egy egyszerû PostScript® formátumú fejlécet:

```
#!/bin/sh
#
#  make-ps-header - PostScript fejléc létrehozása a szabvány kimenetre
#  Helye: /usr/local/libexec/make-ps-header
#

#
#  Ezek itt a PostScript által használt egységekben vannak megadva
#  (72/col vagy 28/cm).  Írjuk át az általunk használt papírméretre,
#  A4-re vagy amit éppen használunk:
#
page_width=612
page_height=792
border=72

#
#  A paraméterek ellenõrzése.
#
if [ $# -ne 3 ]; then
    echo "Usage: `basename $0` <user> <host> <job>" 1>&2
    exit 1
fi

#
#  Mentsük el ezeket, leginkább az olvashatóság miatt.
#
user=$1
host=$2
job=$3
date=`date`

#
#  Küldjük el a PostScript-kódot a szabványos kimenetre.
#
exec cat <<EOF
%!PS

%
%  Gondoskodjunk róla, hogy ne zavarjuk az utánunk következõ
%  felhasználó nyomtatási feladatának végrehajtását.
%
save

%
%  Csináljunk egy csúf vastag szegélyt, körbe a papíron.
%
$border $border moveto
$page_width $border 2 mul sub 0 rlineto
0 $page_height $border 2 mul sub rlineto
currentscreen 3 -1 roll pop 100 3 1 roll setscreen
```

```
$border 2 mul $page_width sub 0 rlineto closepath
0.8 setgray 10 setlinewidth stroke 0 setgray

%
%  Jelenítsük meg a felhasználó azonosítóját szép, feltûnõ
%  betûkkel.
%
/Helvetica-Bold findfont 64 scalefont setfont
$page_width ($user) stringwidth pop sub 2 div $page_height 200 sub moveto
($user) show

%
%  Most pedig mutassuk az unalmas részleteket.
%
/Helvetica findfont 14 scalefont setfont
/y 200 def
[ (Job:) (Host:) (Date:) ] {
200 y moveto show /y y 18 sub def }
forall

/Helvetica-Bold findfont 14 scalefont setfont
/y 200 def
[ ($job) ($host) ($date) ] {
        270 y moveto show /y y 18 sub def
} forall

%
% Ennyi lett volna.
%
restore
showpage
EOF
```

Ezzel a szkripttel pedig mindegyik konverziós és szövegszûrõ elõször létrehoz egy fejléclapot, majd elvégzi a felhasználó nyomtatási feladatát. Íme egy korábban már bemutatott DVI szûrõ, amit most kiegészítünk a fejléclapok használatával:

```
#!/bin/sh
#
#  psdf - DVI szûrõ PostScript nyomtatóhoz
#  Helye: /usr/local/libexec/psdf
#
#  Az lpr -d parancs hatására hívódik meg.
#

orig_args="$@"

fail() {
    echo "$@" 1>&2
    exit 2
```

```
}

while getopts "x:y:n:h:" option; do
    case $option in
        x|y)  ;; # Ignore
        n)    login=$OPTARG ;;
        h)    host=$OPTARG ;;
        *)    echo "LPD started `basename $0` wrong." 1>&2
              exit 2
              ;;
    esac
done

[ "$login" ] || fail "No login name"
[ "$host" ] || fail "No host name"

( /usr/local/libexec/make-ps-header $login $host "DVI File"
  /usr/local/bin/dvips -f ) | eval /usr/local/libexec/lprps $orig_args
```

Láthatjuk, hogy a szûrõnek a felhasználói- és a gépnév megállapításához végig kell néznie a paraméterek listáját. Ez lényegében minden más konverziós szûrõnél ugyanígy néz ki. Ez a lista azonban a szövegszûrõk esetén némileg eltér (lásd a Hogyan mûködnek a szûrõk? szakaszt).

Már az elõbbiekben is tárgyaltuk, hogy ez a megoldás, habár eléggé egyszerû, az `lpr` számára nem teszi lehetõvé a fejléclapok letiltását (a `-h` opció). Ha a felhasználóink kímélni akarják a fákat (vagy meg akarják úszni a fejléclapok égbeszökõ költségeit), akkor ezt nem tudják megtenni, hiszen a szûrõk minden nyomtatási feladathoz készíteni fognak fejléceket.

Ezt a korlátozást csak úgy tudjuk elsöpörni, ha bevetjük a A nyomtató használatának nyilvántartása szakaszban leírt cselt, tehát készítünk egy olyan kimeneti szûrõt, amely megkeresi az LPD-vel generált fejléceket és létrehozza azok PostScript® változatát. Ha valaki az `lpr -h` paranccsal küld nyomtatnivalót, akkor LPD nem készít hozzá fejléclapot, ahogy a kimeneti szûrõnk sem. A kimeneti szûrõ minden más esetben beolvassa az LPD által küldött szöveget és átküldi a neki megfelelõ PostScript® kódot a nyomtatóra.

Ha soros PostScript® nyomtatónk van, akkor használhatjuk a `psof` kimeneti szûrõhöz tartozó `lprps` parancsot is, ami pontosan az elõbbit végzi el. Hozzátennénk azonban, hogy a `psof` nem számolja a fejléclapokat.

9.4.3. Hálózati nyomtatás

A FreeBSD tud hálozaton is nyomtatni, vagyis tud távoli számítógépeknek is nyomtatási feladatot küldeni. A hálózati nyomtatás kifejezés általánosságban véve két különbözõ dologra utalhat:

- Egy távoli számítógéphez kapcsolt nyomtató hozzáférését. A géphez a nyomtató a hagyományos soros vagy párhuzamos csatolófelületen keresztül kapcsolódik, amit aztán az LPD alkalmas beállításával a hálózaton mindenki számára elérhetõvé teszünk. A Távoli számítógépekre csatlakoztatott nyomtatók címû szakasz errõl szól.
- Egy közvetlenül a hálózatra kapcsolt nyomtató hozzáférését. A nyomtató tehát rendelkezik még

egy hálózati csatlakozással is a hagyományos soros vagy párhuzamos felület mellett (vagy éppen helyett). Egy ilyen nyomtató a következõképpen mûködhet:

- Elfogadja az LPD kéréseit, és még képes nyomtatási feladatokat is tárolni. Ebben az esetben teljesen egyenértékû egy LPD alkalmazást futtató számítógéppel. Ekkor nincs más teendõnk, csak követnünk kell a Távoli számítógépeken telepített nyomtatók címû szakasz utasításait.
- Hálózati adatfolyamokkal dolgozik. Ebben az esetben a nyomtatót "hozzá kell kapcsolnunk" a hálózaton található egyik számítógéphez, ami majd a nyomtatási feladatok tárolásáért és folyamatos küldéséért lesz felelõs. A Nyomtatók hálózati adatcsatlakozással szakasz az ilyen fajtájú nyomtatók telepítésére tesz néhány javaslatot.

Távoli számítógépekre csatlakoztatott nyomtatók

Az LPD nyomtatási rendszer alapból képes más, szintén LPD-t (vagy vele kompatibilis rendszert) futtató számítógépekre nyomtatási feladatokat küldeni. Ezzel lényegében az egyik géphez hozzá tudunk kapcsolni egy nyomtatót, amit aztán a többiek számára elérhetõvé teszünk. Ez olyan nyomtatók esetében is mûködik, amelyek ismerik az LPD által alkalmazott protokollt.

A távoli nyomtatáshoz elõször telepítsük a nyomtatót valamelyik számítógépre az Alacsonyszintû nyomtatóbeállítás szakaszban leírtak szerint, és ezzel az lesz a *nyomtatószerverünk*. Ezután, amennyiben szükségesnek találjuk, végezzünk magasabb szintû nyomtatóbeállításokat is. Ne felejtsük el kipróbálni a nyomtatón, hogy rendesen mûködik az LPD mindegyik olyan beállításával, amit engedélyeztünk. Emellett gondoskodjunk minden olyan jogosultságról is, amivel a *helyi számítógéprõl* el tudjuk érni a *távoli számítógép* által felkínált LPD szolgáltatást (lásd Távoli számítógépekrõl érkezõ kérések szabályozása).

Ha olyan nyomtatót használunk, aminek a hálózati felülete kompatibilis az LPD rendszerrel, akkor az elõbb említett *nyomtatószerver* lényegében maga lesz a nyomtató, valamint a *nyomtató neve* a rajta beállított név. Ezzel kapcsolatban olvassuk el a nyomtatóhoz és/vagy a hálózati csatolójához mellékelt dokumentációt.

Amikor a Hewlett Packard Laserjet típusú nyomtatóit használjuk, a `text` nevû nyomtatónév magától elvégzi a LF és CRLF formátumú sortörések közti átalakítást, ezért ilyenkor nincs szükségünk a hpif szkriptre.

Ezután ha szeretnénk más gépek részére is elérhetõvé tenni a frissen telepített nyomtatónkat, adjuk meg mindegyikük /etc/printcap állományában a következõket:

1. Tetszõlegesen választott nevet, álneveket. Az egyszerûség kedvéért azonban itt érdemes ugyanazokat a neveket választani, mint amit a nyomtatószerveren is használunk.
2. Szándékosan hagyjuk az `lp` tulajdonságot üresen (`:lp=:`).
3. Hozzunk létre egy nyomtatási könyvtárat, és jelöljük meg a helyét az `sd` tulajdonsággal. Az LPD itt fogja összegyûjteni a nyomtatási feladatokat, mielõtt elküldené azokat a nyomtatószervernek.
4. Adjuk meg a nyomtatószerver nevét az `rm` tulajdonság segítségével.
5. Az `rp` tulajdonsággal adjuk meg a *nyomtatószerverre* csatlakoztatott nyomtató nevét.

Kész! Az /etc/printcap állományban már nem kell megadni konverziós szûrõket, oldalbeállításokat és semmi más egyebet.

Lássunk mindezekre egy példát. A rose nevû számítógéphez két nyomtató csatlakozik, a bamboo és a rattan. Most pedig beállítjuk, hogy az orchid nevû gép felhasználói képesek legyenek ezekkel a nyomtatókkal dolgozni. Ekkor a most következõk szerint fog kinézni az orchid (a Fejléclapok engedélyezése szakaszban bemutatott) /etc/printcap állománya. Tartalmazza a teak nevû nyomtató beállításait is, és ehhez fogjuk hozzáadni a rose másik két nyomtatóját:

```
#
#  /etc/printcap (orchid) - a rose két (távoli) nyomtatójának
#  hozzáadása
#

#
#  A "teak" egy helyi nyomtató, közvetlenül az orchidhoz
#  csatlakozik:
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
        :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:\
        :if=/usr/local/libexec/ifhp:\
        :vf=/usr/local/libexec/vfhp:\
        :of=/usr/local/libexec/ofhp:

#
#  A "rattan" rose-hoz csatlakozik, így küldhetünk neki nyomtatási
#  feladatot:
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :lp=:rm=rose:rp=rattan:sd=/var/spool/lpd/rattan:

#
#  A "bamboo" is a rose-hoz tartozik:
#
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :lp=:rm=rose:rp=bamboo:sd=/var/spool/lpd/bamboo:
```

Ezután már csak létre kell hoznunk a megfelelõ nyomtatási könyvtárakat az orchid nevû gépen:

```
# mkdir -p /var/spool/lpd/rattan /var/spool/lpd/bamboo
# chmod 770 /var/spool/lpd/rattan /var/spool/lpd/bamboo
# chown daemon:daemon /var/spool/lpd/rattan /var/spool/lpd/bamboo
```

Mostantól kezdve az orchid felhasználói képesek lesznek nyomtatni a rattan és bamboo nevû nyomtatókon is. Ezért, ha az orchid egyik felhasználója beírja, hogy:

```
% lpr -P bamboo -d sushi-leírás.dvi
```

Az orchid gépen mûködõ LPD rendszer ezt a nyomtatási feladatot a bemásolja a /var/spool/lpd/bamboo nevû nyomtatási könyvtárba és feljegyzi róla, hogy a nyomtatásához DVI

szûrõre lesz szükség. Ahogy rose gépen található bamboo nyomtatási könyvtárában elegendõ hely keletkezik, a két LPD átküldi egymás közt a rose nevû gépre az állományt. Ezután az állomány egészen addig várakozik a rose nyomtatási sorában, amíg végezetül kinyomtatásra nem kerül. A rose fogja átalakítani DVI-rõl PostScript® formátumra átalakítani (mivel a bamboo egy PostScript® nyomtató).

Nyomtatók hálózati adatcsatlakozással

Amikor hálózati kártyát vásárolunk a nyomtatónkhoz, általában két változatukkal találkozhatunk: az egyikük nyomtatási rendszerként mûködik (ez a drágább), a másikuk pedig egyszerûen csak soros vagy párhuzamos csatlakozón továbbítandó adatként közvetíti az adatokat a nyomtató felé (az olcsóbb). A drágábbik változatot az elõzõ, Távoli számítógépekre csatlakoztatott nyomtatók címû szakaszban leírtak szerint tudjuk használni.

Az /etc/printcap állományban ugyan meg tudjuk adni, hogy a nyomtató soros vagy párhuzamos portra csatlakozik, és azon keresztül milyen adatátviteli sebességgel (amennyiben soros), forgalomirányítással, tabulálással, sortörési konvenció szerint stb. kommunikáljunk vele. Azonban TCP/IP vagy más hálózati porton ülõ nyomtatók adatait itt nem tudjuk kifejteni.

A hálózatra kötött nyomtatók használatához lényegében egy olyan külön kifejlesztett kommunikációs programra van szükségünk, amely a szöveg- vagy konverziós szûrõkhöz hasonló módon hívható meg. Erre rögtön adunk is egy példát: a netprint szkript a szabványos bemenetrõl beolvassa az összes kinyomtatandó adatot és átküldi azokat a hálózatra csatlakoztatott nyomtatónak. A szkript elsõ paramétereként a nyomtató hálózati nevét adjuk meg, másodiknak pedig portot. Azonban megjegyezzünk, hogy ez csak egyirányú kommunikációt tesz lehetõvé (a FreeBSD-tõl a nyomtatóig). Sok hálózati nyomtató viszont két irányban is képes kommunikálni, ezért érdemes lehet ezt kihasználni (a nyomtató állapotának lekérdezésére, nyilvántartások készítésére stb).

```
#!/usr/bin/perl
#
#  netprint - A hálózatra csatlakoztatott nyomtató szövegszûrõje
#  Helye: /usr/local/libexec/netprint
#
$#ARGV eq 1 || die "Usage: $0 <printer-hostname> <port-number>";

$printer_host = $ARGV[0];
$printer_port = $ARGV[1];

require 'sys/socket.ph';

($ignore, $ignore, $protocol) = getprotobyname('tcp');
($ignore, $ignore, $ignore, $ignore, $address)
    = gethostbyname($printer_host);

$sockaddr = pack('S n a4 x8', &AF_INET, $printer_port, $address);

socket(PRINTER, &PF_INET, &SOCK_STREAM, $protocol)
    || die "Can't create TCP/IP stream socket: $!";
```

```
connect(PRINTER, $sockaddr) || die "Can't contact $printer_host: $!";
while (<STDIN>) { print PRINTER; }
exit 0;
```

Rengeteg szûrõben fel tudjuk használni ezt a szkriptet. Például tegyük fel, hogy egy Diablo 750-N típusú sornyomtatót csatlakoztattunk a hálózatra, amely az 5100-as porton várja a nyomtatandó adatokat. A hálózati neve most `scrivener` lesz. Íme a hozzá tartozó szövegszûrõ:

```
#!/bin/sh
#
#  diablo-if-net - Az 5100-as porton figyelõ `scrivener' nevû Diablo
#  nyomtató szövegszûrõje. Helye: /usr/local/libexec/diablo-if-net
#
exec /usr/libexec/lpr/lpf "$@" | /usr/local/libexec/netprint scrivener 5100
```

9.4.4. A nyomtató használatának szabályozása

Ebben a szakaszban a nyomtató használatának korlázásáról írunk. Az LPD rendszeren keresztül meghatározhatjuk, hogy ki képes helyben vagy távolról hozzáférni a nyomtatóhoz, mennyi másolatot nyomtathat, mennyi és egyenként mekkora nyomtatási feladatokat küldhet.

A másolatok számának szabályozása

Az LPD segítségével a felhasználók egy állományt könnyen ki tudnak nyomtatni akár többször is. Ha (például) a felhasználó egy nyomtatási feladat kiküldéséhez az `lpr -#5` parancsot használja, akkor a nyomtatási feladatban levõ összes állományból öt példányt kap. Ennek létjogosultságát azonban nekünk kell megítélni.

Amennyiben úgy érezzük, hogy a további példányok készítése csupán felesleges papír- és tintapazarlás, akkor az `sc` tulajdonság megadásával az /etc/printcap állományban kikapcsolhatjuk az lpr(1) `- lehetõség használatát. Így amikor a felhasználók a -` kapcsolóval küldenek el feladatokat a nyomtatóra, a következõt fogják tapasztalni:

```
lpr: multiple copies are not allowed
```

Fordítása:

```
lpr: másolatok nyomtatása nem engedélyezett
```

Vigyázzunk arra, hogy ha távoli számítógépen zajlik a nyomtatás (lásd Távoli számítógépekre csatlakoztatott nyomtatók), akkor az `sc` tulajdonságot a távoli számítógép /etc/printcap állományában is be kell állítani, máskülönben a felhasználók egy másik számítógéprõl mindig képesek lesznek több példány nyomtatására.

Nézzünk erre egy példát. Itt most a `rose` nevû számítógép /etc/printcap állományát vesszük szemügyre. Ebben a `rattan` egy nagyon szívélyes nyomtató lesz, ezért engedélyezi a másolatok

nyomtatását, azonban a `bamboo` nevû lézernyomtató nála már sokkal válogatósabb lesz, ezért a beállításai közt az `sc` tulajdonsággal kikapcsoljuk a másodpéldányok nyomtatását:

```
#
#  /etc/printcap (rose) - A másolatok korlátozása a "bamboo"
#  nevû nyomtatón
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:\
        :lp=/dev/lpt0:\
        :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:sc:\
        :lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:rw:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

Az `sc` tulajdonságot az `orchid`/etc/printcap állományában is meg kell adni (és ha már itt vagyunk, akkor tegyük meg ugyanezt a `teak` esetében is):

```
#
#  /etc/printcap (orchid) - Nincsenek másodpéldányok sem a helyi
#  "teak" nyomtatón, sem pedig a távoli "bamboo" nyomtatón
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
        :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:sc:\
        :if=/usr/local/libexec/ifhp:\
        :vf=/usr/local/libexec/vfhp:\
        :of=/usr/local/libexec/ofhp:

rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :lp=:rm=rose:rp=rattan:sd=/var/spool/lpd/rattan:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :lp=:rm=rose:rp=bamboo:sd=/var/spool/lpd/bamboo:sc:
```

Az `sc` tulajdonság használatával ugyan megakadályozzuk az `lpr -#` parancs teljesítését, azonban ez még mindig nem óv meg minket attól, hogy a felhasználók képesek legyenek többször egymás után lefuttatni az lpr(1) parancsot, vagy éppen egyetlen nyomtatási feladatban több állományt is elküldeni:

```
% lpr forsale.sign forsale.sign forsale.sign forsale.sign forsale.sign
```

Számos módszer kínálkozik az effajta visszaélések kivédésére (beleértve a figyelmen kívül hagyást is), lehet velük kísérletezgetni!

A nyomtatók hozzáférésének szabályozása

A UNIX® csoportkezelésével és az /etc/printcap állományban található `rg` tulajdonság felhasználásával korlátozni tudjuk, ki milyen nyomtatón dolgozhat. Ehhez mindössze annyit kell tennünk, hogy besoroljuk egy csoportba azokat a felhasználókat, amelyek hozzáférhetnek a nyomtatóhoz, és az `rg` tulajdonsággal megnevezzük azt.

A csoporton kívüli felhasználókat (köztük magát a `root` felhasználót is) pedig ezután így üdvözli a rendszer, ha megpróbálnak valamit kinyomtatni egy korlátozott felhasználású nyomtatón:

```
lpr: Not a member of the restricted group
```

Az üzenet fordítása:

```
lpr: Nem jogosult felhasználó
```

Ha erre a távoli számítógépek esetén szükségünk lenne (lásd Távoli számítógépekre csatlakoztatott nyomtatók), akkor tegyük ugyanazt, mint amit az `sc` (a másodpéldányok letiltása, "suppress multiple copies") tulajdonság esetén is, vagyis az `rg` tulajdonságot adjuk meg azokon a távoli számítógépeken is, amelyek hozzá tudnak férni a megosztott nyomtatóhoz.

Például megengedjük, hogy a `rattan` nevû nyomtatót bárki használhassa, azonban a `bamboo` nyomtatót csak az `artists` nevû csoport használhatja. Következzen hát akkor a `rose` korábbról már ismert /etc/printcap állománya:

```
#
#  /etc/printcap (rose) - A bamboo hozzáférésének korlátozása
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:\
        :lp=/dev/lpt0:\
        :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:\
        :lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:rw:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

Most ne bántsuk a másik (az `orchid` nevû gépen levõ) /etc/printcap állományt. Így persze az `orchid` bármelyik felhasználója nyomtathat a `bamboo` nyomtatón. De ez most egy olyan eset, ahol egyébként lekorlátozzuk a `orchid` elérését is, ezért az ott beengedett felhasználók már akár használhatják is a nyomtatót. Vagy sem.

Minden nyomtatóhoz csak egy ilyen csoportot adhatunk meg.

A beküldött nyomtatási feladatok méretének szabályozása

Ha sok felhasználó szeretne a nyomtatóinkhoz hozzáférni, akkor minden bizonnyal meg akarunk adni egy felsõ határt a felhasználók által beküldhetõ nyomtatások méretére vonatkozóan. Mivel a nyomtatási könyvtáraknak otthont adó állományrendszer is egyszer betelhet, ezért mindenképpen érdemes gondoskodni arról, hogy mindenki nyomtatási feladatát el tudjuk rendesen tárolni.

Az LPD az `mx` tulajdonsággal lehetõséget ad arra, hogy lekorlátozzuk a nyomtatási feladatokban található egyes állományok méretét. Ennek mértékegysége egy `BUFSIZ` blokk, ami pedig 1024 byte. Ha értékül nullát adunk meg, akkor nincs korlátozás, viszont ha semmit sem rögzítünk, akkor az `mx` tulajdonság alapértéke, vagyis 1000 blokk lesz a határ.

Ez az érték a nyomtatási feladatokban levõ *egyes állományok* méretére vonatkozik, *nem* pedig a nyomtatási feladatok teljes méretére.

Fontos tudni, hogy az LPD nem dobja vissza a méreten felüli állományokat. Ehelyett a méret alatti részt szépen berakja a sorba és kinyomtatja, a többi pedig elhagyja. Lehetne rajta vitázni, hogy ez mennyire helyes cselekedet.

Példaképpen definiáljunk a korábban használt `rattan` és `bamboo` nyomtatóinkhoz ilyen korlátokat. Mivel az `artists` csoport tagjai hajlamosak nagy PostScript® állományokat küldeni, ezért most lekorlátozzuk ezt öt megabyte-ra. A szöveges nyomtatónk esetén azonban nem lesz semmilyen határ:

```
#
#  /etc/printcap (rose)
#

#
#  Itt nincs korlát a nyomtatási feladatokra:
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:mx#0:sd=/var/spool/lpd/rattan:\
        :lp=/dev/lpt0:\
        :if=/usr/local/libexec/if-simple:

#
#  Öt megabyte a PostScript:
#
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:mx#5000:\
        :lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:rw:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

Ismét hozzátesszük, hogy ezek a korlátok csak a helyi felhasználókra vonatkoznak. Amennyiben távolról is el lehet érni ezt a nyomtatót, a távoli felhasználókat nem fog semmilyen korlátozás érinteni. Azokon a számítógépeken is meg kell adnunk az /etc/printcap állományban az `mx` tulajdonságot. Ehhez a Távoli számítógépekre csatlakoztatott nyomtatók címû szakaszban találunk

segítséget.

Van még egy speciális módszer, amivel képesek vagyunk szabályozni a távolról érkezõ kérések méretét. Errõl a Távoli számítógépekrõl érkezõ kérések szabályozása szakaszban olvashatunk.

Távoli számítógépekrõl érkezõ kérések szabályozása

Az LPD nyomtatási rendszer több módot is szolgáltat a távolról érkezõ nyomtatási feladatok szabályozására:

Az elérés szabályozása

Az /etc/hosts.equiv és /etc/hosts.lpd állományok segítségével beállíthatjuk, hogy mely távoli számítógépektõl fogadjon el kéréseket az LPD. Az LPD minden kérés elfogadásakor ellenõrzi, hogy a küldõ számítógép címe szerepel-e az említett állományok valamelyikében. Ha nem, akkor az LPD visszautasítja a kérést.

A két állomány felépítése egyszerû, mert bennük minden sorban egy-egy hálózati nevet adunk meg. Hozzátennénk azonban, hogy legyünk óvatosak, mivel az /etc/hosts.equiv állományt az ruserok(3) protokoll is használja, ezért ennek módosítása hatással van az rsh(1) és rcp(1) programok mûködésére.

Például most nézzük meg a rose/etc/hosts.lpd állományát:

```
orchid
violet
madrigal.fishbaum.de
```

Ennek megfelelõen tehát a rose elfogadja az orchid, violet és madrigal.fishbaum.de nevû távoli számítógépek kéréseit. Ha bármilyen más gép próbál hozzáférni a rose által felkínált LPD szolgáltatáshoz, visszautasítja.

A méret szabályozása

Szabályozhatjuk többek közt azt is, hogy mennyi szabad területnek kell fennmaradnia a nyomtatási könyvtárnak otthont adó állományrendszeren. A helyi nyomtató könyvtárában ehhez hozzunk létre egy minfree nevû állományt. Ide írjuk be, mennyi szabad lemezblokk (512 byte-os egység a lemezen) szükségeltetik egy távolról beérkezõ nyomtatási feladat fogadásához.

Így gondoskodhatunk róla, hogy a távoli felhasználók nem fogják eltömíteni az állományrendszerünket, illetve ezzel egyúttal adhatunk némi elõnyt a helyi felhasználóknak is: õk ugyanis még azután is képesek lesznek nyomtatási feladatokat küldeni a nyomtatónak, miután az állományrendszeren található szabad terület mennyisége már rég a minfree állományban szereplõ érték alá csökkent.

Példaként most a bamboo nevû nyomtatónkhoz adjunk meg egy ilyen minfree állományt. Ehhez az /etc/printcap állományból tudjuk kideríteni a hozzá tartozó nyomtatási könyvtárat. Lássuk tehát belõle a bamboo bejegyzését:

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
```

```
        :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:mx#5000:\
        :lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:rw:mx#5000:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

A nyomtatási könyvtárat az `sd` tulajdonság határozza meg. Úgy állítjuk most be, hogy az LPD számára a távoli nyomtatási feladatok fogadásához ebben a könyvtárban legalább három megabyte (6144 blokk) szabad területnek mindig lennie kell:

```
# echo 6144 > /var/spool/lpd/bamboo/minfree
```

A felhasználók szabályozása

Az /etc/printcap állományban megadható `rs` tulajdonság segítségével korlátozhatjuk a helyi nyomtatókhoz hozzáférni képes távoli felhasználókat. Amikor az `rs` tulajdonság szerepel egy helyben csatlakozó nyomtató leírásánál, akkor az LPD csak abban az esetben fogad el távoli felhasználóktól nyomtatási feladatot, *ha* az adott feladatot küldõ felhasználónak ugyanazon a néven van a helyi gépen is hozzáférése. Máskülönben az LPD vissza fogja utasítani a kérést.

Ez a tulajdonság különösen fontos olyan környezetben, ahol (például) több szervezeti egység használ egyetlen közös hálózatot és bizonyos felhasználók képesek átlépni szervezeti egységük határait, mivel ha a hozzáférést adunk nekik a rendszereinkhez, akkor képesek a saját helyükrõl használni ezeket. Ha ehelyett *csupán* a nyomtatóinkat és a számítógépünk összes erõforrását akarjuk megosztani, akkor létrehozhatunk a számukra olyan "token" hozzáféréseket is, amikhez nem tartozik sem felhasználói könyvtár, sem pedig parancsértelmezõ (pontosabban a /usr/bin/false).

9.4.5. A nyomtató használatának nyilvántartása

Tehát szükségünk lenne a nyomtatások költségének elszámolására. Miért is ne tennénk ilyet? A papír és a tinta bizony pénzbe kerül, amihez még hozzájárulnak más egyéb karbantartási költségek is - a nyomtatók dugig vannak mindenféle mozgó alkatrésszel, amelyek elõbb-utóbbi el is romlanak. Tegyük fel, hogy a nyomtatóink kapacitása, kihasználtsága és karbantartási költsége alapján már megállapítottunk egy elszámolási egységet (oldalanként, méterenként, akárminként). De hogyan lássunk hozzá a nyomtatások költségének tényleges nyilvántartásához?

Van egy rossz hírünk: az LPD nyomtatási rendszer önmaga nem tud segíteni ebben a feladatban. A nyilvántartás nagyban függ a használt nyomtatóktól, a nyomtatott formátumoktól és nyomtató *általunk* kiszabott költségeitõl.

A nyilvántartás létrehozásához át kell írnunk a nyomtatóhoz tartozó szûrõt (a nyers szövegek költségének felszámításához) és konverziós szûrõket (a különféle formátumok költségei miatt), amikkel aztán számolhatjuk vagy lekérdezhetjük a kinyomtatott lapokat. Egyetlen kimeneti szûrõ használatával szinte semmire se megyünk, mivel az nem képes nyilvántartás vezetésére. Errõl bõvebb útmutatást a Szûrõk szakaszban találhatunk.

Általánosságban véve két módon vezethetünk nyilvántartást:

- Az *idõszakos elszámolás* a gyakoribb, mivel ez az egyszerûbb. Amikor valaki végrehajt egy

nyomtatási feladatot, a szûrõ a nyilvántartást tároló állományba feljegyzi a felhasználó azonosítóját, a gépének nevét és a kinyomtatott oldalakat. Ezután minden hónapban, félévben, évben vagy akár tetszõleges idõközönként összegyûjtjük a nyomtatók nyilvántartásait és külön feljegyezzük az egyes felhasználók nyomtatásait, majd benyújtjuk róla a számlát. Töröljük az összes naplóállományt, és tiszta lappal kezdjük a következõ idõszakot.

- Az *azonnali elszámolás* már nem annyira népszerû, mivel nehezebb megvalósítani. Ekkor a felhasználók már közvetlenül a nyomtatás után megkapják a számlát, hasonlóan a lemezkvótákhoz. Meg tudjuk akadályozni ezzel azt is, hogy a felhasználók túlléphessék az elõre kiszabott "nyomtatási kvótájukat", amit persze menet közben lehet ellenõrizni és állítgatni. A felhasználók és kvótájuk nyomonkövetéséhez viszont szükségünk lesz egy kis adatbáziskezelésre is.

Az LPD nyomtatási rendszer mind a két módszer kivitelezéséhez tud segítséget nyújtani, hiszen amikor szûrõket állítunk be (vagyis szinte mindig), lehetõségünk van a nyilvántartást végzõ programrészleteket is beilleszteni. És ami feltétlenül elõnyös: óriási mértékû rugalmasságot ajánl fel a nyilvántartás megvalósításához. Például magunk választhatjuk ki, hogy idõszakos vagy azonnali elszámolást alkalmazunk. Meg tudjuk adni, milyen információkat rögzítsünk: felhasználói neveket, számítógépek neveit, a nyomtatási feladatok típusát, vagy a kinyomtatott oldalakat, a felhasznált lapok területét, a nyomtatások idõbeli igényeit és így tovább. Ehhez mindössze csak a szûrõket kell módosítani.

Nyilvántartás gyorsan és egyszerûen

A FreeBSD-ben egybõl találunk is két programot, amivel pillanatok alatt ki tudunk alakítani egy egyszerû idõszakos elszámolási rendszert. Ezek Az lpf szövegszûrõ címû szakaszban ismertetett `lpf` és a nyomtatók nyilvántartásait tartalmazó állományok adatainak összegyûjtését és kiértékelését végzõ pac(8).

Ahogy korábban már leírtuk a szûrõkrõl szóló szakaszban (Szûrõk), az LPD a szöveg- és konverziós szûrõket parancssorból a nyilvántartást tároló állomány nevével indítja el. Ezt a paramétert a szûrõk aztán fel tudják használni a nyilvántartások feljegyzéséhez. Az állomány nevét az /etc/printcap állományban szereplõ `af` tulajdonsággal tudjuk megadni, vagy teljes elérési úttal, vagy pedig a nyomtatási könyvtárhoz viszonyítva.

Az LPD az `lpf` szûrõt a lap szélességének és hosszának megadásával indítja el (ezeket az értékeket a `pw` és `pl` tulajdonságokból származtatja). Az `lpf` ezek felhasználásával meg tudja mondani, mennyi papírt használtunk el. Miután kiküldte az állományt a nyomtatóra, nyilvántartásba is veszi. Ezek a típusú bejegyzések valahogy így néznek ki:

```
2.00 rose:andy
3.00 rose:kelly
3.00 orchid:mary
5.00 orchid:mary
2.00 orchid:zhang
```

Minden nyomtatóhoz érdemes külön nyilvántartást vezetni, mivel az `lpf` nem tartalmaz semmilyen beépített zárolási megoldást, ezért két `lpf` párhuzamos futtatása könnyen összezagyválhatja a közösen használt nyilvántartások tartalmát. Az /etc/printcap állományban az `af=acct` tulajdonság

megadásával könnyen létre tudunk hozni minden nyomtatóhoz külön nyilvántartást. Ilyenkor minden nyomtató könyvtárában megjelenik egy acct nevû állomány.

Amikor elérkezünk a nyomtatások kiszámlázásához, futtassuk le a pac(8) programot. Ehhez mindössze annyit kell tennünk, hogy átlépünk az elszámolni kívánt nyomtató könyvtárába és begépeljük a `pac` parancsot. Ekkor kapunk egy ehhez hasonló, dollár alapú kimutatást:

```
  Login               pages/feet   runs    price
orchid:kelly               5.00    1   $  0.10
orchid:mary               31.00    3   $  0.62
orchid:zhang               9.00    1   $  0.18
rose:andy                  2.00    1   $  0.04
rose:kelly               177.00  104   $  3.54
rose:mary                 87.00   32   $  1.74
rose:root                 26.00   12   $  0.52

total                    337.00  154   $  6.74
```

A pac(8) a következõ paramétereket várja:

`-P nyomtató`

Az kiértékelendõ *nyomtató* neve. Ez a paraméter csak akkor használható, ha az /etc/printcap állományban az `af` tulajdonságnak teljes elérési utat adtunk meg.

`-c`

A felhasználók nevei helyett a fizetendõ összeg szerint rendezze a listát.

`-m`

Hagyja figyelmen kívül a nyilvántartásban szereplõ gépek hálózati neveit. Ennek hatására az `alpha` géprõl nyomtató `smith` meg fog egyezni a `gamma` géprõl nyomtatóval. A beállítás nélkül ez a két felhasználó el fog térni.

`-p ár`

A paraméterként megadott *ár* dollár értékkel számol oldalanként vagy lábanként az /etc/printcap állományban megadott `pc` tulajdonság értéke helyett (ami alapból két cent). Az *ár* lebegõpontos (valós) számként is megadható.

`-r`

A rendezési sorrend megfordítása.

`-s`

Hozzon létre egy elszámolást, majd törölje a hozzá kapcsolódó nyilvántartási adatokat.

név...

Csak az adott *nevû* felhasználók adatait értékelje ki.

A pac(8) által alapértelmezés szerint generált kimutatásban láthatjuk az egyes gépekrõl származó egyes felhasználók kinyomtatott oldalait. Ha nekünk viszont nem számít, hogy honnan küldték a

kéréseket (mivel bárhonnan lehet küldeni), akkor a `pac -m` paranccsal az alábbi táblázatot készíttethetjük el:

```
  Login               pages/feet   runs    price
andy                        2.00     1   $  0.04
kelly                     182.00   105   $  3.64
mary                      118.00    35   $  2.36
root                       26.00    12   $  0.52
zhang                       9.00     1   $  0.18

total                     337.00   154   $  6.74
```

Itt megtaláljuk a ténylegesen kifizetendõ összegeket is, amik kiszámításához a pac(8) az /etc/printcap állomány `pc` tulajdonságát használja (ez alapból 200, avagy 2 cent oldalanként). Ezzel a tulajdonsággal tehát egy cent századrészében mérve tudjuk megadni az oldalakénti vagy lábankénti árakat. Ezt a beállítást természetesen a pac(8) `-p` opciójával felül tudjuk bírálni. Arra azonban vigyázzunk, hogy a `-p` után dollárban kell megadnunk az árat. Emiatt tehát a

```
# pac -p1.50
```

parancs szerint minden egyes oldal másfél dollárba fog kerülni. Ezzel az opcióval aztán alaposan megdönthetjük az árakat.

Végezetül megemlítjük, hogy a `pac -s` parancs az általa létrehozott elszámolást egy külön állományba menti, amelynek a neve nagyjából megegyezik a nyilvántartást végzõével, de `_sum`-ra (mint "summary", azaz elszámolás) végzõdik. Ezután nullázza a nyilvántartást. Amikor a pac(8) programot újra lefuttatjuk, újból beolvassa a korábban elmentett elszámolásokat, majd hozzászámolja a többit a hagyományos nyilvántartási adatokból.

Hogyan tudjuk számolni a kinyomtatott lapokat?

A nyilvántartás pontos vezetéséhez még távolról is valamilyen módon meg kell tudnunk mondani, hogy mennyi lapot használt egy nyomtatási feladat végrehajtása. Ez a nyomtatás nyilvántartásának egyik alapvetõ problémája.

A nyers szövegek esetében ez nem is annyira bonyolult: egyszerûen számoljuk össze, hogy a nyomtatási feladatban mennyi sor kinyomtatására lesz szükség és vessük össze ezt a nyomtató által lapoként kinyomtatott sorok számával. Ne felejtsük el számításba venni a szövegben felbukkanó törlések hatását, vagy az olyan hosszú sorokat, amelyek a valóságban több sorban fognak megjelenni.

Viszont (Az lpf szövegszûrõ címû szakaszban bemutatott) `lpf` program ezeket mind lekezeli a nyilvántartások készítése során. Ezért ha szintén egy nyilvántartást vezetni képes szövegszûrõt akarunk írni, akkor mindenképpen érdemes megnéznünk az `lpf` forráskódját.

De hogyan bánjunk el a többi formátummal?

Nos, a DVI-Laserjet és DVI-PostScript® közti átalakítások esetén a kinyomtatott lapok számának

megállapításához meg kell tanítanunk a szûrõnket értelmezni a `dvilj` vagy `dvips` parancsok kimenetét. Ugyanezt meg tudjuk tenni más formátumok és más konverziós programok használata során is.

Azonban ezek a módszerek nem veszik számításba, hogy a nyomtató egyáltalán kinyomtatta-e az összes elküldött oldalt. Sok minden történhet még addig, például beragadhat a papír, kifogyhat a tinta vagy akár felrobbanhat a nyomtató - a felhasználónak ettõl függetlenül még fizetnie kell.

Mit lehet ilyenkor tenni?

A *precíz* nyilvántartásnak csak egyetlen *biztos* módja létezik. Olyan nyomtatót szerezzünk be, amely képes megmondani, mennyi lapot használt el a nyomtatás során, majd egy ilyet csatlakoztassunk soros porton vagy hálózaton keresztül. Szinte majdnem az összes PostScript® nyomtató támogatja ezt a lehetõséget, ahogy sok más gyártmány és típus is (például a hálózati Imagen lézernyomtatók). A nyomtatóhoz tartozó szûrõt ehhez úgy kell módosítani, hogy lekérdezzük a kinyomtatott lapok számát a nyomtatás után és *kizárólag* erre az értékre alapozva készítünk nyilvántartást. Itt nincs szükség sem a sorok számolására, sem pedig az állományok (könnyen elhibázható) átvizsgálására.

Természetesen lehetünk nagylelkûek és ne számítsunk fel semmit a nyomtatásért.

9.5. A nyomtatók használata

Ebbõl a szakaszból megtudhatjuk, hogyan használjuk a FreeBSD-n beállított nyomtatónkat. Röviden most itt foglaljuk össze az ide tartozó felhasználói parancsokat:

lpr(1)
: Nyomtatási feladatokat hajt végre.

lpq(1)
: Ellenõrzi a nyomtatási sorokat.

lprm(1)
: Feladatokat vesz ki a nyomtatási sorokból.

Ezek mellett létezik még a nyomtatók és a hozzájuk tartozó sorok irányítására alkalmas parancs is, az lpc(8), amelyre a A nyomtatók vezérlése címû szakaszban fogunk részleteiben kitérni.

A nyomtatók/sorok /etc/printcap állományban szereplõ nevük szerinti megadásához az lpr(1), lprm(1) és lpq(1) parancsok közül mindegyik elfogadja a `-P nyomtatónév` paramétert. Ennek köszönhetõen képesek vagyunk nyomtatási feladatokat küldeni, eltávolítani vagy felügyelni az egyes nyomtatók soraiban. Ha nem használjuk a `-P` kapcsolót, akkor az érintett nyomtató a `PRINTER` környezeti változó által meghatározott lesz. Végül, ha a `PRINTER` nevû környezeti változót sem állítottuk be, akkor a parancsok alapértelmezett módon az `lp` nevû nyomtatót fogják használni.

A továbbiakban az *alapértelmezett nyomtató* kifejezés a `PRINTER` környezeti változó által megnevezett nyomtatóra fog utalni, illetve ha ezt nem definiáltuk, akkor az `lp` nevû nyomtatóra.

9.5.1. Nyomtatási feladatok végrehajtása

Az állományok kinyomtatásához írjuk be:

```
% lpr állománynév ...
```

Ezzel kinyomtatjuk az összes felsorolt állományt az alapértelmezett nyomtatón. Ha nem adunk meg állományokat, akkor az lpr(1) parancs a szabványos bemenetrõl várja a nyomtatandó adatokat. Például ezzel a paranccsal néhány igen fontos rendszerállományt tudunk kinyomtatni:

```
% lpr /etc/host.conf /etc/hosts.equiv
```

A nyomtató megválasztásához így adjuk ki a parancsot:

```
% lpr -P nyomtatónév állománynév ...
```

Ez a példa kinyomtatja az aktuális könyvtár részletes listáját a `rattan` nevû nyomtatón:

```
% ls -l | lpr -P rattan
```

Mivel egyetlen állományt sem adtunk meg az lpr(1) programnak, az `lpr` parancs a nyomtatandó adatokat a szabványos bemenetrõl várja, ami jelen esetünkben a `ls -l` parancs kimenete.

Az lpr(1) ezeken felül még képes értelmezni rengeteg formázásra, konverzióra, másolatok készítésére stb. utasító kapcsolót is. Errõl bõvebben a Nyomtatási beállítások címû szakaszban lesz szó.

9.5.2. Nyomtatási feladatok felügyelete

Amikor az lpr(1) programmal nyomtatunk, az összes nyomtatandónk egy "nyomtatási feladatnak" nevezett csomagba kerül, ami pedig az LPD nyomtatási rendszerébe. Minden nyomtatóhoz tartozik egy nyomtatási sor, ahol részünkrõl és mások által eddig kiadott nyomtatási feladatokat találhatjuk. A nyomtató ezután ezeket érkezési sorrend szerint dolgozza fel.

Az alapértelmezett nyomtatóhoz tartozó sor állapotát az lpq(1) programmal tudjuk megnézni. Ha egy adott nyomtatóra vagyunk kíváncsiak, akkor használjuk a `-P` kapcsolót. Például a

```
% lpq -P bamboo
```

parancs a `bamboo` nevû nyomtató sorát fogja megmutatni. Példaképpen lássuk is ilyen esetben az `lpq` parancs eredményét:

```
bamboo is ready and printing
Rank   Owner    Job  Files                                 Total Size
```

```
active kelly    9    /etc/host.conf, /etc/hosts.equiv   88 bytes
2nd    kelly    10   (standard input)                   1635 bytes
3rd    mary     11   ...                                78519 bytes
```

Itt három nyomtatási feladatot láthatunk a `bamboo` nyomtatási sorában. Az elsõ nyomtatási feladat, amit a `kelly` nevû felhasználó küldött, a 9-es "feladatszámot" kapta. A nyomtatóhoz tartozó összes feladat kap egy ilyen egyedi számot. Többnyire nyugodtan figyelmen kívül hagyhatjuk, azonban szükségünk lehet rá, ha éppen törölni kívánjuk a hozzá tartozó nyomtatási feladatot. Ezzel majd a Nyomtatási feladatok eltávolítása címû szakaszban foglalkozunk.

A kilences számú nyomtatási feladat két állományt tartalmaz: ha a parancssorban több állományt adunk meg az lpr(1) programnak, akkor az egy nyomtatási feladatnak számít. Ez egyben a pillanatnyilag aktív nyomtatási feladat (ezt a "Rank" oszlopban szereplõ `active` érték jelzi), tehát a nyomtató éppen ezzel foglalatoskodik. A második nyomtatási feladat közvetlenül az lpr(1) szabványos bemenetére érkezett. A harmadik a `mary` nevû felhasználótól jött, és ez egy nagyobb méretû nyomtatási feladat. A nyomtatandó állomány elérési útvonala túlságosan hosszú ahhoz, hogy ki lehessen írni, ezért az lpr(1) csak három pontot jelez ki helyette.

Az lpq(1) kimenetének elsõ sorai is nagyon hasznos információt tartalmaz: megtudhatjuk, mit csinál éppen (legalább is az LPD szerint) a nyomtató.

A `-l` kapcsolóval az lpq(1) parancstól kérhetünk sokkal részletesebb listázást is. Például így nézhet ki a `lpq -l` parancs eredménye:

```
waiting for bamboo to become ready (offline ?)
kelly: 1st              [job 009rose]
       /etc/host.conf                   73 bytes
       /etc/hosts.equiv                 15 bytes

kelly: 2nd              [job 010rose]
       (standard input)                 1635 bytes

mary: 3rd                               [job 011rose]
      /home/orchid/mary/research/venus/alpha-regio/mapping 78519 bytes
```

9.5.3. Nyomtatási feladatok eltávolítása

Ha meggondoltuk volna magunkat egy nyomtatási feladattal kapcsolatban, az lprm(1) paranccsal még törölni tudjuk a sorból. Az lprm(1) gyakran még a folyamatban levõ nyomtatási feladatot is képes eltávolítani, azonban elõfordulhat, hogy a nyomtatási feladat egy része már elvégzésre került.

Az alapértelmezett nyomtató sorából csak úgy tudunk nyomtatási feladatokat törölni, ha elõször az lpq(1) segítségével megkeressük a számukat. Ha ez megvan, írjuk be:

```
% lprm feladatám
```

Adott nyomtatóról a -P kapcsoló segítségével tudunk nyomtatási feladatot törölni. A most következõ parancs a bamboo nevû nyomtatóról törli a 10-es számú nyomtatási feladatot:

```
% lprm -P bamboo 10
```

Az lprm(1) parancs esetén még használhatóak az alábbi rövidítések is:

lprm -
: Eltávolítja a hozzánk tartozó az összes nyomtatási feladatot (az alapértelmezett nyomtatón).

lprm *felhasználó*
: Eltávolítja az adott *felhasználó*hoz tartozó összes nyomtatási feladatot (az alapértelmezett nyomtatón). Kizárólag a rendszergazdák képesek erre, a rendes felhasználók csak a saját nyomtatási feladataikat törölhetik.

lprm
: A nyomtatási feladat száma, a felhasználói név vagy a -megadása nélkül az lprm(1) törli az alapértelmezett nyomtatón éppen aktív nyomtatási feladatot, amennyiben az a miénk. Csak a rendszergazdák képesek bármilyen aktív nyomtatási feladatot törölni.

Ha kiegészítjük az imént említett rövidítéseket a -P paraméter megadásával, akkor az alapértelmezett nyomtató helyett bármelyik másikat is használhatjuk. Például ez a parancs eltávolítja az aktuális felhasználó összes nyomtatási feladatot a rattan nevû nyomtatón:

```
% lprm -P rattan -
```

Hálózati környezetben az lprm(1) csak arról a géprõl engedi törölni a nyomtatási feladatokat, amelyrõl küldték ezeket, még abban az esetben is, amikor ugyanaz a nyomtató más számítógépekrõl is elérhetõ. A következõ parancssorozat ezt igyekszik szemléltetni:

```
% lpr -P rattan myfile
% rlogin orchid
% lpq -P rattan
Rank   Owner      Job  Files                          Total Size
active seeyan     12    ...                           49123 bytes
2nd    kelly      13   myfile                         12 bytes
% lprm -P rattan 13
rose: Permission denied
% logout
% lprm -P rattan 13
dfA013rose dequeued
cfA013rose dequeued
```

9.5.4. Túl a nyers szövegen: nyomtatási beállítások

Az lpr(1) parancs számos olyan beállítást enged, amelyekkel a szövegek formázását, grafikák átalakítását illetve más állományformátumok használatát, másolatok készítését, nyomtatási feladatok irányítását és még sok minden mást el tudunk végezni. Ebben a szakaszban pontosan ezekrõl a kapcsolókról lesz szó.

Formázási és konverziós beállítások

Az lpr(1) most következõ opciói a nyomtatási feladatokban található állományok formázását vezérlik. Akkor használjuk ezeket a beállításokat, ha a nyomtatási feladat nem tartalmaz nyers szöveget, vagy ha nyers szöveget akarunk formázni a pr(1) segédprogrammal.

Például az alábbi parancs kinyomtat egy halászati-jelentés.dvi nevû (a TeX betûszedû rendszerbõl már jól ismert) DVI állományt a `bamboo` nevû nyomtatón:

```
% lpr -P bamboo -d halászati-jelentés.dvi
```

Ezek a beállítások a nyomtatási feladatban szereplõ minden egyes állományra vonatkoznak, ezért nem keverhetjük (például) a DVI és ditroff formátumú állományokat egy nyomtatási feladaton belül. Ehelyett külön nyomtatási feladatokban kell elküldenünk az eltérõ formátumú állományokat, és mindegyik nyomtatási feladathoz külön konverziós beállításokat kell megadnunk.

A `-p` és `-T` kapcsolók kivételével az itt felsorolt összes beállításnak a kiválasztott nyomtatóhoz szüksége van a megfelelõ konverziós szûrõre. Például a `-d` opció használatához kell egy konverziós szûrõ a DVI formátumhoz. A Konverziós szûrõk címû szakasz errõl ad bõvebb tájékoztatást.

`-c`

Cifplot állományok nyomtatása.

`-d`

DVI állományok nyomtatása.

`-f`

FORTRAN forrás nyomtatása.

`-g`

Plot formátumú adatok nyomtatása.

`-i szám`

A kinyomtatott szöveg behúzásának növelése a *szám* értékével. Ha nem adjuk meg a *számot*, akkor ennek értéke 8 lesz. Ez a beállítás csak bizonyos konverziós szûrõkkel mûködik.

Ne hagyjunk helyet az `-i` és a szám között.

`-l`

A szöveg formázás nélküli nyomtatása, vezérlõkarakterekkel együtt.

`-n`

Ditroff (eszközfüggetlen troff) adat nyomtatása.

-p

Nyomtatás elõtt a szöveg formázása a pr(1) programmal. Lásd pr(1).

`-T cím`

Az állomány neve helyett a fejlécben a *címet* jeleníti meg a pr(1). Ennek a beállításnak csak a `-p` opcióval együtt van hatása.

`-t`

Troff adat nyomtatása.

`-v`

Raszteres adatok nyomtatása.

Vegyünk az iméntiekre egy példát. A következõ parancs az ls(1) szépen megformázott man oldalát nyomtatja ki az alapértelmezett nyomtatón:

```
% zcat /usr/shared/man/man1/ls.1.gz | troff -t -man | lpr -t
```

A zcat(1) kitömöríti az ls(1) man oldalának forrását és átadja a troff(1) parancsnak, ami ebbõl létrehoz a GNU troff formátumának megfelelõ kimenetet és továbbadja az lpr(1) parancsnak, ami végül elküldi a nyomtatási feladatot az LPD nyomtatási rendszernek. Mivel az lpr(1) parancsnak megadtuk az `-t` kapcsolót, a nyomtatási rendszer a GNU troff formátumban érkezõ adatokat magától át fogja alakítani olyan formátumra, amit a nyomtató is képes lesz megérteni.

Nyomtatási feladatok kezelése

Az lpr(1) most felsorolandó beállításaival az LPD rendszert arra tudjuk utasítani, hogy a nyomtatási feladatot különleges módon kezelje:

-# *példányszám*

Egyetlen példány helyett hozzon létre *példányszám* számú példányt a nyomtatási feladatban található összes állományból. A rendszergazda a nyomtató kímélése érdekében ezt a lehetõséget letilthatja, amivel inkább a fénymásoló használatára ösztönzi a felhasználókat. Lásd A másolatok számának szabályozása szakasz.

A beállítás illusztrálásaként most az alapértelmezett nyomtatón elõször nyomtassunk ki három példányt a parser.c, majd ezután a parser.h állományokból:

```
% lpr -#3 parser.c parser.h
```

-m

A rendszer küldjön levelet a nyomtatási feladat teljesítése után. Ekkor az LPD a nyomtatási feladat elvégzése után levelet küld a helyi postafiókunkba. A levélben kifejti, hogy sikeres volt-e a nyomtatás, vagy esetleg valamilyen hiba keletkezett, és ha hiba történt, akkor pontosan mi is volt az.

-s

Ne másolja közvetlenül az állományokat a nyomtatási könyvtárba, hanem készítsen hozzájuk szimbolikus linkeket.

Egy nagyobb nyomtatási feladat elvégzése esetén javasolt használni ezt a kapcsolót. Ezzel a megoldással helyet tudunk spórolni a nyomtatási könyvtárban (amikor a nyomtatási feladatok könnyen megtelítheti a nyomtatási könyvtárat tároló állományrendszert). Emellett idõt is takarítunk meg, mivel az LPD-nek nem kell a nyomtatási feladat minden egyes bitjét átmásolni a nyomtatási könyvtárba.

Van azonban egy hátránya: mivel az LPD ekkor közvetlenül az eredeti állományra fog hivatkozni, ezért a nyomtatás befejezéséig azt nem módosíthatjuk vagy törölhetjük.

Ha egy távoli nyomtatónak küldjük a nyomtatási feladatot, akkor az LPD a helyi és a távoli számítógép között mégis kénytelen lesz átmásolni a nyomtatási feladatot, így a `-s` kapcsoló egyedül csak a helyi nyomtatási könyvtárban fog helyet spórolni. Ettõl eltekintve még ilyenkor is hasznunkra válhat.

-r

Törölje a nyomtatási feladatban szereplõ állományokat, miután átmásolta ezeket a nyomtatási könyvtárba, vagy miután a `-s` kapcsoló használatával kinyomtatta ezeket. Nagy körültekintéssel használjuk!

A fejléclapok beállításai

Az lpr(1) most következõ beállításai a nyomtatási feladatok fejlécében megjelenõ szövegekre vannak hatással. Így ha letiltottuk a fejléclapok használatát, akkor ezek a kapcsolók lényegében semmit sem állítanak. A Fejléclapok címû szakaszból tudhatunk meg többet ezek beállításáról.

-C *szöveg*

A fejléclapon megjelenõ hálózati név helyett a *szöveg* fog szerepelni. A hálózati név általában annak a gépnek a neve, ahonnan a nyomtatási feladatot küldték.

-J *szöveg*

A fejléclapon megjelenõ nyomtatási feladat neve helyett a *szöveg* fog megjelenni. A nyomtatási feladat neve általában a benne szereplõ elsõ állomány nevével egyezik meg, ha a szabványos bemenetrõl nyomtatunk, akkor egyszerûen csak stdin.

-h

Ne nyomtasson fejléclapot.

Bizonyos helyeken elõfordulhat, hogy ennek a kapcsolónak nincs semmilyen

hatása a fejléclapok létrehozásának módszerébõl fakadóan. A részleteket lásd a Fejléclapok szakaszban.

9.5.5. A nyomtatók vezérlése

A nyomtatóink rendszergazdájaként nekünk kell telepítenük, üzembe helyeznünk és kipróbálnunk ezeket. Az lpc(8) parancs használatával még jobban képesek vagyunk kapcsolatba lépni velük. Az lpc(8) paranccsal:

- el tudjuk indítani és le tudjuk állítani a nyomtatókat;
- be- és ki tudjuk kapcsolni a nyomtatási soraikat;
- át tudjuk rendezni az egyes sorokban található nyomtatási feladatokat.

Elõször is essen pár szó a fogalmakról: ha a nyomtató *leállt*, akkor semmit sem fog kinyomtatni a sorából. A felhasználók továbbra is képesek nyomtatási feladatokat küldeni, amik azonban egészen addig fognak várakozni, amíg a nyomtatót *el nem indítjuk* vagy a sorát ki nem ürítjük.

Ha egy sort *kikapcsolunk*, akkor (a `root` kivételével) egyetlen felhasználó sem képes nyomtatási feladatokat küldeni a nyomtatónak. A *bekapcsolt* sorok képesek csak nyomtatási feladatot fogadni. A nyomtató *elindítható* kikapcsolt sorral is, ilyenkor egészen addig folytatja a nyomtatási feladatok elvégzését, amíg a sor ki nem ürül.

Általánosan elmondható, hogy az lpc(8) parancs használatához a `root` felhasználó jogosultságaira van szükségünk. Az lpc(8) parancsot minden más esetben csak a nyomtató állapotának ellenõrzésére vagy a megakadt nyomtató újraindítására használhatjuk.

Foglaljuk röviden össze az lpc(8) parancsait. A legtöbb parancs kiadásához még szükséges egy *nyomtatónév* paraméter megadása is, amivel megnevezzük az utasítani kívánt nyomtatót. Helyette használható az `all` szó is, amivel az /etc/printcap állományban szereplõ összes nyomtatót egyszerre utasíthatjuk.

`abort nyomtatónév`

Az aktuális nyomtatási feladat megszakítása és a nyomtató leállítása. Ha a nyomtatási sort még nem kapcsoltuk ki, a felhasználók küldhetnek további nyomtatási feladatokat.

`clean nyomtatónév`

A nyomtató könyvtárából töröljük a régi állományokat. Esetenként adódhat, hogy bizonyos nyomtatási feladatok állományait nem takarította el az LPD, különösen abban az esetben, amikor a nyomtatás vagy az adminisztrálás során keletkezett valamilyen hiba. Ez a parancs segít megtalálni a nyomtatási könyvtárból már kikopott állományokat és törli ezeket.

`disable nyomtatónév`

Az újonnan érkezõ nyomtatási feladatok besorolásának kikapcsolása. Ha a nyomtató még mûködik, akkor folytatni fogja a sorban még bennmaradt nyomtatási feladatok elvégzését. A rendszergazda (a `root`) még a kikapcsolt sorok esetén is küldhet nyomtatási feladatokat.

Ez a parancs valójában akkor hasznos, ha egy új nyomtató vagy egy új szûrõ mûködését próbálgatjuk: ilyenkor érdemes kikapcsolni a nyomtatási sort és `root` felhasználóként

nyomtatási feladatokat küldeni. A többi felhasználó a tesztelés befejezéséig nem tud majd nyomtatási feladatokat küldeni, vagyis egészen addig, amíg a nyomtatási sort vissza nem kapcsoljuk az `enable` paranccsal.

`down nyomtatónév üzenet`

A nyomtató üzemen kívül helyezése. Lényegében megegyezik egy `disable` és utána egy `stop` parancs kiadásával. Az *üzenet* akkor jelenik meg, amikor a valaki megpróbálja lekérdezni a nyomtató állapotát az `lpc status` paranccsal, vagy amikor megnézi a nyomtatási sorát az lpq(1) paranccsal.

`enable nyomtatónév`

A nyomtatóhoz tartozó nyomtatási sor bekapcsolása. A felhasználók ezután már képesek lesznek a nyomtatónak feladatokat küldeni, azonban egészen addig nem nyomtatódik ki semmi, amíg a nyomtatót el nem indítjuk.

`help parancsnév`

Megmutatja a *parancsnév* parancshoz tartozó súgót. A *parancsnév* megadása nélkül a rendelkezésre álló parancsok listáját kapjuk meg.

`restart nyomtatónév`

Elindítja a nyomtatót. A felhasználók ezt a parancsot tudják használni abban az esetben, amikor valamilyen megmagyarázhatatlan okból az LPD mûködése megáll, viszont ezzel nem tudják elindítani a `stop` vagy `down` parancsokkal leállított nyomtatót. A `restart` parancs megegyezik az `abort` és a `start` egymás utáni kiadásával.

`start nyomtatónév`

Elindítja a nyomtatót, és a nyomtató nekilát kinyomtatni a sorában levõ nyomtatási feladatokat.

`stop nyomtatónév`

Leállítja a nyomtatót, és a nyomtató az aktuális nyomtatási feladat befejezése után már nem kezd neki újabbnak. Ettõl függetlenül a felhasználók még továbbra is képesek feladatokat küldeni a nyomtatási sorába.

`topq nyomtatónév feladat-vagy-felhasználónév`

Átrendezi a *nyomtatónév* nevû nyomtató sorát úgy, hogy a megadott azonosítójú *feladatot* vagy a megadott *felhasználónévhez* tartozó nyomtatási feladatokat a sor elejére teszi. Ennél a parancsnál *nyomtatónévnek* nem adhatjuk meg az `all` értéket.

`up nyomtatónév`

Üzembe helyezi a nyomtatót, tulajdonképpen a `down` parancs ellentéte. Megegyezik egy egymás után kiadott `start` és `enable` paranccsal.

Az lpc(8) a fenti parancsokat a parancssorból fogadja el. Ha itt nem adunk meg neki semmilyen parancsot, akkor az lpc(8) interaktív módba vált, ahol ugyanezeket a parancsokat adhatjuk ki, egészen az `exit`, `quit` parancsok vagy az állományvége jelzés begépeléséig.

9.6. Más nyomtatási rendszerek

Ha derekasan végigolvastuk eddig ezt a fejezetet, akkor mostanra már valószínûleg mindent tudunk a FreeBSD-ben található LPD nyomtatási rendszerrõl. Ezzel együtt tisztában vagyunk a hiányosságaival is, aminek kapcsán természetes módon felmerülhet bennünk a kérdés: "Milyen más (FreeBSD-vel is mûködni képes) nyomtatási rendszerek léteznek még?"

LPRng

Az LPRng, aminek jelentése "LPR Next Generation" (Az LPR következõ generációja), a PLP teljesen újraírt változata. Patrick Powell és Justin Mason (a PLP eredeti karbantartója) együttes munkájának gyümölcse az LPRng. Az LPRng honlapja: http://www.lprng.org/.

CUPS

A CUPS, vagy más néven a "Common UNIX Printing System" (Közös UNIX®-os nyomtatási rendszer), egy hordozható nyomtatási réteget nyújt a UNIX®-alapú operációs rendszerek számára. Az Easy Software Products fejlesztése és szinte az összes UNIX® gyártó és felhasználó szemében elfogadott szabványos nyomtatási rendszer.

A CUPS a nyomtatási feladatok és sorok kezelését az internetes nyomtatási protokollon (Internet Printing Protocol, IPP) használatával oldja meg. Csökkentett képességekkel ugyan, de a sornyomtató démon (Line Printer Daemon, LPD), szerverüzenet-blokk (Server Message Block, SMB), és AppSocket (más néven JetDirect) protokollokat is ismeri. A CUPS a komolyabb UNIX®-os nyomtatási feladatokhoz ezeken felül még a hálózati nyomtatók közti választást és PostScript nyomtatók leírásán (PostScript Printer Description, PPD) alapuló nyomtatási beállításokat is támogatja.

A CUPS honlapja: http://www.cups.org/.

HPLIP

A HPLIP, másnéven HP Linux® Imaging and Printing, egy HP által kidolgozott programcsalád, amely támogatja a HP eszközök nyomtatási, lapolvasási és faxolási lehetõségeit. A benne található programok bizonyos nyomtatási feladatokhoz backendként a CUPS nyomtatási rendszert használják.

A HPLIP honlapja a http://hplipopensource.com/hplip-web/index.html címen érhetõ el.

9.7. Hibakeresés

Miután az lptest(1) programmal elvégeztünk néhány egyszerû próbát, a várt helyett a következõk egyikét kaphatjuk eredményül:

Egy kis idõ után minden remekül mûködött, vagy nem dobta ki az egész lapot.

A nyomtató nyomtatott egy keveset, aztán egy ideig csendben maradt és nem csinált semmit. Ilyenkor a nyomtatnivalók megjelenéséhez minden bizonnyal meg kell nyomnunk a nyomtatón levõ "PRINT REMAINING" vagy "FORM FEED" feliratú gombokat.

Ebben az esetben a nyomtató valószínûleg még arra várt, hogy még a nyomtatás megkezdése elõtt érkezik valamilyen további adat. Ettõl a gondtól úgy szabadulhatunk meg, ha beállítunk egy

szövegszûrõt, amely minden (szükséges) esetben küld egy "FORM FEED" (lapdobás) jelzést is a nyomtatónak. Ez kell általában ahhoz, hogy a szövegnek a nyomtató belsõ pufferében megmaradt része azonnal kinyomtatódjon. Akkor is a javunkra válhat ez, ha minden egyes nyomtatási feladatot külön lapon akarunk kezdeni, mivel így a következõ nyomtatási feladat sosem közvetlenül ott kezdõdik, ahol az elõzõ feladat befejezte a nyomtatást.

A /usr/local/libexec/if-simple szûrõ helyett a következõ szkript használhatával tudunk minden nyomtatási feladat elvégzése után elküldeni egy lapdobást:

```
#!/bin/sh
#
# if-simple - Egyszerû lpd szövegszûrõ
# Helye: /usr/local/libexec/if-simple
#
# Egyszerûen átmásolja a szabvány bemenetet a szabvány kimenetre, és
# figyelmen kívül hagyja az összes többi paramétert.  Minden nyomtatási
# nyomtatási feladat elvégzése után küld egy lapdobást (\f).

/bin/cat && printf "\f" && exit 0
exit 2
```

"Lépcsõsen" jelentek meg a sorok.

Ekkor a következõt látjuk a lapon:

```
!"#$%&'()*+,-./01234
              "#$%&'()*+,-./012345
                           #$%&'()*+,-./0123456
```

Az ún. *lépcsõhatás* áldozatává váltunk, amelyet a sortörést jelzõ karakter eltérõ értelmezései okoznak. A UNIX® stílusú operációs rendszerek erre mindössze egyetlen karaktert használnak: ez a 10-es kódú ASCII karakter (sordobás, Line Feed, LF). Az MS-DOS®, OS/2® és mások pedig két karakterrel oldják meg ezt a feladatot: a 10-es *és* 13-as kódú (kocsivissza, Carriage Return, CR) ASCII karakterekkel. A sortöréseknél sok nyomtató az MS-DOS® szokásait követi.

Amikor a FreeBSD-vel nyomtatunk, akkor csak egyetlen karaktert használunk sortörésre. Ennek láttán a nyomtató lépteti a sort, azonban a fej vízszintes pozícióját nem változtatja meg a következõ sor nyomtatásának megkezdésekor. Erre lenne a kocsivissza karakter, vagyis ennek hatására fogja a nyomtató a papír bal oldalára visszaállítani a következõ nyomtatandó karakter pozícióját.

A FreeBSD így szeretné utasítani a nyomtatót:

A nyomtató kocsivisszát kap	A nyomtató visszalépteti a pozíciót
A nyomtató sordobást kap	A nyomtató új sort kezd

Néhány módszer ennek kiváltására:

- A nyomtatón található kapcsolók vagy vezérlõpanel segítségével próbáljuk meg átállítani a vezérlõkarakterek nyomtató szerinti értelmezését. Keressük meg a nyomtató kézikönyvében, hogyan tudjuk ezt megcsinálni.

Ha a FreeBSD mellett más operációs rendszerekkel is használni akarjuk a nyomtatót, akkor azok indítása elõtt mindig *át kell állítani* a nyomtatót a megfelelõ értelmezés alkalmazására. Ilyenkor valószínûleg a lentebb szereplõ megoldásokat részesítjük majd inkább elõnyben.

- Állítsuk be úgy a FreeBSD soros vonali meghajtóját, hogy magától alakítsa át az LF karaktereket CR+LF párokká. Természetesen ez a megoldás *csak* a soros portra csatlakozó nyomtatók esetében mûködhet. Ehhez az /etc/printcap állományban a nyomtató leírásánál az `ms#` tulajdonságnál adjuk meg az `onlcr` módot.
- Küldjünk olyan *kódot* a nyomtatónak, amelynek hatására ideiglenesen máshogy fogja kezelni az LF karaktereket. Nézzük meg a nyomtatóhoz mellékelt útmutatóban, hogy milyen kódokat tudunk ilyen célra használni. Ha találtunk ilyen kódot, akkor írjuk át úgy a hozzá tartozó szövegszûrõt, hogy a nyomtatási feladatok elõtt mindig elküldjük azt.

  Most bemutatjuk egy olyan szövegszûrõ kódját, amely a Hewlett-Packard PCL kódjait ismerõ nyomtatókhoz készült. Ebben a szûrõben elõször kiadjuk, hogy az LF karaktereket LF és CR karakterek kombinációjának tekintse a nyomtató, majd elküldjük magát a nyomtatási feladatot, és a nyomtatási feladat eredményének utolsó lapja után elküldünk egy lapdobást. Szinte az összes Hewlett Packard nyomtatóval mûködnie kell.

```
#!/bin/sh
#
# hpif - Egyszerû lpd bemeneti szûrõ a HP-PCL alapú nyomtatókhoz
# Helye: /usr/local/libexec/hpif
#
# Egyszerûen átmásolja a szabvány kimenetet a szabvány bemenetre, és
# figyelmen kívül hagyja a paramétereket. Elküldi a nyomtatónak, hogy
# az LF karaktereket CR+LF-ként kezelje, majd a feladat befejeztével
# lapot dobat.

printf "\033&k2G" && cat && printf "\033&l0H" && exit 0
exit 2
```

  Példaként megadjuk még az `orchid` nevû számítógép /etc/printcap állományát is. Ebben egyetlen nyomtató csatlakozik a párhuzamos portra, amelynek a típusa LaserJet 3Si és a neve `teak`. Az elõbb bemutatott szövegszûrõt használja:

```
#
#  /etc/printcap (orchid)
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
        :lp=/dev/lpt0:sh:sd=/var/spool/lpd/teak:mx#0:\
        :if=/usr/local/libexec/hpif:
```

Egymásra írja a sorokat.

A nyomtató nem lépteti a sorokat, ezért az összes sor egymáson jelenik meg.

Ez pontosan a ritka "ellentéte" a fentebb leírt lépcsõhatásnak. A FreeBSD által sortörésre használt LF karakterek valamiért CR karakterekként viselkednek, ezért a nyomtató nem sort vált, hanem a lap bal szélére állítja a fejet.

A nyomtatón található kapcsolókkal vagy vezérlõpanellel így állítsuk be a sordobás és kocsivissza karakterek értelmezését:

Amit a nyomtató kap	Arra a nyomtató nyomtat
CR	CR
LF	CR + LF

A nyomtató elhagy karaktereket.

Miközben nyomtatunk, a nyomtató bizonyos karaktereket nem hajlandó megjeleníteni. A probléma ennél nagyobb, ha a nyomtató mûködése közben egyre több és több karaktert hagy ki.

Itt az a gond, hogy a nyomtató nem képes tartani az iramot a számítógép által a soros vonalon átküldött adatok sebességével (ez a probléma nem jelentkezhet a párhuzamos nyomtatók esetén). Két módon kerekedhetünk felül ezen:

- Ha a nyomtató ismeri a XON/XOFF típusú forgalomirányítást, akkor az `ms#` tulajdonságnál adjuk meg a FreeBSD számára az `ixon` beállítást.
- Ha a nyomtató ismeri a "Request to Send / Clear to Send" alapú hardveres kézfogást (más néven `RTS/CTS` forgalomirányítást), akkor az `ms#` tulajdonságnál a `crtscts` beállítást adjuk meg. Gondoskodjunk róla, hogy a számítógépet és a nyomtatót összekötõ kábel meg tudjon majd birkózni ezzel a típusú forgalomirányítással.

Mindenféle szemetet nyomtat.

A nyomtató nem a nyomtatni kívánt szöveget hozza létre, hanem összevissza nyomtat.

Ez a soros nyomtatók helytelen kommunikációs beállításának egy másik jellemzõ tünete. Ellenõrizzük a `br` tulajdonságnál megadott adatátviteli sebességet és az `ms#` tulajdonságnál megadott paritási beállításokat. Egyeztessük a nyomtató saját és az /etc/printcap állományban tárolt beállításait.

Semmi sem történik.

Ha semmi sem történt, akkor a gond magával a FreeBSD-vel lehet, nem pedig a hardverrel. Az /etc/printcap állományba a vizsgálni kívánt nyomtató leírásához (az `lf` tulajdonsággal) illesszünk be naplózást. Például így fog kinézni a `rattan` nevû nyomtató bejegyzése az `lf` tulajdonság megadásával kibõvítve:

```
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:\
        :lp=/dev/lpt0:\
        :if=/usr/local/libexec/if-simple:\
```

```
        :lf=/var/log/rattan.log
```

Miután ezt megcsináltuk, próbálkozzunk újra. Nézzük meg a naplóállományban (ami a példánkban a /var/log/rattan.log nevén érhetõ el), hogy látunk-e valamilyen hibaüzenetet. Az itt tapasztalt hibaüzenetek nyomán elindulva igyekezzünk megszüntetni a probléma forrását.

Ha nem adjuk meg az `lf` tulajdonságot, akkor az LPD erre a célra alapértelmezés szerint a /dev/console állományt használja.

Chapter 10. Bináris Linux kompatibilitás

10.1. Áttekintés

A FreeBSD számos más UNIX®-szerû operációs rendszerhez nyújt bináris kompatibilitást, köztük a Linuxhoz is. Elcsodálkozhatnánk rajta, hogy vajon miért kell tudnia a FreeBSD-nek Linux binárisokat futtatnia. A válasz erre nagyon egyszerû. Rengeteg cég és fejlesztõ kizárólag csak Linuxra fejleszt, hiszen ez mostanság egy nagyon "izgalmas téma" az informatika világában. Emiatt azonban a FreeBSD közösségnek külön gyõzködnie kell ezeket a cégeket és fejlesztõket, hogy készítsék el a termékeik natív FreeBSD-s változatát. Ezzel az a gond, a legtöbb ilyen cég egyszerûen nem veszi észre, hogy ha létezne a terméküknek FreeBSD-re írt változata, akkor még többen használnák. Így továbbra is csak Linuxra fejlesztenek. Mit tudnak tenni ilyenkor a FreeBSD használói? Nos, ekkor jön jól a FreeBSD bináris szintû kompatibilitása.

Dióhéjban úgy tudnánk összefoglalni, hogy ennek köszönhetõen a FreeBSD felhasználók képesek a linuxos alkalmazások közel 90%-át mindenféle további módosítás nélkül futtatni. Így tehát használható a StarOffice™, getenv(3) Linux változata, az Adobe® Acrobat®, RealPlayer®, VMware, Oracle®, WordPerfect®, Doom, Quake, és még sok minden más. Sõt, egyes tapasztalatok szerint bizonyos helyzetekben a FreeBSD által futtatott Linux binárisok sokkal jobban teljesítenek, mint Linux alatt.

Azonban vannak olyan Linuxra jellemzõ, az operációs rendszer szintjén meghúzódó eszközök, amelyek FreeBSD alatt nem használhatóak. FreeBSD-n nem fognak mûködni azok a Linux binárisok, amelyek túlzottan kihasználják az olyan i386™-os rendszerhívásokat, mint például a virtuális 8086 mód.

A fejezet elolvasása során megismerjük:

- hogyan engedélyezzük rendszerünkön a Linux kompatibilitást;
- hogyan telepítsünk linuxos osztott könyvtárakat;
- hogyan telepítsünk linuxos alkalmazásokat a FreeBSD rendszerünkre;
- a FreeBSD Linux kompatibilitásának implementációs részleteit.

A fejezet elolvasásához ajánlott:

- külsõ szoftverek telepítésének ismerete (Alkalmazások telepítése. csomagok és portok).

10.2. Telepítés

A bináris Linux kompatibilitás alapértelmezés szerint nem engedélyezett. Legkönnyebben úgy tudjuk elérhetõvé tenni, ha betöltjük a `linux` nevû KLD modult ("Kernel LoaDable"). Ehhez `root` felhasználóként a következõket kell begépelni:

```
# kldload linux
```

Ha minden egyes rendszerindítás során engedélyezni szeretnénk a bináris kompatibilitást, akkor

tegyük bele az /etc/rc.conf állományba ezt a sort:

```
linux_enable="YES"
```

A modul betöltõdését a kldstat(8) paranccsal tudjuk ellenõrizni:

```
% kldstat
Id Refs Address    Size     Name
 1    2 0xc0100000 16bdb8   kernel
 7    1 0xc24db000 d000     linux.ko
```

Ha valamiért nem akarjuk vagy nem éppen nem tudjuk betölteni a modult, akkor a bináris Linux kompatibilitást az `options COMPAT_LINUX` beállítással be is tudjuk építeni a rendszermagba. Ennek pontos menetét a A FreeBSD rendszermag testreszabásaben találjuk meg.

10.2.1. Linuxos futtatókönyvtárak telepítése

A linuxos könyvtárakat két módon is felrakhatjuk: egyrészt a linux_base port telepítésével, másrészt manuálisan.

A könyvtárak telepítése a linux_base porttal

A futtatókönyvtárakat a lehetõ legegyszerûbben a emulators/linux_base porton keresztül tudjuk telepíteni. Teljesen úgy történik, mint a Portgyûjtemény akármelyik másik portjának telepítése. Csupán ennyit kell beírnunk:

```
# cd /usr/ports/emulators/linux_base-f10
# make install distclean
```

A FreeBSD 8.0 kiadását megelõzõ változataiban az emulators/linux_base-f10 port helyett az emulators/linux_base-fc4 portot használjuk.

A telepítés végeztével kaptunk is egy mûködõ bináris Linux kompatibilitást, habár egyes programok még panaszkodhatnak a rendszerkönyvtárak alverzióit illetõen. Általánosságban véve ez azonban nem okoz nagyobb gondot.

A emulators/linux_base portnak több változata is használható, melyek az egyes Linux disztribúcióknak feleltethetõek meg. Ilyenkor mindig érdemes közülük azt választani, amelyik a leginkább megfelel a telepíteni kívánt linuxos alkalmazás igényeinek.

A könyvtárak telepítése manuálisan

Ha korábban még nem telepítettük volna a Portgyûjteményt, akkor egyénileg kell felraknunk az egyes könyvtárakat. Közülük azokra lesz szükségünk, amelyeket maga az alkalmazás is használni akar, valamint a futásidejû linkerre. Emellett még a FreeBSD rendszerünkön levõ Linux binárisok

számára a /compat/linux könyvtárban létre kell hoznunk a gyökér ún. "árnyékkönyvtárát" is. A FreeBSD alatt elindított Linux programok elõször ebben a könyvtárban fogják keresni a hozzájuk tartozó osztott könyvtárakat. Így tehát, amikor egy linuxos program betölti például a /lib/libc.so függvénykönyvtárat, akkor a FreeBSD elõször a /compat/linux/lib/libc.so állományt próbálja meg megnyitni, majd ha az nem létezik, akkor a /lib/libc.so állományt. Az osztott könyvtárak ezért a /compat/linux/lib árnyékkönyvtárba telepítendõek, és nem oda, ahova a linuxos `ld.so` mutat.

Általánosságban szólva eleinte elég csak azokat az osztott könyvtárakat megkeresni és felrakni, amelyekre a telepítendõ linuxos alkalmazásunknak ténylegesen szüksége van. Egy idõ után úgyis összegyûlnek azok a fontosabb függvénykönyvtárak, amelyek segítségével már minden további ráfordítás nélkül futtatni tudjuk a frissen importált programokat.

Hogyan telepítsünk újabb osztott könyvtárakat?

Mit tegyünk, ha az emulators/linux_base port telepítése után az alkalmazás még mindig hiányol néhány osztott könyvtárat? Honnan tudhatjuk meg, hogy milyen osztott könyvtárak kellenek majd egy Linux bináris használatához, és honnan szerezzük be ezeket? Erre alapvetõn két lehetõségünk van (az utasításokat `root` felhasználóként kell majd végrehajtanunk).

Ha hozzáférünk egy Linux rendszerhez, akkor szedjük össze az alkalmazásunk futtatásához szükséges osztott könyvtárakat, és másoljuk ezeket a FreeBSD partíciójára. Például:

Tegyük fel, hogy FTP-n keresztül leszedtük a Doom Linux változatát, és felraktuk egy általunk elérhetõ Linux rendszerre. Az `ldd linuxdoom` parancs segítségével ki tudjuk deríteni, milyen osztott könyvtárak kellenek majd nekünk:

```
% ldd linuxdoom
libXt.so.3 (DLL Jump 3.1) => /usr/X11/lib/libXt.so.3.1.0
libX11.so.3 (DLL Jump 3.1) => /usr/X11/lib/libX11.so.3.1.0
libc.so.4 (DLL Jump 4.5pl26) => /lib/libc.so.4.6.29
```

Az utolsó oszlopban levõ állományokat másoljuk át, tegyük ezeket a /compat/linux könyvtárba, és hozzunk létre az elsõ oszlopban szereplõ szimbolikus linkeket. Így tehát a következõ állományok kellenének:

```
/compat/linux/usr/X11/lib/libXt.so.3.1.0
/compat/linux/usr/X11/lib/libXt.so.3 -> libXt.so.3.1.0
/compat/linux/usr/X11/lib/libX11.so.3.1.0
/compat/linux/usr/X11/lib/libX11.so.3 -> libX11.so.3.1.0
/compat/linux/lib/libc.so.4.6.29
/compat/linux/lib/libc.so.4 -> libc.so.4.6.29
```

Ha már rendelkezünk az `ldd` kimenetének elsõ oszlopában szereplõ fõverziószámú osztott könyvtárral, akkor nem kell átmásolni az utolsó oszlopban levõ állományokat, hiszen így is mûködnie kellene mindennek. Ha viszont egy újabb változattal találkozunk, akkor érdemes mégis inkább átmásolni. Miután a szimbolikus linkeket átirányítottuk az új változatra, a régit akár törölhetjük is. Ha

például ezek a könyvtárak elérhetõek a rendszerünkön:

```
/compat/linux/lib/libc.so.4.6.27
/compat/linux/lib/libc.so.4 -> libc.so.4.6.27
```

Észrevesszük, hogy az `ldd` kimenetében az új bináris egy újabb változatot igényel:

```
libc.so.4 (DLL Jump 4.5pl26) -> libc.so.4.6.29
```

Ha csak az utolsó jegyében marad le valamivel a verziószám, akkor nem kell különösebben aggódnunk a /lib/libc.so.4.6.29 miatt sem, hiszen a programnak egy picivel korábbi verzióval is remekül kellene tudnia mûködni. Természetesen, ha akarjuk, ettõl függetlenül lecserélhetjük a libc.so állományt, ami ezt eredményezi:

```
/compat/linux/lib/libc.so.4.6.29
/compat/linux/lib/libc.so.4 -> libc.so.4.6.29
```

A szimbolikus linkek karbantartása *csak* a Linux binárisok esetén szükséges. A FreeBSD saját futásidejû linkere magától megkeresi a megfelelõ fõverziószámú könyvtárakat, ezért emiatt általában nem kell aggódni.

10.2.2. Linux ELF binárisok telepítése

Az ELF binárisok futtatása elõtt néha még szükség van a "megbélyegzés" (branding) használatára is. Ha egy bélyegezetlen ELF binárist akarunk elindítani, akkor a következõ hibaüzenetet kapjuk:

```
% ./egy-linux-elf-bináris
ELF binary type not known
Abort
```

A FreeBSD rendszermagjának a brandelf(1) paranccsal tudunk segíteni a FreeBSD és a Linux binárisainak megkülönböztetésében.

```
% brandelf -t Linux egy-linux-elf-bináris
```

A GNU által fejlesztett eszközök manapság már automatikusan elhelyezik az ELF binárisok azonosításához szükséges bélyegeket, ezért ez a lépés a jövõben egyre inkább feleslegessé válik.

10.2.3. Tetszõleges RPM formátumú csomag telepítése

A FreeBSD a telepített (akár linuxos) alkalmazások nyomonkövetésére saját csomagadatbázissal rendelkezik, amelynek következtében a Linux® által felkínált RPM adatbázisokat nem támogatja.

Ennek ellenére akármelyik RPM alapú Linux® alkalmazás telepíthetõ rendszerünkre a következõ módon:

```
# cd /compat/linux
# rpm2cpio -q < /a/linuxos/allomány.helye.rpm | cpio -id
```

Ezt követõen a brandelf(1) segítségével állítsuk be az ELF binárisokat (könyvtárakat viszont ne!) megfelelõ típusúra. Ekkor ugyan nem leszünk képesek rendesen eltávolítani az így telepített szoftvert, de ez a módszer teszteléshez megfelelõ.

10.2.4. A névfeloldó beállítása

Ha a névfeloldás (DNS) valamiért nem mûködne, vagy egy ehhez hasonló üzenetet kapunk:

```
resolv+: "bind" is an invalid keyword resolv+:
"hosts" is an invalid keyword
```

Akkor a /compat/linux/etc/host.conf állományba be kell illesztenünk a következõ sorokat:

```
order hosts, bind
multi on
```

Az itt megszabott sorrend szerint elõször az /etc/hosts állományt nézi át, és majd csak ezután próbálja meg feloldani a nevet. Ha a /compat/linux/etc/host.conf állomány nem létezik, akkor a linuxos alkalmazás a FreeBSD /etc/host.conf állományát találja meg, és panaszkodni fog a FreeBSD eltérõ formátumára. Távolítsuk el a `bind` szócskát, ha nem állítottunk be névszervert az /etc/resolv.conf állományhoz.

10.3. A Mathematica® telepítése

Ebben a szakaszban megismerhetjük, hogyan telepítsük a Mathematica® 5.X Linux változatát FreeBSD rendszerekre.

A Mathematica® vagy a Mathematica® for Students linuxos változatai közvetlenül megrendelhetõek a fejlesztõtõl: http://www.wolfram.com/.

10.3.1. A Mathematica® telepítõjének elindítása

Elõször is jeleznünk kell a FreeBSD-nek, hogy a Mathematica® binárisai a linuxos ABI-t (Application Binary Interface) fogják használni. Itt legkönnyebben úgy járhatunk el, ha egyszerûen beállítjuk, hogy a rendszer a bélyegezetlen ELF binárisokat automatikusan Linux binárisoknak tekintse:

```
# sysctl kern.fallback_elf_brand=3
```

Ennek köszönhetõen a FreeBSD most már az összes bélyegezetlen ELF bináris esetén a linuxos ABI-t fogja használni, és így a telepítõt akár már közvetlenül a CD-rõl is indíthatjuk.

Most másoljuk át a MathInstaller nevû állományt a merevlemezünkre:

```
# mount /cdrom
# cp /cdrom/Unix/Installers/Linux/MathInstaller helyi_könyvtár
```

Az állományban cseréljük ki az elsõ sorban található `/bin/sh` hivatkozást a `/compat/linux/bin/sh` hivatkozásra. Ezzel biztosíthatjuk, hogy a telepítõt a linuxos sh(1) fogja elindítani. Ezután a kedvenc szövegszerkesztõnkkel vagy a következõ szakaszban található szkript segítségével helyettesítsük benne a `Linux)` szöveg összes elõfordulását a `FreeBSD)` szöveggel. Mivel a Mathematica® telepítõje az `uname -s` parancsra kapott válaszból állapítja meg az operációs rendszer típusát, ezért ezzel a módosítással a FreeBSD-t is a Linuxhoz hasonló módon fogja kezelni. A `MathInstaller` elindítása után most már telepíthetõ a Mathematica®.

10.3.2. A Mathematica® állományainak módosítása

A Mathematica® telepítése során létrejött szkripteket a használatuk elõtt át kell írnunk. Amennyiben a Mathematica®hoz tartozó programokat a /usr/local/bin könyvtárba telepítettük, akkor itt találjuk a math, mathematica, Mathematica és MathKernel állományokra mutató szimbolikus linkeket. Ezek mindegyikében cseréljük ki a `Linux)` karakterláncot a `FreeBSD)` szövegre a kedvenc szövegszerkesztõnkkel vagy az alábbi szkripttel:

```
#!/bin/sh
cd /usr/local/bin
for i in math mathematica Mathematica MathKernel
  do sed 's/Linux)/FreeBSD)/g' $i > $i.tmp
  sed 's/\/bin\/sh/\/compat\/linux\/bin\/sh/g' $i.tmp > $i
  rm $i.tmp
  chmod a+x $i
done
```

10.3.3. A Mathematica® jelszavának megszerzése

A Mathematica® elsõ indítása során kérni fog egy jelszót. Ha még nem kértünk volna jelszót a fejlesztõtõl, akkor a "számítógépünk azonosítójának" (machine ID) megállapításához indítsuk el a telepítés könyvtárában található `mathinfo` nevû programot. Ez az azonosító lényegében az elsõdleges Ethernet kártyánk MAC-címe lesz, ezért a Mathematica® nem futtatható több számítógépen.

Amikor e-mailen, telefonon vagy faxon keresztül regisztráljuk a terméket a Wolframnál, akkor meg kell adnunk nekik ezt az azonosítót "machine ID" néven, amire õk elküldik a hozzá tartozó jelszót.

10.3.4. A Mathematica® frontendjének futtatása hálózaton keresztül

A Mathematica® a szabványos betûkészletekkel meg nem jeleníthetõ szimbólumokhoz

(integráljelek, szummák, görög betûk, matematikai jelölések stb.) használ néhány olyan speciális betûtípust, amelyek nem minden esetben állnak rendelkezésre. Az X által használt protokoll miatt ezeket a betûtípusokat *helyben* kell telepíteni. Ennek értelmében a Mathematica® CD-jén található betûtípusokat telepítenünk kell a számítógépünkre is. A CD-n ezeket általában a /cdrom/Unix/Files/SystemFiles/Fonts könyvtárban találjuk meg, vagy a merevlemezen a /usr/local/mathematica/SystemFiles/Fonts könyvtárban. Ezen belül pedig a Type1 és X alkönyvtárakra van szükségünk. Az alábbiakban leírtak szerint több módon is használhatjuk ezeket.

Az egyik ilyen módszer, ha átmásoljuk az imént említett könyvtárakat a többi mellé, vagyis a /usr/X11R6/lib/X11/fonts könyvtárba. Ekkor szükségünk lesz még a fonts.dir állomány átírására is, ahova fel kell vennünk a betûtípusok neveit, majd ennek megfelelõen az elsõ sorban módosítanunk a könyvtárban található betûtípusok számát. De ugyanígy lefuttathatjuk ebben a könyvtárban a mkfontdir(1) parancsot is.

Az a másik megoldás, ha a könyvtárakat így másoljuk át a /usr/X11R6/lib/X11/fonts helyre:

```
# cd /usr/X11R6/lib/X11/fonts
# mkdir X
# mkdir MathType1
# cd /cdrom/Unix/Files/SystemFiles/Fonts
# cp X/* /usr/X11R6/lib/X11/fonts/X
# cp Type1/* /usr/X11R6/lib/X11/fonts/MathType1
# cd /usr/X11R6/lib/X11/fonts/X
# mkfontdir
# cd ../MathType1
# mkfontdir
```

Most adjuk hozzá az új könyvtárakat a betûtípusok könyvtáraihoz:

```
# xset fp+ /usr/X11R6/lib/X11/fonts/X
# xset fp+ /usr/X11R6/lib/X11/fonts/MathType1
# xset fp rehash
```

Ha az Xorg szervert használjuk, akkor az xorg.conf állományban megadhatjuk ezen könyvtárak automatikus betöltését is.

Az XFree86™ típusú szerverek esetén az XF86Config konfigurációs állományt kell módosítanunk.

Ha még *nincs* /usr/X11R6/lib/X11/fonts/Type1 nevû könyvtárunk, akkor a példában szereplõ MathType1 könyvtárat nyugodtan átnevezhetjük Type1 nevûre.

10.4. A Maple™ telepítése

A Maple™ egy Mathematica®hoz hasonló kereskedelmi alkalmazás. A használatához elõször meg kell vásárolni a http://www.maplesoft.com/ címrõl, majd a licenc megszerzéséhez ugyanott

regisztrálni. FreeBSD-re a szoftvert a következõ egyszerû lépéseken keresztül tudjuk telepíteni.

1. Indítsuk el a termékhez mellékelt INSTALL nevû szkriptet. Válasszuk a telepítõprogram által felkínált opciók közül a "RedHat" címkéjût. A telepítés célkönyvtára legyen a /usr/local/maple.
2. Ha eddig még nem tettük volna meg, rendeljük meg a Maple™ licencét a Maple Waterloo Software-tõl (http://register.maplesoft.com/) és másoljuk az /usr/local/maple/license/license.dat állományba.
3. Az Maple™-höz mellékelt INSTALL_LIC szkript elindításával telepítsük a FLEXlm licenckezelõt. A szervernek adjuk meg a számítógépünk hálózati nevét.
4. Javítsuk át a /usr/local/maple/bin/maple.system.type állományt a következõ módon:

```
   ----- itt kezdõdik a módosítás ---------
*** maple.system.type.orig      Sun Jul  8 16:35:33 2001
--- maple.system.type   Sun Jul  8 16:35:51 2001
***************
*** 72,77 ****
--- 72,78 ----
          # the IBM RS/6000 AIX case
          MAPLE_BIN="bin.IBM_RISC_UNIX"
          ;;
+     "FreeBSD"|\
      "Linux")
          # the Linux/x86 case
        # We have two Linux implementations, one for Red Hat and
   ----- módosítás vége -------------------
```

   Vigyázzunk, hogy a `"FreeBSD"|\` kezdetû sor végén nem szabad semmilyen további whitespace karakternek lennie.

   Ez a javítás arra utasítja a Maple™-t, hogy a "FreeBSD"-t Linux rendszerként ismerje fel. A bin/maple szkript hívja a bin/maple.system.type szkriptet, amely pedig a `uname -a` hívással próbálja kideríteni az operációs rendszer nevét. Ettõl függõen választja ki, hogy milyen típusú binárisokat fog futtatni.

5. Indítsuk el a licenckezelõ szervert.

   A most következõ szkripttel könnyedén el tudjuk indítani az `lmgrd` programot. A szkriptet /usr/local/etc/rc.d/lmgrd.sh néven hozzuk létre:

```
   ----- nyissz -----------

#! /bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bi
n
PATH=${PATH}:/usr/local/maple/bin:/usr/local/maple/FLEXlm/UNIX/LINUX
export PATH
```

```
LICENSE_FILE=/usr/local/maple/license/license.dat
LOG=/var/log/lmgrd.log

case "$1" in
start)
    lmgrd -c ${LICENSE_FILE} 2>> ${LOG} 1>&2
    echo -n " lmgrd"
    ;;
stop)
    lmgrd -c ${LICENSE_FILE} -x lmdown 2>> ${LOG} 1>&2
    ;;
*)
    echo "Usage: `basename $0` {start|stop}" 1>&2
    exit 64
    ;;
esac

exit 0
   ----- nyissz -----------
```

6. Próbáljuk meg elindítani a Maple™-t:

```
% cd /usr/local/maple/bin
% ./xmaple
```

Szerencsés esetben innentõl kezdve már minden mûködik. És ne felejtsünk el írni a Maplesoftnak, hogy szeretnénk egy natív FreeBSD verziót a termékükbõl!

10.4.1. Általános buktatók

- A FLEXlm licenckezelõvel esetenként nehéz lehet elboldogulni. Errõl a témáról bõvebben a http://www.globetrotter.com/ címen találunk leírásokat.
- Az `lmgrd` nagyon válogatós a licencállományokat illetõen és bármilyen apróságra kiakad. Egy szabályos licencállomány valahogy így néz ki:

```
# ====================================================
# License File for UNIX Installations ("Pointer File")
# ====================================================
SERVER chillig ANY
#USE_SERVER
VENDOR maplelmg

FEATURE Maple maplelmg 2000.0831 permanent 1 XXXXXXXXXXXX \
         PLATFORMS=i86_r ISSUER="Waterloo Maple Inc." \
         ISSUED=11-may-2000 NOTICE=" Technische Universitat Wien" \
```

```
        SN=XXXXXXXXX
```

A sorozatszámot természetesen eltávolítottuk. Itt a `chillig` a számítógép neve.

Az itt megadott licencállomány remekül használható egészen addig a pontig, amíg békén hagyjuk a "FEATURE" kezdetû sort (melyet a licenckulcs véd).

10.5. A MATLAB® telepítése

Ez a leírás azt mutatja be, hogyan telepítsük FreeBSD rendszerekre a MATLAB® version 6.5 Linux változatát. A Java Virtual Machine™ (lásd A Java™ futtató környezet élesítése) használatától eltekintve meglepõen jól mûködik.

A MATLAB® Linux változata közvetlenül megrendelhetõ a The MathWorks-tõl, a http://www.mathworks.com címen. Ne felejtsük el beszerezni a licencállományt és az elkészítéséhez szükséges útmutatót. Ha már úgyis arra járunk, jelezzük a fejlesztõknek, hogy igényt tartanánk a termékük natív FreeBSD-s változatára is!

10.5.1. A MATLAB® telepítése

A MATLAB® telepítéséhez a következõket kell tennünk:

1. Helyezzük be a telepítõ CD-t és csatlakoztassuk. A telepítõszkript javaslatának megfelelõen váltsunk át a `root` felhasználóra. A szóbanforgó szkript elindításához gépeljük be a következõt:

   ```
   # /compat/linux/bin/sh /cdrom/install
   ```

   A telepítõ grafikus. Ha a megjelenítõ használatáról szóló hibaüzeneteket kapunk, akkor adjuk ki a `setenv HOME ~FELHASZNÁLÓ` parancsot, ahol a *FELHASZNÁLÓ* annak a felhasználónak a neve legyen, amivel az imént meghívtuk a su(1) programot.

2. Amikor a MATLAB® könyvtárát kell megadnunk, ezt írjuk be: `/compat/linux/usr/local/matlab`.

   A telepítés további részeinek megkönnyítése érdekében írjuk be ezt a parancssorba: `set MATLAB=/compat/linux/usr/local/matlab`

3. Miután megkaptuk a MATLAB® licencét, az útmutatás szerint szerkesszük át.

   A licencállományt a kedvenc szövegszerkesztõnkkel akár már korábban elõ is készíthetjük, és majd amikor a telepítõnek szüksége lesz rá, másoljuk be $MATLAB/license.dat helyre.

4. Futtassuk le a telepítést.

Ezzel befejezõdött a MATLAB® hagyományos telepítése. Innentõl már csak a FreeBSD rendszer "hozzátapasztásán" fogunk dolgozni.

10.5.2. A licenckezelõ elindítása

1. Hozzunk létre szimbolikus linkeket a licenckezelõ szkriptjeire:

```
# ln -s $MATLAB/etc/lmboot /usr/local/etc/lmboot_TMW
# ln -s $MATLAB/etc/lmdown /usr/local/etc/lmdown_TMW
```

2. Hozzunk létre egy indítószkriptet /usr/local/etc/rc.d/flexlm.sh néven. A lentebb látható minta a MATLAB®hoz mellékelt $MATLAB/etc/rc.lm.glnx86 állomány egy módosított változata. Benne az állományok helyét és a licenckezelõ indításának körülményeit változtattuk meg (hogy Linux emuláció alatt fusson).

```
#!/bin/sh
case "$1" in
  start)
        if [ -f /usr/local/etc/lmboot_TMW ]; then
              /compat/linux/bin/sh /usr/local/etc/lmboot_TMW -u felhasználó &&
echo 'MATLAB_lmgrd'
        fi
        ;;
  stop)
    if [ -f /usr/local/etc/lmdown_TMW ]; then
            /compat/linux/bin/sh /usr/local/etc/lmdown_TMW  > /dev/null 2>&1
    fi
        ;;
  *)
    echo "Usage: $0 {start|stop}"
    exit 1
    ;;
esac

exit 0
```

Tegyük ezt az állományt végrehajthatóvá:

```
# chmod +x /usr/local/etc/rc.d/flexlm.sh
```

A fenti szkriptben cseréljük ki a *felhasználó* nevét a rendszerünkben levõ egyik felhasználó nevére (ami persze nem a `root`).

3. A licenckezelõt az alábbi paranccsal indítsuk el:

```
# /usr/local/etc/rc.d/flexlm.sh start
```

10.5.3. A Java™ futtató környezet élesítése

A Java™ futtató környezet (Java™ Runtime Environment, JRE) linkjét irányítsuk át egy FreeBSD alatt mûködõ változatéra:

```
# cd $MATLAB/sys/java/jre/glnx86/
# unlink jre; ln -s ./jre1.1.8 ./jre
```

10.5.4. A MATLAB® indítószkriptjének elkészítése

1. Hozzunk létre egy ilyen indítószkriptet a /usr/local/bin/matlab könyvtárban:

```
#!/bin/sh
/compat/linux/bin/sh /compat/linux/usr/local/matlab/bin/matlab "$@"
```

2. Futtassuk le a `chmod +x /usr/local/bin/matlab` parancsot.

A szkript lefutása során az emulators/linux_base verziójától függõen hibákat is kaphatunk. Ha el akarjuk kerülni ezeket, akkor szerkesszük át a /compat/linux/usr/local/matlab/bin/matlab állomány következõ sorát:

```
if [ `expr "$lscmd" : '.*->.*'` -ne 0 ]; then
```

(a 13.0.1 számú verzióban ez 410. sor) erre:

```
if test -L $newbase; then
```

10.5.5. A MATLAB® leállító szkriptjének elkészítése

A MATLAB® szabálytalan kilépéseit az alábbi utasítások nyomán tudjuk megszüntetni.

1. Hozzunk létre egy $MATLAB/toolbox/local/finish.m nevû állományt, majd írjuk bele ezt a sort:

```
! $MATLAB/bin/finish.sh
```

A `$MATLAB` szöveget pontosan így írjuk be.

Ugyanebben a könyvtárban találjuk a beállításaink kilépés elõtti mentéséért felelõs finishsav.m és finishdlg.m állományokat. Ha ezek valamelyikét módosítjuk, akkor az elõbbi parancsot közvetlenül a `save` után szúrjuk be.

2. Hozzunk létre egy $MATLAB/bin/finish.sh állományt, amelyben szerepeljen a következõ:

```
#!/usr/compat/linux/bin/sh
(sleep 5; killall -1 matlab_helper) &
exit 0
```

3. Tegyük végrehajthatóvá:

```
# chmod +x $MATLAB/bin/finish.sh
```

10.5.6. A MATLAB® használata

Most már a `matlab` parancs begépelésével bármikor elindíthatjuk.

10.6. Az Oracle® telepítése

10.6.1. Elõszó

Ez a leírás azt mutatja be, hogyan telepítsük FreeBSD-re az Oracle® 8.0.5 és Oracle® 8.0.5.1 Enterprise Edition Linux változatait.

10.6.2. A Linux környezet telepítése

Telepítsük az emulators/linux_base és devel/linux_devtools portokat a Portgyûjteményböl. Amennyiben ennek során nehézségekbe ütköznénk, próbálkozzunk a korábbi változataikkal.

Fel kell raknunk a Red Hat Tcl csomagját is, ha az alkalmazáshoz tartozó intelligens ügynököt is futtatni szeretnénk. Ez a tcl-8.0.3-20.i386.rpm. A hivatalos RPM port segítségével az alábbi általános parancson keresztül tudunk csomagokat telepíteni:

```
# rpm -i --ignoreos --root /compat/linux --dbpath /var/lib/rpm csomag
```

A *csomag* telepítésének semmilyen hibát nem kellene okoznia.

10.6.3. Az Oracle® környezetének létrehozása

Az Oracle® telepítéséhez elõször ki kell alakítanunk a megfelelõ környezetet. Ez a leírás *kifejezetten*

arról szól, hogy FreeBSD-n hogyan futtassuk a linuxos Oracle®-t, nem pedig az Oracle® telepítési útmutatójában bemutatottakat taglalja.

A rendszermag hangolása

Ahogy az Oracle® telepítési útmutatójában is olvashatjuk, be kell állítanunk az osztott memória maximális méretét. FreeBSD alatt erre a célra ne használjuk az `SHMMAX` értéket, mivel az `SHMMAX` az `SHMMAXPGS` és `PGSIZE` értékekbõl számolódik ki. Ezért nekünk itt a `SHMMAXPGS` értékét kell meghatároznunk. Minden egyéb beállítás történhet az útmutatóban megadottak szerint. Például:

```
options SHMMAXPGS=10000
options SHMMNI=100
options SHMSEG=10
options SEMMNS=200
options SEMMNI=70
options SEMMSL=61
```

Hangoljuk be ezeket az értékeket az Oracle® tervezett használatához.

Emellett a konfigurációs állományban ne feledkezzünk meg az alábbi beállítások megadásáról sem:

```
options SYSVSHM #SysV osztott memória
options SYSVSEM #SysV szemaforok
options SYSVMSG #SysV folyamatok közti kommunikáció
```

Az Oracle® hozzáférése

Egy rendes hozzáféréshez hasonlóan hozzunk létre egy külön `oracle` hozzáférést is rendszerünkön. Az `oracle` hozzáférés csak annyiban különleges, hogy linuxos parancsértelmezõt kell társítanunk hozzá. Ehhez vegyük fel `/compat/linux/bin/bash` sort az /etc/shells állományba, majd állítsuk át az `oracle` nevû felhasználó parancsértelmezõjét a /compat/linux/bin/bash programra.

Környezet

A megszokott Oracle® környezeti változók, mint például az `ORACLE_HOME` és `ORACLE_SID` mellett még definiálnunk kell a következõket is:

Változó	Érték
`LD_LIBRARY_PATH`	`$ORACLE_HOME/lib`
`CLASSPATH`	`$ORACLE_HOME/jdbc/lib/classes111.zip`
`PATH`	`/compat/linux/bin /compat/linux/sbin /compat/linux/usr/bin /compat/linux/usr/sbin /bin /sbin /usr/bin /usr/sbin /usr/local/bin $ORACLE_HOME/bin`

Javasoljuk, hogy az összes környezeti változót a .profile állományban adjuk meg. Ennek megfelelõen a példa beállításai így fognak kinézni benne:

```
ORACLE_BASE=/oracle; export ORACLE_BASE
ORACLE_HOME=/oracle; export ORACLE_HOME
LD_LIBRARY_PATH=$ORACLE_HOME/lib
export LD_LIBRARY_PATH
ORACLE_SID=ORCL; export ORACLE_SID
ORACLE_TERM=386x; export ORACLE_TERM
CLASSPATH=$ORACLE_HOME/jdbc/lib/classes111.zip
export CLASSPATH
PATH=/compat/linux/bin:/compat/linux/sbin:/compat/linux/usr/bin
PATH=$PATH:/compat/linux/usr/sbin:/bin:/sbin:/usr/bin:/usr/sbin
PATH=$PATH:/usr/local/bin:$ORACLE_HOME/bin
export PATH
```

10.6.4. Az Oracle® telepítése

A Linux emulátorban meghúzódó apró egyenletlenségek miatt a telepítés elõtt létre kell hoznunk egy .oracle nevû alkönyvtárat a /var/tmp könyvtárban. Helyezzük ezt az `oracle` felhasználó tulajdonába. Ezt követõen minden további gond nélkül képesek leszünk az Oracle® telepítésére. Ha netalán mégis problémákba ütköznénk, elõször mindig az Oracle® telepítési és konfigurációs állományait ellenõrizzük! Az Oracle® telepítése után rakjuk fel a következõ szakaszokban bemutatandó javításokat.

Gyakran problémát okoz, ha a TCP protokollt még nem telepítettük. Ennek következményeképpen ugyanis nem tudnak elindulni a TCP alapú szolgáltatások. Az alábbi mûveletek ebben igyekeznek segíteni:

```
# cd $ORACLE_HOME/network/lib
# make -f ins_network.mk ntcontab.o
# cd $ORACLE_HOME/lib
# ar r libnetwork.a ntcontab.o
# cd $ORACLE_HOME/network/lib
# make -f ins_network.mk install
```

Ne felejtsük el ismét elindítani a root.sh szkriptet!

A root.sh javítása

Az Oracle® telepítése során `root` (privilegizált) felhasználóként elvégzendõ mûveleteket a root.sh elnevezésû szkriptben találjuk. Ez a szkript az orainst könyvtárba kerül. A `chown` parancs helyes lefutásához alkalmazzuk az alább mellékelt javítást, vagy az egész szkriptet egy linuxos parancsértelmezõbõl indítsuk el.

```
*** orainst/root.sh.orig Tue Oct 6 21:57:33 1998
--- orainst/root.sh Mon Dec 28 15:58:53 1998
***************
*** 31,37 ****
# This is the default value for CHOWN
```

```
# It will redefined later in this script for those ports
# which have it conditionally defined in ss_install.h
! CHOWN=/bin/chown
#
# Define variables to be used in this script
--- 31,37 ----
# This is the default value for CHOWN
# It will redefined later in this script for those ports
# which have it conditionally defined in ss_install.h
! CHOWN=/usr/sbin/chown
#
# Define variables to be used in this script
```

Ha nem CD-rõl telepítjük az Oracle®-t, akkor akár a root.sh forrását is kijavíthatjuk. A neve rthd.sh, és a forrásfa orainst könyvtárában találhatjuk.

A genclntsh javítása

A `genclntsh` szkript a kliensek által használt osztott könyvtár létrehozására alkalmazható. Általában demók fordításához van rá szükség. Az alábbi javítás alkalmazásával a `PATH` változó értéke törölhetõ:

```
*** bin/genclntsh.orig Wed Sep 30 07:37:19 1998
--- bin/genclntsh Tue Dec 22 15:36:49 1998
***************
*** 32,38 ****
#
# Explicit path to ensure that we're using the correct commands
#PATH=/usr/bin:/usr/ccs/bin export PATH
! PATH=/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin export PATH
#
# each product MUST provide a $PRODUCT/admin/shrept.lst
--- 32,38 ----
#
# Explicit path to ensure that we're using the correct commands
#PATH=/usr/bin:/usr/ccs/bin export PATH
! #PATH=/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin export PATH
#
# each product MUST provide a $PRODUCT/admin/shrept.lst
```

10.6.5. Az Oracle® futtatása

Ha rendesen követtük az iménti utasításokat, akkor most már úgy tudjuk futtatni az Oracle®-t, mintha csak Linuxon futna.

10.7. Az SAP® R/3® telepítése

Az SAP® típusú rendszerek telepítéséhez FreeBSD-re hivatalosan nem kaphatunk mûszaki

segélynyújtást - csak a minõsített platformokat támogatják.

10.7.1. Elõszó

Ez a leírás az SAP® R/3® rendszer és Oracle® adatbázis Linux változatainak telepítését mutatja be FreeBSD-n, beleértve a FreeBSD és az Oracle® telepítését. Kétféle konfigurációt írunk le:

- SAP® R/3® 4.6B (IDES) és Oracle® 8.0.5, FreeBSD 4.3-STABLE
- SAP® R/3® 4.6C és Oracle® 8.1.7, FreeBSD 4.5-STABLE

Habár ez a dokumentum igyekszik az összes fontos lépést a lehetõ legrészletesebb módon tárgyalni, semmiképpen sem célja az Oracle® és az SAP® R/3® alkalmazásokhoz mellékelt telepítési útmutatók kiváltása.

A kifejezetten az SAP® vagy az Oracle® Linux változataira vonatkozó kérdések, valamint az Oracle® és az SAP® OSS konkrét használatával kapcsolatos leírások tekintetében a saját dokumentációjukat olvassuk el.

10.7.2. A szoftver

Az SAP® telepítéséhez az alábbi CD-ket használtuk fel:

SAP® R/3® 4.6B, Oracle® 8.0.5

Név	Szám	Leírás
KERNEL	51009113	SAP Kernel Oracle / telepítõ / AIX, Linux, Solaris
RDBMS	51007558	Oracle / RDBMS 8.0.5.X / Linux
EXPORT1	51010208	IDES / DB-Export / 1. lemez
EXPORT2	51010209	IDES / DB-Export / 2. lemez
EXPORT3	51010210	IDES / DB-Export / 3. lemez
EXPORT4	51010211	IDES / DB-Export / 4. lemez
EXPORT5	51010212	IDES / DB-Export / 5. lemez
EXPORT6	51010213	IDES / DB-Export / 6. (utolsó) lemez

Emellett még használtuk az Oracle® 8 Server (az elõzetes 8.0.5 változat a Linux 2.0.33 verziójához) CD-jét is, amely igazából nem feltétlenül szükséges, valamint a FreeBSD (a 4.3 RELEASE kiadása után nem sokkal levõ) 4.3-STABLE változatát.

SAP® R/3® 4.6C SR2, Oracle® 8.1.7

Név	Szám	Leírás
KERNEL	51014004	SAP Kernel Oracle / SAP Kernel 4.6D változat / DEC, Linux

Név	Szám	Leírás
RDBMS	51012930	Oracle 8.1.7/ RDBMS / Linux
EXPORT1	51013953	4.6C kiadás SR2 / Export / 1. lemez
EXPORT1	51013953	4.6C kiadás SR2 / Export / 2. lemez
EXPORT1	51013953	4.6C kiadás SR2 / Export / 3. lemez
EXPORT1	51013953	4.6C kiadás SR2 / Export / 4. (utolsó) lemez
LANG1	51013954	4.6C kiadás SR2 / Nyelvi támogatás / német, angol, francia / 1. lemez

A telepítendõ nyelvtõl függõen egyéb nyelvi támogatást tartalmazó CD használata is szükségessé válhat. Itt most csak a német és angol nyelveket használjuk, ezért elegendõ az elsõ CD. Csendben hozzátesszük, hogy mind a négy EXPORT CD száma megegyezik. Ugyanígy a három nyelvi CD-nek is megegyeznek a számai (ez eltér a 4.6B IDES kiadás CD számozásától). Az írás pillanatában a FreeBSD 4.5-STABLE (2002.03.20-i) változatát használjuk.

10.7.3. SAP® füzetek

Az SAP® R/3® telepítésével kapcsolatban az alábbi füzetek bizonyultak hasznosnak:

SAP® R/3® 4.6B, Oracle® 8.0.5

Szám	Cím
0171356	SAP Software on Linux: Essential Comments
0201147	INST: 4.6C R/3 Inst. on UNIX - Oracle
0373203	Update / Migration Oracle 8.0.5 -→ 8.0.6/8.1.6 LINUX
0072984	Release of Digital UNIX 4.0B for Oracle
0130581	R3SETUP step DIPGNTAB terminates
0144978	Your system has not been installed correctly
0162266	Questions and tips for R3SETUP on Windows NT / W2K

SAP® R/3® 4.6C, Oracle® 8.1.7

Szám	Cím
0015023	Initializing table TCPDB (RSXP0004) (EBCDIC)
0045619	R/3 with several languages or typefaces

Szám	Cím
0171356	SAP Software on Linux: Essential Comments
0195603	RedHat 6.1 Enterprise version: Known problems
0212876	The new archiving tool SAPCAR
0300900	Linux: Released DELL Hardware
0377187	RedHat 6.2: important remarks
0387074	INST: R/3 4.6C SR2 Installation on UNIX
0387077	INST: R/3 4.6C SR2 Inst. on UNIX - Oracle
0387078	SAP Software on UNIX: OS Dependencies 4.6C SR2

10.7.4. Hardverkövetelmények

Az alábbi hardvereszközök szükségesek az SAP® R/3® rendszer telepítéséhez. Az éles használathoz ennél természetesen valamivel több kell majd:

Változat	4.6B	4.6C
Processzor	Két Pentium® III 800MHz	Két Pentium® III 800MHz
Memória	1GB ECC	2GB ECC
Szabad hely a merevlemezen	50 - 60GB (IDES)	50 - 60GB (IDES)

Éles használatra nagyobb gyorsítótárral rendelkezõ Xeon™ processzorokat, nagysebességû háttértárakat (SCSI, hardveres RAID vezérlõvel), USV és ECC memória modulok ajánlottak. A nagy tárigényt egyébként az elõre beállított IDES rendszer indokolja, ami egy 27 GB méretû adatbázist hoz létre a telepítés során. Ez a terület általában elegendõ egy frissen induló rendszer és hozzá tartozó alkalmazásadatok tárolására.

SAP® R/3® 4.6B, Oracle® 8.0.5

A következõ hardverkonfigurációt használtuk: két 800 MHz-es Pentium® III processzor és a hozzájuk tartozó alaplap, egy Adaptec® 29160 Ultra160 SCSI-vezérlõ (a 40/80 GB méretû DLT szalagos meghajtó és CD-meghajtó használatához) és egy Mylex® AcceleRAID™ RAID-vezérlõ (2 csatorna, 6.00-1-00 verziójú firmware és 32 MB memória), amihez két 17 GB-os (tükrözött) merevlemez és négy 36 GB-os merevlemez (RAID 5) csatlakozik.

SAP® R/3® 4.6C, Oracle® 8.1.7

Itt a hardver egy Dell™ PowerEdge™ 2500 volt: kétprocesszoros alaplap, két darab 1000 MHz-es Pentium® III processzorral (fejenként 256 KB gyorsítótárral), 2 GB PC133-as ECC SDRAM memóriával, PERC/3 DC PCI RAID-vezérlõvel (128 MB memória), valamint egy EIDE DVD-meghajtóval. A RAID-vezérlõre két, egyenként 18 GB méretû merevlemezt (tükrözve) és négy 36 GB méretû merevlemezt csatlakoztattunk (RAID 5-ben).

10.7.5. A FreeBSD telepítése

Elõször is telepítenünk kell a FreeBSD-t. Ez több módon is lehetséges, ezekrõl a Saját telepítőeszköz elkészítéseben olvashatunk bõvebben.

A lemezek felosztása

Az egyszerûség kedvéért az SAP® R/3® 46B és SAP® R/3® 46C SR2 telepítése során is ugyanazt a felosztást használtuk. Egyedül az eszközök nevei változtak, mivel a telepítés eltérõ hardvereken történt (/dev/da) és /dev/amr, tehát ha az AMI MegaRAID® esetén a /dev/da0s1a helyett a /dev/amr0s1a eszközt láthatjuk):

Állományrendszer	Méret	Csatlakozási pont
/dev/da0s1a	1 GB	/
/dev/da0s1b	6 GB	lapozóállomány
/dev/da0s1e	2 GB	/var
/dev/da0s1f	8 GB	/usr
/dev/da1s1e	45 GB	/compat/linux/oracle
/dev/da1s1f	2 GB	/compat/linux/sapmnt
/dev/da1s1g	2 GB	/compat/linux/usr/sap

Elõre állítsuk be és inicializáljuk a két logikai meghajtót a Mylex® és a PERC/3 RAID-vezérlõkön. A hozzá tartozó szoftver a BIOS indításának fázisában hívható be.

A lemezek felosztása némileg eltér az SAP® által javasoltaktól, mivel az SAP® szerint az Oracle® könyvtárait (néhány másikkal együtt) külön-külön érdemes csatlakoztatni - mi most az egyszerûsítés kedvéért csak létrehoztuk ezeket.

A `make world` és egy új rendszermag

Töltsük le a legfrissebb -STABLE forrásokat. Fordítsuk újra az összes forrást (`make world`) és a beállításainak elvégzése után a saját rendszermagunkat is. Itt ne felejtsük el megadni az SAP® R/3® és az Oracle® mûködéséhez szükséges paramétereket.

10.7.6. A Linux környezet telepítése

Az linuxos alaprendszer telepítése

Elsõként a linux_base portot kell felraknunk (`root` felhasználóként):

```
# cd /usr/ports/emulators/linux_base-fc4
# make install distclean
```

A linuxos fejlesztõi környezet telepítése

Ha az Oracle®-t FreeBSD-re a Az Oracle® telepítéseban leírtak szerint akarjuk telepíteni, akkor szükségünk lesz a linuxos fejlesztõeszközökre is:

```
# cd /usr/ports/devel/linux_devtools
# make install distclean
```

A linuxos fejlesztõkörnyezetet csak az SAP® R/3® 46B IDES telepítésénél raktuk fel. Nincs rá szükségünk, ha a FreeBSD rendszeren nem akarjuk újralinkelni az Oracle® adatbázist. Pontosan ez a helyzet, amikor egy Linux rendszerhez gyártott Oracle® készletet használunk.

A szükséges RPM csomagok telepítése

Az `R3SETUP` elindításához PAM támogatásra is szükségünk lesz. Amikor elõször próbáltuk meg telepíteni a FreeBSD 4.3-STABLE változatára az SAP®-t, felraktuk a PAM-et és az összes hozzá tartozó csomagot, majd végül úgy bírtuk mûködésre, hogy kényszerítettük a PAM telepítését is. Az SAP® R/3® 4.6C SR2 esetén szintén sikerült önmagában felrakni a PAM RPM csomagját is, tehát úgy néz ki, hogy a függõségeit már nem kell telepíteni:

```
# rpm -i --ignoreos --nodeps --root /compat/linux --dbpath /var/lib/rpm \
pam-0.68-7.i386.rpm
```

Az Oracle® 8.0.5 verziójához mellékelt intelligens ügynök futtatásához fel kell rakni a RedHat tcl-8.0.5-30.i386.rpm nevû Tcl csomagját is (máskülönben a az Oracle® telepítése közben szükséges újralinkelés nem fog mûködni). Vannak ugyan egyébként is gondok az Oracle® újralinkelésével, azonban ez linuxos probléma, nem pedig FreeBSD-s.

Néhány további tipp

Hasznos lehet, ha felvesszük a `linprocfs` bejegyzést az /etc/fstab állományba. Ennek pontos részleteit a linprocfs(5) man oldalon találjuk meg. Másik fontos paraméter a `kern.fallback_elf_brand=3`, amelyet az /etc/sysctl.conf állományba kell beszúrnunk.

10.7.7. Az SAP® R/3® környezetének létrehozása

A szükséges állományrendszerek és csatlakozási pontok létrehozása

Egy egyszerûbb telepítéshez elég csupán a következõ állományrendszereket elkészíteni:

csatlakozási pont	méret GB-ban
/compat/linux/oracle	45 GB
/compat/linux/sapmnt	2 GB
/compat/linux/usr/sap	2 GB

Készítenünk kell még néhány linket is, különben az SAP® telepítõje panaszkodni fogni az

ellenõrzésük során:

```
# ln -s /compat/linux/oracle /oracle
# ln -s /compat/linux/sapmnt /sapmnt
# ln -s /compat/linux/usr/sap /usr/sap
```

Az egyik ilyen telepítés közben megjelenõ hibaüzenet (a *PRD* rendszer és az SAP® R/3® 4.6C SR2 telepítése esetén):

```
INFO 2002-03-19 16:45:36 R3LINKS_IND_IND SyLinkCreate:200
    Checking existence of symbolic link /usr/sap/PRD/SYS/exe/dbg to
    /sapmnt/PRD/exe. Creating if it does not exist...

WARNING 2002-03-19 16:45:36 R3LINKS_IND_IND SyLinkCreate:400
    Link /usr/sap/PRD/SYS/exe/dbg exists but it points to file
    /compat/linux/sapmnt/PRD/exe instead of /sapmnt/PRD/exe. The
    program cannot go on as long as this link exists at this
    location. Move the link to another location.

ERROR 2002-03-19 16:45:36 R3LINKS_IND_IND Ins_SetupLinks:0
    can not setup link '/usr/sap/PRD/SYS/exe/dbg' with content
    '/sapmnt/PRD/exe'
```

A felhasználók és könyvtárak létrehozása

Az SAP® R/3® rendszernek két felhasználóra és három csoportra van szüksége. Az igényelt felhasználók nevei az SAP® rendszer azonosítójától (System ID, SID) függenek, amely három betûbõl áll. Egyes ilyen rendszerazonosítók az SAP® számára vannak fenntartva. (Például a `SAP` és a `NIX`. Ezek teljes listáját az SAP® dokumentációjában találjuk meg.) Erre az IDES telepítéséhez az `IDS`, a 4.6C SR2 telepítésénél a `PRD` neveket adtuk, mivel ezeket a rendszereket éles használatra szánták. Ennélfogva a következõ csoportokat hoztuk létre hozzájuk (a csoportok azonosítói ugyan eltérhetnek az általunk használtaktól):

csoport azonosítója	csoport neve	leírás
100	dba	Adatbázis adminisztrátor
101	sapsys	SAP® rendszer
102	oper	Adatbázis operátor

Az Oracle® alapértelmezett telepítésénél csak a `dba` csoport jön létre. A `dba` csoportot `oper` csoportként is használhatjuk (bõvebb információkért lásd az Oracle® és az SAP® dokumentációját).

Ezenkívül az alábbi felhasználókra van még szükségünk:

felhasználói azonosító	felhasználói név	általános név	csoport	egyéb csoportok	leírás
1000	idsadm/prdadm	*sid*adm	sapsys	oper	SAP® adminisztrátor
1002	oraids/oraprd	ora*sid*	dba	oper	Oracle® adminisztrátor

Az adduser(8) parancs használata során a következõkre lesz szükségünk egy "SAP® Administrator" létrehozásához (figyeljük a parancsértelmezõt (shell) és a felhasználói könyvtárat (home directory)):

```
Name: sidadm
Password: ******
Fullname: SAP Administrator SID
Uid: 1000
Gid: 101 (sapsys)
Class:
Groups: sapsys dba
HOME: /home/sidadm
Shell: bash  (/compat/linux/bin/bash)
```

Ugyanígy az "Oracle® Administrator" esetében:

```
Name: orasid
Password: ******
Fullname: Oracle Administrator SID
Uid: 1002
Gid: 100 (dba)
Class:
Groups: dba
HOME: /oracle/sid
Shell: bash  (/compat/linux/bin/bash)
```

A `dba` és `oper` csoportok használata során ne felejtsük el megadni az `oper` csoportot sem.

Könyvtárak létrehozása

A könyvtárakat általában külön állományrendszerekként hozzák létre, de ez teljesen az igényeinken múlik. Mi most egyszerû könyvtárakként alakítottuk ki ezeket, ezért tulajdonképpen ugyanazon a RAID 5 tömbön találhatóak meg:

Ehhez elõször beállítjuk az egyes könyvtárak tulajdonosait és engedélyeit (`root` felhasználóként):

```
# chmod 775 /oracle
# chmod 777 /sapmnt
# chown root:dba /oracle
```

```
# chown sidadm:sapsys /compat/linux/usr/sap
# chmod 775 /compat/linux/usr/sap
```

Másodsorban `orasid` felhasználóként hozzuk létre az /oracle/SID alkönyvtárait:

```
# su - orasid
# cd /oracle/SID
# mkdir mirrlogA mirrlogB origlogA origlogB
# mkdir sapdata1 sapdata2 sapdata3 sapdata4 sapdata5 sapdata6
# mkdir saparch sapreorg
# exit
```

Az Oracle® 8.1.7 telepítésénél még további könyvtárakra is szükségünk lesz:

```
# su - orasid
# cd /oracle
# mkdir 805_32
# mkdir client stage
# mkdir client/80x_32
# mkdir stage/817_32
# cd /oracle/SID
# mkdir 817_32
```

A client/80x_32 könyvtárnak pontosan ilyen névvel kell rendelkeznie. Ne cseréljük ki a benne szereplõ *x*-et semmire se!

A harmadik lépésben létrehozzuk a `sidadm` felhasználóhoz tartozó könyvtárakat:

```
# su - sidadm
# cd /usr/sap
# mkdir SID
# mkdir trans
# exit
```

Az /etc/services

A SAP® R/3® mûködéséhez fel kell vennünk néhány olyan bejegyzést is az /etc/services állományba, amelyek a FreeBSD telepítése során nem jönnek létre. Így tehát írjuk be az alábbi sorokat (legalább a használni kívánt példány számához illõ sorokat adjuk meg - ez jelen esetünkben most a `00`. Természetesen az sem okoz gondot, ha a `dp`, `gw`, `sp` és `ms` esetén beírjuk az összes példánynak megfelelõ portot `00`-tól `99`-ig). Amennyiben a SAProuter vagy az SAP® OSS használatára lenne szükségünk, akkor adjuk meg a SAProuter által lefoglalt `99`-es példánynak megfelelõ 3299-es portot a rendszerünkön:

```
sapdp00  3200/tcp # SAP menetirányító        3200 + a példány száma
```

```
sapgw00  3300/tcp # SAP átjáró              3300 + a példány száma
sapsp00  3400/tcp #                         3400 + a példány száma
sapms00  3500/tcp #                         3500 + a példány száma
sapmsSID 3600/tcp # SAP üzenetkezelõ szerver 3600 + a példány száma
sapgw00s 4800/tcp # biztonságos SAP átjáró   4800 + a példány száma
```

A szükséges nyelvi beállítások

Az SAP®-nek legalább két olyan nyelvre van szüksége, amely nem része az alap RedHat telepítéseknek. Az SAP® a saját FTP szervereirõl elérhetõvé tette az ehhez szükséges RPM csomagokat (amelyek viszont csak OSS típusú hozzáférés birtokában tölthetõek le). A 0171356 számú jegyzet tartalmazza a beszerzendõ RPM-ek listáját.

Megcsinálhatjuk úgy is, hogy egyszerûen csak linkeket hozunk létre (például a *de_DE* és *en_US* könyvtárakra), habár ezt egy éles rendszer esetében semmiképpen sem ajánljuk (az IDES rendszerrel tapasztalataink szerint eddig még remekül mûködött). Az alábbi nyelvi beállítások fognak tehát nekünk kelleni:

```
de_DE.ISO-8859-1
en_US.ISO-8859-1
```

Így hozzuk létre hozzájuk a linkeket:

```
# cd /compat/linux/usr/shared/locale
# ln -s de_DE de_DE.ISO-8859-1
# ln -s en_US en_US.ISO-8859-1
```

A telepítés során az iméntiek hiánya gondokat okozhat. Ha folyamatosan figyelmen kívül hagyjuk az ezekbõl fakadó hibákat (vagyis a CENTRDB.R3S állományban a gondot okozó lépések `STATUS` értékét `OK`-ra állítjuk), akkor komolyabb erõfeszítések megtétele nélkül majd képtelenek leszünk bejelentkezni a frissen telepített SAP® rendszerünkbe.

A rendszermag finomhangolása

Az SAP® R/3® rendszerek temérdek mennyiségû erõforrást igényelnek. Ennek kielégítésére az alábbi paramétereket adjuk hozzá a rendszermag beállításait tartalmazó állományhoz:

```
# Adjunk a memóriazabálóknak (SAP és Oracle):
options MAXDSIZ="(1024*1024*1024)"
options DFLDSIZ="(1024*1024*1024)"
# Kell néhány System V beállítás is:
options SYSVSHM # SYSV típusú osztott memória be
options SHMMAXPGS=262144 # a megosztható memória maximális mérete lapokban
#options SHMMAXPGS=393216 # a 46C telepítésekor ezt használjuk
options SHMMNI=256 # az osztott memóriákhoz tartozó azonosítók maximális száma
options SHMSEG=100 # a futó programonként megosztható szegmensek maximuma
options SYSVMSG # SYSV típusú üzenetsorok
```

```
options MSGSEG=32767 # a rendszerben keringõ üzenetszegmensek maximális száma
options MSGSSZ=32 # az üzenetszegmensek mérete. 2 hatványa LEGYEN
options MSGMNB=65535 # maximális karakter üzenetsoronként
options MSGTQL=2046 # a rendszerben levõ üzenetek maximuma
options SYSVSEM # SYSV típusú szemaforok
options SEMMNU=256 # a szemaforok UNDO struktúráinak száma
options SEMMNS=1024 # a rendszerben levõ szemaforok száma
options SEMMNI=520 # a szemaforok azonosítóinak mennyisége
options SEMUME=100 # az UNDO kulcsok száma
```

Az itt megadott minimum értékek az SAP® által kiadott dokumentációkból származnak. Mivel a Linux változathoz errõl nincs külön leírás, ezért a (32 bites) HP-UX változat dokumentációi között érdemes ennek utánanézni. Mivel a 4.6C SR2 telepítéséhez használt rendszeren valamivel több fizikai memória állt rendelkezésünkre, ezért az osztott szegmensek méretét nagyobbra tudtuk megválasztani mind az SAP®, mind az Oracle® esetében, ami magyarázza a megosztható lapok nagyobb számát.

A FreeBSD i386™ változatának telepítése során hagyjuk meg a `MAXDSIZ` és `DFLDSIZ` értékek alapértelmezett 1 GB-os maximumát. Ellenkezõ esetben ezekhez hasonló furcsa hibaüzeneteket láthatunk: `ORA-27102: out of memory` vagy `Linux Error: 12: Cannot allocate memory`.

10.7.8. Az SAP® R/3® telepítése

Az SAP® CD-k elõkészítése

Sok CD-t kell a telepítés során mozgatni, tehát csatlakoztatni és leválasztani. Ha viszont elegendõ meghajtóval rendelkezünk, akkor akár csatlakoztathatjuk egyszerre is az összeset. Vagy felmásolhatjuk a CD-k tartalmát a nekik megfelelõ könyvtárakba:

```
/oracle/SID/sapreorg/cd-neve
```

ahol a *cd-neve* a következõk valamelyike: KERNEL, RDBMS, EXPORT1, EXPORT2, EXPORT3, EXPORT4, EXPORT5 és EXPORT6 (4.6B/IDES), valamint KERNEL, RDBMS, DISK1, DISK2, DISK3, DISK4 és LANG (4.6C SR2). A csatlakoztatott CD-ken található állományok neveinek nagybetûseknek kell lenniük. Ha nem így lenne, akkor a csatlakoztatásnál adjuk meg a `-g` opciót. Így tehát a következõ parancsokat kell kiadnunk:

```
# mount_cd9660 -g /dev/cd0a /mnt
# cp -R /mnt/* /oracle/SID/sapreorg/cd-neve
# umount /mnt
```

A telepítõszkript futtatása

Elsõként egy install nevû könyvtárat kell elõkészítenünk:

```
# cd /oracle/SID/sapreorg
# mkdir install
# cd install
```

Ezután futtassuk le a telepítõszkriptet, ami pedig bemásolja az install könyvtárba szinte az összes fontos állományt:

```
# /oracle/SID/sapreorg/KERNEL/UNIX/INSTTOOL.SH
```

Az IDES (4.6B) változathoz egy teljes SAP® R/3® bemutató rendszer is tartozik, ezért a megszokott három CD helyett hat EXPORT típusú CD-bõl áll. Itt a CENTRDB.R3S telepítõsablon csak a szabvány központi példányt hozza létre (R/3® és az adatbázis), az IDES központi példányát már nem. Ezért az EXPORT1 könyvtárból ki kell másolnunk a CENTRDB.R3S állományt, különben az `R3SETUP` csak három EXPORT CD-t fog kérni.

Az újabb SAP® 4.6 SR2 kiadáshoz négy EXPORT CD tartozik. A telepítés folyamatát a CENTRAL.R3S állományban levõ paraméterek vezérlik. A korábbi kiadásokkal ellentétben nincsenek külön sablonok az adatbázissal és a nélküle telepítendõ központi példányok számára. Az SAP® az adatbázisok telepítésére külön sablont használ. Újrakezdéskor a telepítést ettõl függetlenül elegendõ az eredeti állománnyal újraindítani.

A telepítés közben és után az SAP®-nek a `hostname` paranccsal csak a gép saját nevét, nem pedig a teljes hálózati nevét kell megadnunk. Ilyenkor ezt vagy egyenként begépeljük, vagy létrehozunk rá egy álnevet az `ora_sid_` és `_sid_adm` (valamint a megfelelõ lépésekben a `root`) felhasználóknak: `alias hostname='hostname -s'`. Ezenkívül még az SAP® telepítésekor létrehozott mindkét felhasználó .profile és .login állományait is beállíthatjuk ennek megfelelõen.

Az `R3SETUP` 4.6B verziójának indítása

Ne felejtsük el jól beállítani az `LD_LIBRARY_PATH` környezeti változót:

```
# export LD_LIBRARY_PATH=/oracle/IDS/lib:/sapmnt/IDS/exe:/oracle/805_32/lib
```

A telepítés könyvtárában `root` felhasználóként indítsuk el az `R3SETUP` programot:

```
# cd /oracle/IDS/sapreorg/install
# ./R3SETUP -f CENTRDB.R3S
```

A szkript ezek után feltesz néhány kérdést (az alapértelmezett válaszok zárójelben, közvetlenül a megadottak után):

Kérdés	Alapértelmezés	Válasz
Enter SAP System ID	[C11]	IDS`Enter`
Enter SAP Instance Number	[00]	`Enter`

Kérdés	Alapértelmezés	Válasz
Enter SAPMOUNT Directory	[/sapmnt]	`Enter`
Enter name of SAP central host	[troubadix.domain.de]	`Enter`
Enter name of SAP db host	[troubadix]	`Enter`
Select character set	[1] (WE8DEC)	`Enter`
Enter Oracle server version (1) Oracle 8.0.5, (2) Oracle 8.0.6, (3) Oracle 8.1.5, (4) Oracle 8.1.6		1`Enter`
Extract Oracle Client archive	[1] (Yes, extract)	`Enter`
Enter path to KERNEL CD	[/sapcd]	/oracle/IDS/sapreorg/KERNEL
Enter path to RDBMS CD	[/sapcd]	/oracle/IDS/sapreorg/RDBMS
Enter path to EXPORT1 CD	[/sapcd]	/oracle/IDS/sapreorg/EXPORT1
Directory to copy EXPORT1 CD	[/oracle/IDS/sapreorg/CD4_DIR]	`Enter`
Enter path to EXPORT2 CD	[/sapcd]	/oracle/IDS/sapreorg/EXPORT2
Directory to copy EXPORT2 CD	[/oracle/IDS/sapreorg/CD5_DIR]	`Enter`
Enter path to EXPORT3 CD	[/sapcd]	/oracle/IDS/sapreorg/EXPORT3
Directory to copy EXPORT3 CD	[/oracle/IDS/sapreorg/CD6_DIR]	`Enter`
Enter path to EXPORT4 CD	[/sapcd]	/oracle/IDS/sapreorg/EXPORT4
Directory to copy EXPORT4 CD	[/oracle/IDS/sapreorg/CD7_DIR]	`Enter`
Enter path to EXPORT5 CD	[/sapcd]	/oracle/IDS/sapreorg/EXPORT5
Directory to copy EXPORT5 CD	[/oracle/IDS/sapreorg/CD8_DIR]	`Enter`
Enter path to EXPORT6 CD	[/sapcd]	/oracle/IDS/sapreorg/EXPORT6
Directory to copy EXPORT6 CD	[/oracle/IDS/sapreorg/CD9_DIR]	`Enter`
Enter amount of RAM for SAP + DB		850`Enter` (megabyte)
Service Entry Message Server	[3600]	`Enter`
Enter Group-ID of sapsys	[101]	`Enter`
Enter Group-ID of oper	[102]	`Enter`
Enter Group-ID of dba	[100]	`Enter`
Enter User-ID of _sid_adm	[1000]	`Enter`
Enter User-ID of ora_sid_	[1002]	`Enter`
Number of parallel procs	[2]	`Enter`

Ha a CD-ket nem különbözõ helyekre másoltuk, akkor az SAP® telepítõje nem fogja megtalálni ezeket (a rajtuk levõ LABEL.ASC segít neki az azonosításban) és kérni fogja a CD csatlakoztatását, illetve a csatlakozási pontjának megadását.

A CENTRDB.R3S sem minden esetben mentes a hibáktól. A tapasztalataink szerint az EXPORT4 címkéjû CD-t kérte újra, miközben a helyes kulcsokat jelezte ki (6_LOCATION, majd 7_LOCATION stb.), így egyszerûen csak lépjünk tovább az értékek meghagyásával.

Függetlenül az imént említett problémáktól, egészen az Oracle® adatbáziskezelõ telepítéséig mindennek mûködnie kellene.

Az R3SETUP 4.6C SR2 elindítása

Állítsuk be jól az LD_LIBRARY_PATH környezeti változó értékét. Ez némileg eltér a 4.6B és az Oracle® 8.0.5 párosának beállításától:

```
# export LD_LIBRARY_PATH=/sapmnt/PRD/exe:/oracle/PRD/817_32/lib
```

A telepítés könyvtárából root felhasználóként indítsuk el az R3SETUP programot:

```
# cd /oracle/PRD/sapreorg/install
# ./R3SETUP -f CENTRAL.R3S
```

A szkript ezek után feltesz néhány kérdést (az alapértelmezett válaszok zárójelben, közvetlenül a megadottak után):

Kérdés	Alapértelmezés	Válasz
Enter SAP System ID	[C11]	PRD Enter
Enter SAP Instance Number	[00]	Enter
Enter SAPMOUNT Directory	[/sapmnt]	Enter
Enter name of SAP central host	[majestix]	Enter
Enter Database System ID	[PRD]	PRD Enter
Enter name of SAP db host	[majestix]	Enter
Select character set	[1] (WE8DEC)	Enter
Enter Oracle server version (2) Oracle 8.1.7		2 Enter
Extract Oracle Client archive	[1] (Yes, extract)	Enter
Enter path to KERNEL CD	[/sapcd]	/oracle/PRD/sapreorg/KERNEL
Enter amount of RAM for SAP + DB	2044	1800 Enter (megabyte)
Service Entry Message Server	[3600]	Enter
Enter Group-ID of sapsys	[100]	Enter
Enter Group-ID of oper	[101]	Enter
Enter Group-ID of dba	[102]	Enter

Kérdés	Alapértelmezés	Válasz
Enter User-ID of `oraprd`	[1002]	`Enter`
Enter User-ID of `prdadm`	[1000]	`Enter`
LDAP support		3 `Enter` (nincs támogatás)
Installation step completed	[1] (continue)	`Enter`
Choose installation service	[1] (DB inst,file)	`Enter`

Az OSUSERDBSID_IND_ORA és OSUSERIDADM_IND_ORA lépésekben az `orasid` és `sidadm` felhasználók létrehozása hibákra futhat.

Függetlenül az említett problémáktól, az Oracle® adatbáziskezelõ telepítéséig mindennek remekül kell mûködnie.

10.7.9. Az Oracle® 8.0.5 telepítése

Az Oracle® Linux változatának telepítése során felmerülõ problémák tekintetében keressük fel az SAP® füzeteket és az Oracle® Readme állományait. A legtöbb, ha nem is az összes gondot az egymással nem kompatibilis függvénykönyvtárak okozzák.

Az Oracle® telepítésének részleteit a Az Oracle® telepítése címû szakaszban találjuk.

Az Oracle® 8.0.5 telepítése az `orainst` segítségével

Az Oracle® 8.0.5 verziójának használata esetén néhány további függvénykönyvtár újralinkelésére is szükség lesz, mivel az Oracle® 8.0.5 még a régi glibc könyvtárral lett fordítva (RedHat 6.0), viszont a RedHat 6.1 már a glibc újabb verzióját használja. A linkelés mûködéséhez az alábbi csomagokat kell még telepítenünk:

- compat-libs-5.2-2.i386.rpm
- compat-glibc-5.2-2.0.7.2.i386.rpm
- compat-egcs-5.2-1.0.3a.1.i386.rpm
- compat-egcs-c++-5.2-1.0.3a.1.i386.rpm
- compat-binutils-5.2-2.9.1.0.23.1.i386.rpm

A részleteket lásd az SAP® füzeteiben vagy az Oracle® Readme állományaiban. Amennyiben ez nem oldható meg, akkor az eredeti binárisok, esetleg az eredeti RedHat rendszerbõl származó újralinkelt binárisok is használhatóak (habár a telepítés pillanatában személyesen ezt nem tudtuk ellenõrizni).

Az intelligens ügynök lefordításához fel kell raknunk a RedHat saját Tcl csomagját. Ha ehhez nem tudjuk beszerezni a tcl-8.0.3-20.i386.rpm csomagot, akkor a RedHat 6.1 változatához készült tcl-8.0.5-30.i386.rpm is megteszi.

Az újralinkeléstõl eltekintve a telepítés többi része szinte adja magát:

```
# su - oraids
```

```
# export TERM=xterm
# export ORACLE_TERM=xterm
# export ORACLE_HOME=/oracle/IDS
# cd $ORACLE_HOME/orainst_sap
# ./orainst
```

Az *Oracle® On-Line Text Viewer* kikapcsolásán (mivel az jelenleg Linux alatt sem érhetõ el) kívül mindegyik képernyõt hagyjuk jóvá az `Enter` billentyû lenyomásával. Az Oracle® ezután a rendelkezésre álló `gcc`, `egcs` vagy `i386-redhat-linux-gcc` helyett a `i386-glibc20-linux-gcc` használatával újra akarja linkelni magát.

Idõ hiányában az Oracle® 8.0.5 PreProduction kiadásából emeltünk ki binárisokat, de az adatbáziskezelõ rendszer felélesztésére tett elsõ kísérleteink kudarcba fulladtak, és ezután a megfelelõ RPM-ek összeszedése valódi rémálomnak bizonyult.

Az Oracle® 8.0.5 Pre-production Release for Linux (Kernel 2.0.33) telepítése

A telepítés nagyon könnyû. Csatlakoztassuk a CD-t, majd indítsuk el a telepítõt. Ezután meg kell adnunk az Oracle® felhasználói könyvtárát és a telepítõ odamásolja az összes binárist. Habár a telepítés megkezdése elõtt a korábbi kísérleteink nyomát nem tüntettük el.

Ezt követõen az Oracle® adatbázisrendszer minden további gond nélkül elindítható.

10.7.10. Az Oracle® 8.1.7 Linux változatának telepítése

Szedjük le az oracle8172.tgz állományt a Linux rendszeren létrehozott könyvtárából, és bontsuk ki a /oracle/SID/817_32/ könyvtárba.

10.7.11. Az SAP® R/3® telepítésének folytatása

Elõször is ellenõrizzük az `isamd` (*sid*adm) és `oraids` (ora*sid*) felhasználók környezeti beállításait. A .profile, .login és .cshrc állományaikban a korábbi beállítások szerint kell szerepelnie a `hostname` parancsnak. Ha még mindig a teljes hálózati név lenne meg bennük, akkor a `hostname` parancsot át kell írni mind a három állományban a `hostname -s` parancsra.

Az adatbázis feltöltése

Ezután az `R3SETUP` folytatható vagy újraindítható (attól függõen, hogy a kilépést választottuk-e vagy sem). Az `R3SETUP` ekkor létrehozza az adatbázisban a táblákat és az `R3load` meghívásával feltölti ezeket adatokkal (a 46B IDES változat esetében az EXPORT1 - EXPORT6, a 46C esetében pedig a DISK1 - DISK4 lemezekrõl).

Amikor a feltöltés befejezõdött (ami akár órákig is eltarthat), szükség lesz még néhány jelszó megadására is. A próbatelepítéseknél nyugodtan használhatjuk a jól ismert alapértelmezett jelszavakat (azonban mindenképpen változtassuk meg ezeket, ha egy kicsit is számít a biztonság!):

Kérdés	Válasz
Enter Password for sapr3	sap`Enter`

Kérdés	Válasz
Confirum Password for sapr3	sap`Enter`
Enter Password for sys	change_on_install`Enter`
Confirm Password for sys	change_on_install`Enter`
Enter Password for system	manager`Enter`
Confirm Password for system	manager`Enter`

A 4.6B telepítése során még gondjaink akadtak a `dipgntab` használatával.

Az Oracle® Listener elindítása

Így kell elindítani az `orasid` felhasználóval az Oracle® Listenert:

```
% umask 0; lsnrctl start
```

Ha máshogy próbálkozunk, akkor az ORA-12546 kódú hibát fogjuk kapni, mert a hálózati portok socketei nem rendelkeznek a szükséges engedélyekkel. Lásd a 072984-es SAP® füzet.

Az MNLS táblák frissítése

Ha nem Latin 1 kódolású nyelveket akarunk importálni az SAP® rendszerbe, akkor frissítenünk kell a többnyelvû nyelvi támogatáshoz (Multi National Language Support, MNLS) tartozó táblázatokat. Ezek bemutatását a 15023 és 45619 számú SAP® OSS füzetekben olvashatjuk. Minden más esetben az SAP® telepítésekor nyugodtan kihagyhatjuk.

Ha még nincs is konkrétan szükségünk az MNLS-re, akkor is ellenõriznünk és inicializálnunk kell a TCPDB táblát. A 0015023 és 0045619 számú SAP® füzetekben tudhatunk meg errõl többet.

10.7.12. Telepítés utáni teendõk

Az SAP® R/3® licenckulcsának megszerzése

Az SAP® R/3® licenckulcsát külön kell kérni. Fontos, mert a telepítéshez használatos ideiglenes licenc csak négy hétig érvényes. Elõször szerezzük meg a hardverkulcsot. Jelentkezzünk be az `idsadm` felhasználóval és adjuk ki a `saplicense` parancsot:

```
# /sapmnt/IDS/exe/saplicense -get
```

A `saplicense` paraméter nélkül meghívására válaszul opciókat listáz ki. A licenckulcsot megérkezése után így tudjuk élesíteni:

```
# /sapmnt/IDS/exe/saplicense -install
```

Ezután a következõ értékeket kell megadni:

```
SAP SYSTEM ID   = SID, 3 karakter
CUSTOMER KEY    = hardverkulcs, 11 karakter
INSTALLATION NO = telepítés száma, 10 számjegy
EXPIRATION DATE = éééévhhnn, tehát "99991231"
LICENSE KEY     = licenckulcs, 24 karakter
```

A felhasználók létrehozása

Hozzunk létre egy felhasználót a 000 kliensen belül (a csak rajta belül elvégezhetõ feladatokhoz, aki különbözik a `sap*` és `ddic` felhasználóktól). Felhasználónévként általában a `wartung` nevet választottuk (ami angolul a `service` névnek, avagy szolgáltatásnak felel meg). A `sap_new` és `sap_all` nevû profilok is kellenek. A biztonságosság kedvéért a kliens összes alapértelmezett felhasználójának (beleértve a `sap*` és `ddic` felhasználókat is) változtassuk meg a jelszavát.

A szállítási rendszer, a profilok, mûködési módok stb. beállítása

A `ddic` és `sap*` felhasználóktól eltérõ nevû felhasználóval a 000 kliensen belül legalább a következõket végezzük el:

Feladat	Tranzakció
A szállítási rendszer (Transport System) beállítása, például a *Stand-Alone Transport Domain Entity* értékre	STMS
A rendszer profiljának létrehozása és szerkesztése	RZ10
A mûködési módok és példányok karbantartása	RZ04

Az iménti és az összes többi telepítés utáni lépések leírása teljes egészében megtalálható az SAP® telepítési útmutatóiban.

Az initsid.sap (initIDS.sap) szerkesztése

Az /oracle/IDS/dbs/initIDS.sap állomány tartalmazza a SAP® tartalék profilját. Itt többek közt a használni kívánt szalag méretét, a tömörítés típusát és hasonló paramétereket kell definiálni. A `sapdba` / `brbackup` futtatásához a következõ értékeket változtattuk meg:

```
compress = hardware
archive_function = copy_delete_save
cpio_flags = "-ov --format=newc --block-size=128 --quiet"
cpio_in_flags = "-iuv --block-size=128 --quiet"
tape_size = 38000M
tape_address = /dev/nsa0
tape_address_rew = /dev/sa0
```

Magyarázat:

`compress` (tömörítés): HP DLT1 típusú szalagot használtunk, ami tud hardveres tömörítést.

`archive_function` (archiválási házirend): Ez adja meg, hogy alapértelmezés szerint mi történjen az Oracle® archivált naplóival: az új naplóállományok elõször a szalagra mentõdnek, majd a már lementett naplók ismét mentésre kerülnek és végül törlõdnek. Ezzel sok fejfájástól menekülünk meg, mivel ilyenkor az archiváló szalagok esetleges sérülése esetén is valószínûleg képesek leszünk visszaállítani az adatbázist.

`cpio_flags` (a cpio beállítása): A `-B` használata alapértelmezés, amivel a blokkok mérete 5120 byte-ra állítódik. A DLT típusú szalagokhoz a HP legalább 32 KB-os blokkméretet javasolt, ezért a `--block -size=128` beállítással ezt 64 KB-ra növeltük. Szükségünk volt a `--format=newc` beállításra is, mivel 65535-nél több inode számunk van. Az utolsó beállítás a `--quiet`, amivel megakadályozzuk, hogy a `cpio` lementett blokkokat összefoglaló kijelzésére begerjedjen a `brbackup`.

`cpio_in_flags` (a cpio bemeneti beállításai): A szalagok visszatöltésénél használt beállítások. A formátumot automatikusan felismeri.

`tape_size` (szalagméret): Ezzel adjuk meg általában a szalag nyers kapacitását. Biztonsági okokból (hardveres tömörítést használunk) ez az érték a ténylegesnél valamivel kisebb.

`tape_address` (szalagos eszköz): a `cpio` által használható nem visszatekerhetõ eszköz.

`tape_address_rew` (visszatekerhetõ szalagos eszköz): A `cpio` által használható visszatekerhetõ eszköz.

Telepítés utáni beállítások

Az SAP® alábbi paramétereit kell beállítani a telepítés után (IDES 46B, 1 GB memóriával):

Név	Érték
ztta/roll_extension	250000000
abap/heap_area_dia	300000000
abap/heap_area_nondia	400000000
em/initial_size_MB	256
em/blocksize_kB	1024
ipc/shm_psize_40	70000000

0013026 SAP® füzet:

Név	Érték
ztta/dynpro_area	2500000

0157246 SAP® füzet:

Név	Érték
rdisp/ROLL_MAXFS	16000

Név	Érték
rdisp/PG_MAXFS	30000

A fenti paraméterek használatával egy 1 gigabyte fizikai memóriával rendelkezõ rendszer esetén nagyjából így alakul a memóriahasználat:

```
Mem: 547M Active, 305M Inact, 109M Wired, 40M Cache, 112M Buf, 3492K
Free
```

(547 MB aktív, 305 MB inaktív, 109 MB rögzített, 40 MB gyorsítótár, 112 MB puffer, 3492 KB szabad)

10.7.13. A telepítés során adódó problémák

Az R3SETUP újraindítása egy probléma kijavítása után

Az R3SETUP hiba esetén leáll. Miután átnéztük a hibára utaló naplókat és elhárítottuk a hiba okát, újra el kell indítanunk az R3SETUP programot, majd a REPEAT opció kiválasztásával próbáljuk megismételni az R3SETUP által kifogásolt legutóbbi mûveletet.

Az R3SETUP újraindításához egyszerûen adjuk meg a megfelelõ R3S állományt:

```
# ./R3SETUP -f CENTRDB.R3S
```

a 4.6B verzió esetén, vagy a

```
# ./R3SETUP -f CENTRAL.R3S
```

a 4.6C verzió esetén, függetlenül attól, hogy a hiba a CENTRAL.R3S vagy DATABASE.R3S állományoknál keletkezett.

Egyes lépéseknél az R3SETUP úgy véli, hogy az SAP® programjai mûködnek (mivel a hozzájuk tartozó lépéseket már megtettük), így a hibák miatt az adatbázist esetleg korábban nem tudta elindítani. Ezért a hibák kijavításának végeztével az R3SETUP ismételt indítása elõtt nekünk kell beindítani mind az adatbázist, mind pedig az SAP® rendszert.

Ne felejtsük el újra elindítani az Oracle® Listener segédprogramját sem (az `ora_sid_` felhasználóval adjuk ki a `umask 0; lsnrctl start` parancsot), ha az idõközben leállt volna (például a rendszer kényszerû újraindítása miatt).

OSUSERSIDADM_IND_ORA az R3SETUP közben

Ha az R3SETUP panaszkodik ebben a lépésben, akkor írjuk át az általa ekkor használt sablont (a 4.6B esetén ez a CENTRDB.R3S, illetve a 4.6C esetén ez a CENTRAL.R3S vagy a DATABASE.R3S). Keressük

a [OSUSERSIDADM_IND_ORA] szöveget, vagy csak a STATUS=ERROR bejegyzést, majd írjuk be a következõ értékeket:

```
HOME=/home/sidadm (üres volt)
STATUS=OK (ERROR státusza volt)
```

Ezután indítsuk újra az R3SETUP programot.

OSUSERDBSID_IND_ORA az R3SETUP közben

Az R3SETUP ebben a lépésben is hajlamos panaszkodni. Az itt felbukkanó hiba hasonló az OSUSERSIDADM_IND_ORA lépésben jelentkezõhöz. Szerkesszük át az R3SETUP által ilyenkor használt sablont (4.6B verzió esetén ez a CENTRDB.R3S, illetve 4.6C verziónál a CENTRAL.R3S vagy DATABASE.R3S). Keressük meg a [OSUSERDBSID_IND_ORA] részt, vagy csak a STATUS=ERROR bejegyzést, majd írjuk át az ebben a szakaszban szereplõ értéket így:

```
STATUS=OK
```

Indítsuk újra az R3SETUP programot.

oraview.vrf FILE NOT FOUND hiba az Oracle® telepítése közben

A telepítés megkezdése elõtt nem tiltottuk le az *Oracle® On-Line Text Viewer* felrakását. Habár Linux esetén ez nem használható, alapértelmezés szerint mégis ki van választva. Az Oracle® telepítõ menüjében tiltsuk le ezt és nélküle kezdjük újra a telepítést.

TEXTENV_INVALID hiba az R3SETUP, RFC vagy SAPgui Start programokban

Ha ilyen hibával kerülünk szembe, akkor hiányoznak a megfelelõ nyelvi állományok. A 0171356 SAP® füzet tartalmazza a telepítendõ RPM csomagok felsorolását (például a RedHat 6.1 esetén a saplocales-1.0-3 és saposcheck-1.0-1). Amennyiben figyelmen kívül hagyjuk az ilyen hibákat, és az R3SETUP minden kiakadásánál átírjuk (a CENTRDB.R3S állományban) az STATUS értékét az ERROR értékrõl az OK értékre és újraindítjuk, az SAP® nem állítódik be jól és nem tudunk a SAPgui alkalmazással rácsatlakozni a frissen telepített rendszerre még akkor sem, ha el tudtuk indítani. Amikor a régebbi linuxos SAPgui alkalmazással csatlakozunk, a következõ üzeneteket kapjuk:

```
Sat May 5 14:23:14 2001
*** ERROR => no valid userarea given [trgmsgo. 0401]
Sat May 5 14:23:22 2001
*** ERROR => ERROR NR 24 occured [trgmsgi. 0410]
*** ERROR => Error when generating text environment. [trgmsgi. 0435]
*** ERROR => function failed [trgmsgi. 0447]
*** ERROR => no socket operation allowed [trxio.c 3363]
Speicherzugriffsfehler
```

Ez a viselkedés annak köszönhetõ, hogy az SAP® R/3® nem képes jól összerendelni a nyelvi beállításokat, sõt, magát sem képes jól beállítani (hiányoznak némely bejegyzések az adatbázis

egyes tábláiban). Az SAP®-hez úgy tudunk ilyenkor csatlakozni, ha a DEFAULT.PFL állományba felvesszük a következõ bejegyzéseket (lásd 0043288 füzet):

```
abap/set_etct_env_at_new_mode = 0
install/collate/active = 0
rscp/TCP0B = TCP0B
```

Majd indítsuk újra az egész SAP® rendszert. Ezután már tudunk csatlakozni hozzá, még ha az országra jellemzõ nyelvi beállítások nem is mûködnek tökéletesen. Miután korrigáltuk az ország beállításait (és felraktuk a megfelelõ nyelvi állományokat), távolítsuk el az iménti bejegyzéseket a DEFAULT.PFL állományból és indítsuk újra az SAP® rendszert.

Az ORA-00001 hiba

Ez a hiba FreeBSD alatt az Oracle® 8.1.7 használata során következhet be. Akkor történik, amikor az Oracle® adatbázis nem volt képes rendesen inicializálni magát és összeomlott, aminek révén szemaforokat és memóriát hagyott megosztva a rendszerben. Így az adatbázis következõ indításakor kapunk egy kövér ORA-00001 hibát.

Az `ipcs -a` paranccsal keressük meg ezeket, majd az `ipcrm` segítségével pedig számoljuk fel.

Az ORA-00445 (a PMON háttérprogram nem indult el) hiba

Ez a hiba az Oracle® 8.1.7 használatakor következhet be. Akkor kapjuk ezt a hibát, amikor `prdadm` felhasználóként a elindítjuk `startsap` szkriptet (például `startsap_majestix_00`).

Erre gyógyír lehet, ha ehelyette az adatbázis elindításához az `oraprd` felhasználóval adjuk ki az `svrmgrl` parancsot:

```
% svrmgrl
SVRMGR> connect internal;
SVRMGR> startup;
SVRMGR> exit
```

Az ORA-12546 (A Listener indítása megfelelõ engedélyekkel) hiba

Az Oracle® Listener alkalmazását `oraids` felhasználóként az alábbi paranccsal indítsuk el:

```
# umask 0; lsnrctl start
```

Máskülönben ORA-12546 hibát kapunk, mivel a hálózati portokhoz tartozó socketek nem rendelkeznek a megfelelõ engedélyekkel. Lásd 0072984 SAP® füzet.

Az ORA-27102 (Nincs elég memória) hiba

Akkor fordul elõ ilyen hiba, amikor a `MAXDSIZ` és `DFLDSIZ` értékeit 1 GB-nál (1024 x 1024 x 1024-nél) nagyobbra állítottuk. Mellé még kapunk egy `Linux Error 12: Cannot allocate memory` hibát is.

[DIPGNTAB_IND_IND] az R3SETUP közben

Errõl alapvetõen a 0130581 számú SAP® füzet ad tájékoztatást (az R3SETUP DIPGNTAB lépése hibára fut). Az IDES telepítése során az SAP® rendszer valamiért az "IDS" név helyett egy üres karakterláncot használ. Ez a könyvtárak elérésében kisebb gondokat okoz, mivel az elérési útvonaluk a *SID*-bõl generálódik (ami ebben az esetben az IDS). Tehát a

```
/usr/sap/IDS/SYS/...
/usr/sap/IDS/DVMGS00
```

helyett a következõt próbálja meg elérni:

```
/usr/sap//SYS/...
/usr/sap/D00
```

A telepítés folytatásához létrehoztunk egy linket és egy másik könyvtárat:

```
# pwd
/compat/linux/usr/sap
# ls -l
total 4
drwxr-xr-x 3  idsadm sapsys 512 May 5 11:20 D00
drwxr-x--x 5  idsadm sapsys 512 May 5 11:35 IDS
lrwxr-xr-x 1  root   sapsys 7 May 5 11:35 SYS -> IDS/SYS
drwxrwxr-x 2  idsadm sapsys 512 May 5 13:00 tmp
drwxrwxr-x 11 idsadm sapsys 512 May 4 14:20 trans
```

Észrevettük, hogy a SAP® füzetekben (0029227 és 0008401) ugyanezt a viselkedést írják le. Az SAP® 4.6C telepítésénél azonban ilyen hibával nem találkoztunk.

[RFCRSWBOINI_IND_IND] az R3SETUP közben

Az SAP® 4.6C telepítése folyamán ez a hiba csupán egy korábban bekövetkezett másik hiba utóhatása volt. Itt át kell néznünk az összes érintett naplót és ki kell javítanunk a tényleges problémát.

Amennyiben a naplók átvizsgálása után csak ezt találjuk egyedüli hibának (lásd SAP® füzetek), állítsuk át (a CENTRDB.R3S állományban) a STATUS értékét az OK értékre, majd indítsuk újra az R3SETUP programot. A telepítés befejezése után hajtsuk végre az SE38 tranzakcióból az RSWBOINS riportot. A további RFCRSWBOINI és RFCRADDBDIF lépésekkel kapcsolatban lásd a 0162266 SAP® füzetet.

[RFCRADDBDIF_IND_IND] az R3SETUP közben

Itt az elõbbihez hasonló feltételek élnek: mindenképpen ellenõrizzük a naplókban, hogy a hibát nem egy korábban keletkezett hiba okozta.

Ha tényleg csak az 0162266 SAP® füzetben leírtak érvényesek, akkor (a CENTRDB.R3S

állományban) állítsuk a gondot okozó lépés STATUS értékét az ERROR értékrõl az OK értékre, és indítsuk újra az R3SETUP programot. A telepítés után pedig hajtsuk végre az SE38 tranzakcióból az RADDBDIF riportot.

A sigaction sig31: File size limit exceeded hiba

Ez a *disp és work* SAP® programok indítása során történhet meg. Az SAP® rendszert indító startsap szkriptrõl leválva indulnak el a többi SAP® program elindításáért felelõs alfolyamatok. Ennek eredményeképpen a szkript maga nem fogja észrevenni a hibát.

Az SAP® programok elindulását az ps ax | grep SID paranccsal tudjuk ellenõrizni. Az eredményül kapott listában az összes aktív Oracle® és SAP® programnak szerepelnie kell. Ha ebbõl az tûnik ki, hogy bizonyos programok hiányoznak, vagy nem képesek kapcsolódni az SAP® rendszerhez, akkor az /usr/sap/SID/DVEBMGSnr/work/ könyvtárban nézzük át a hozzájuk tartozó naplóállományokat. Elsõsorban a dev_ms és a dev_disp állományok fontosak számunkra.

A 31-es jelzés akkor keletkezik, ha az Oracle® és az SAP® által használt osztott memória mértéke meghaladja a rendszermag beállításai közt megadott értéket. Ezt tehát ennek növelésével lehet orvosolni:

```
# az éles 46C rendszereknek több kell:
options SHMMAXPGS=393216
# a 46B beéri kevesebbel is:
#options SHMMAXPGS=262144
```

A saposcol nem indul

A saposcol (4.6D verzió) programmal akad néhány probléma. Az SAP® rendszer az saposcol segítségével próbál adatokat gyûjteni a rendszer teljesítményérõl. Mivel ez a program nem feltétlenül szükséges az SAP® rendszer mûködéséhez, ez a probléma nem tekinthetõ komolynak. A korábbi (4.6B) verziókban ugyan mûködik, de semmilyen adatot nem képes begyûjteni (mivel a legtöbb hívás, például a processzorhasználat függvénye, egyszerûen csak nullát ad vissza).

10.8. Témák haladóknak

Ha kíváncsiak vagyunk a Linux emuláció mûködésére, olvassuk el ezt a szakaszt. Az itt leírtak leginkább Terry Lambert (tlambert@primenet.com) FreeBSD chat levelezési lista címére írt levele nyomán kerülnek bemutatásra (Az üzenet azonosítója: <199906020108.SAA07001@usr09.primenet.com>).

10.8.1. Hogyan mûködik?

A FreeBSD rendelkezik egy ún. "végrehajtási osztály betöltõvel" (execution class loader). Ez lényegében a execve(2) rendszerhívás alatt meghúzódó absztrakciós réteg.

A FreeBSD-nek a #! karaktersorozat hatására parancsértelmezõk vagy a hozzájuk tartozó szkriptek betöltésére utasító biztonsági betöltõ helyett van egy listája az alkalmas betöltõkrõl.

A UNIX® rendszerek a hagyományok szerint egyetlen betöltõvel rendelkeznek, ami elõször megvizsgálja a betölteni kívánt állomány bûvös számát (ami általában az elsõ 4 vagy 8 byte) és ez alapján eldönti, hogy az adott formátum támogatott-e. Amennyiben ez így van, meghívja a betöltõt.

Ha a bináris típusa nem ismert a rendszer számára, akkor az execve(2) hívás hibával tér vissza, és a parancsértelmezõ próbálja meg a saját parancsaiként értelmezni.

Eddig ez volt az alapértelmezés, "akármilyen parancsértelmezõnk is volt".

Késõbb az sh(1) kódjába bekerült egy aprócska okosítás, amivel megnézte az állomány elsõ két karakterét, és ha az `:\n` volt, akkor a futtatáshoz maga helyett a csh(1) parancsértelmezõt hívta meg (ezt állítólag elõször a SCO csinálta).

A FreeBSD viszont végignézi a betöltõk teljes listáját, amiben a sor végén szerepel egy általános `! formátumú betöltõ. Ez az állomány futtatásához használatos értelmezõk kódját keresi, és ha egyet sem sikerül azonosítania, akkor a [.filename]`/bin/sh# programot indítja el.

A Linux ABI támogatását a FreeBSD úgy oldja meg, hogy elõször észleli az ELF bináris bûvös számát (ekkor még nem tesz különbséget a FreeBSD, Solaris™, Linux vagy más ELF típusú binárisokat használó operációs rendszerek közt).

Ezután az ELF formátum betöltõje az ELF állomány megjegyzéseket tároló szakaszában *bélyegek* (brand) után kutat, ami SVR4 és Solaris™ ELF binárisok esetén nem létezik.

A Linux binárisokat mûködésükhöz a brandelf(1) segítségével `Linux` típusúnak kell *megbélyegezni*:

```
# brandelf -t Linux állomány
```

Miután ezt megcsináltuk, az ELF betöltõ észre fogja venni az állomány `Linux` típusát.

Mikor az ELF betöltõ észleli, hogy az állomány `Linux` típusú, kicseréli egy mutató értékét a `proc` struktúrában. Minden rendszerhívás ezen a mutatón keresztül érhetõ el (a hagyományos UNIX® rendszerekben ez a rendszerhívásokat tartalmazó `sysent[]` struktúratömb). Emellett a frissen elindított program szoftveres megszakításait tartalmazó tömbjéhez beállítja a speciális jelzések kezelését, valamint a Linux modul által végzett néhány további (kisebb) javítást.

A Linux rendszerhívásokat tartalmazó tömb többek közt tartalmazza a `sysent[]` bejegyzések egy listáját, amelyek címei a rendszermag Linux moduljára mutatnak.

Amikor a Linux bináris hív egy rendszerhívást, a hozzá tartozó szoftveres megszakítás kódja a `proc` struktúrából a neki megfelelõ rendszerhívás kódját hivatkozza, így FreeBSD rendszerhívás belépési pontja helyett a Linuxét kapja meg.

Ráadásul Linux módban a különbözõ állományok hivatkozásai is *átirányítódnak*. Ez lényegében olyan, mint amit az állományrendszerek csatlakoztatásánál a `union` beállítás csinál (ami *nem* egyezik meg az `unionfs` állományrendszerrel!). Ilyenkor az állományokat elõször a /compat/linux/eredeti-hely könyvtárában keresi, és *majd* ha ott nem találja, csak akkor kezdi el keresni az /eredeti-hely ponton. Ezzel oldhatjuk meg, hogy más binárisok futtatását igénylõ binárisok is képesek legyenek rendesen mûködni (például így az egész linuxos eszköztár tud futni a Linux ABI-n keresztül). Egyúttal arra is utal, hogy ha a Linux binárisok számára nem áll

rendelkezésre a megfelelõ bináris, akkor FreeBSD binárisokat is el tudnak indítani. Ha a uname(1) programot pedig bemásoljuk a /compat/linux könyvtáron belülre, akkor a Linux binárisok képtelenek lesznek megmondani, hogy nem Linux alatt futnak.

Így lényegében egy Linux magot találunk a FreeBSD rendszermagjában. A benne megtalálható különbözõ szolgáltatásokat megvalósító függvények: az állománymûveletek, a virtuális memória kezelése, a jelzések küldése és System V típusú folyamatok közti kommunikáció stb. megegyeznek a FreeBSD és a Linux hívásai esetén egyaránt. Egyetlen eltérés, hogy a FreeBSD binárisok a FreeBSD *segédfüggvényein* (glue function), a Linux binárisok pedig a Linux segédfüggvényein keresztül férnek hozzájuk (a legelsõ operációs rendszerek tulajdonképpen csak a saját segédfüggvényeiket tartalmazták: a hívást kezdeményezõ program `proc` struktúrájában a függvények dinamikusan beállított címe helyett egy globális `sysent[]` struktúratömbben tárolták a meghívható függvényeket).

Melyik közülük a FreeBSD natív ABI-ja? Ez teljesen lényegtelen. Alapvetõen az egyetlen különbség csupán annyi (pillanatnyilag, de ez a jövõben még változhat, valószínûleg hamarosan), hogy a FreeBSD *segédfüggvényei* statikusan megtalálhatóak a rendszermagban, míg a Linux *segédfüggvényei* egyaránt elérhetõek modulból vagy statikus linkeléssel.

Na igen, de akkor ez most emuláció? Nem. Ez egy ABI, nem emuláció. Itt szó sincs emulátorról (ahogy szimulátorról sincs).

De akkor mégis miért hívják ezt sokszor "Linux emulációnak"? Hát hogy nehezebb legyen eladni a FreeBSD-t! Komolyra fordítva a szót: ennek a kezdeti változata akkoriban született meg, amikor erre még nem volt rendes szó. Nem mondhattuk, hogy a FreeBSD befordítás vagy egy modul betöltése nélkül képes lett volna Linux binárisokat futtatni, ezért valamilyen módon meg kellett neveznünk az ilyenkor betöltött kódot - ebbõl lett "a Linux emulátor".

Part III: Rendszeradminisztráció

A FreeBSD kézikönyv fennmaradó fejezeteiben a FreeBSD rendszerek adminisztrációjának különbözõ aspektusait mutatjuk be. Mindegyik fejezet elején megtudhatjuk mit is fogunk megismerni a fejezet elolvasása során, illetve arról is információkat kapunk, hogy mivel kell már tisztában lennünk a tárgyalt anyag feldolgozásához.

Ezeket a fejezeteket annak érdekében alakítottuk ki, hogy az adott témákban ismereteket adjunk. Nincs köztük semmilyen sorrendi kötöttség, sõt, ezeket egyáltalán nem is szükséges elolvasni a FreeBSD alapvetõ használatához.

Chapter 11. Beállítás és finomhangolás

11.1. Áttekintés

A FreeBSD egyik fontos szempontja a rendszer megfelelõ beállítása, aminek segítségével elkerülhetjük a késõbbi frissítések során keletkezõ kellemetlenségeket. Ez a fejezet a FreeBSD beállítási folyamatából kíván minél többet bemutatni, köztük a FreeBSD rendszerek finomhangolására szánt paramétereket.

A fejezet elolvasása során megismerjük:

- hogyan dolgozzunk hatékonyan az állományrendszerekkel és a lapozóállományokkal;
- az rc.conf beállításának alapjait és a /usr/local/etc/rc.d könyvtárban található indítási rendszert;
- hogyan állítsunk be és próbáljunk ki egy hálózati kártyát;
- hogyan állítsunk be virtuális címeket a hálózati eszközeinken;
- hogyan használjuk az /etc könyvtárban megtalálható különféle konfigurációs állományokat;
- hogyan hangoljuk a FreeBSD mûködését a `sysctl` változóinak segítségével;
- hogyan hangoljuk a lemezek teljesítményét és módosítsuk a rendszermag korlátozásait.

A fejezet elolvasásához ajánlott:

- a UNIX® és a FreeBSD alapjainak megértése (A UNIX alapjai);
- a rendszermag beállításához és fordításához kötõdõ alapok ismerete (A FreeBSD rendszermag testreszabása).

11.2. Kezdeti beállítások

11.2.1. A partíciók kiosztása

Alappartíciók

Amikor a bsdlabel(8) vagy a sysinstall(8) segítségével állományrendszereket telepítünk, nem szabad figyelmen kívül hagynunk a tényt, hogy a merevlemezes egységekben a külsõ sávokból gyorsabban lehet hozzáférni az adatokhoz, mint a belsõkbõl. Emiatt a kisebb és gyakrabban elérni kívánt állományrendszereket a meghajtó lemezének külsejéhez közel kell létrehozni, míg például a /usr partícióhoz hasonló nagyobb partíciókat annak belsõ része felé. A partíciókat a következõ sorrendben érdemes kialakítani: gyökér (rendszerindító), lapozóállomány, /var és /usr.

A /var méretének tükröznie kell a számítógép szándékolt használatát. A /var partíción foglalnak helyet a felhasználók postaládái, a naplóállományok és a nyomtatási sorok. A postaládák és a naplóállományok egészen váratlan mértékben is képesek megnövekedni attól függõen, hogy mennyi felhasználónk van a rendszerben és hogy mekkora naplókat tartunk meg. Itt a legtöbb felhasználónak soha nem lesz szüksége egy gigabyte-nál több helyre.

Bizonyos esetekben a /var/tmp könyvtárban azért ennél több tárterület

> szükségeltetik. Amikor a pkg_add(1) segítségével egy friss szoftvert telepítünk a rendszerünkre, akkor a program a /var/tmp könyvtárba tömöríti ki a hozzá tartozó csomag tartalmát. Ezért a nagyobb szoftvercsomagok, mint például a Firefox vagy az OpenOffice esetén gondok merülhetnek fel, ha nem rendelkezünk elegendõ szabad területtel a /var/tmp könyvtárban.

A /usr partíció tartalmaz számos, a rendszer mûködéséhez elengedhetetlenül fontos állományt, többek közt a portok gyûjteményét (ajánlott, lásd ports(7)) és a forráskódot (választható). A portok és az alaprendszer forrásai telepítés során választhatóak, de telepítésük esetén akkor ezen a partíción legalább két gigabyte-nyi hely ajánlott.

Vegyük figyelembe a tárbeli igényeket, amikor megválasztjuk a partíciók méretét. Igen kellemetlen lehet, amikor úgy futunk ki az egyik partíción a szabad helyből, hogy a másikat alig használjuk.

> Egyes felhasználók szerint elõfordulhat, hogy a sysinstall(8) `Auto-defaults` opciója a /var és / partíciók méretét túl kicsire választja. Particionáljunk okosan és nagylelkûen!

A lapozóállomány partíciója

Általános szabály, hogy a lapozóállományt tároló partíció mérete legyen a rendszer fizikai memóriájának (RAM) kétszerese. Például, ha a számítógépünk 128 megabyte memóriával rendelkezik, akkor a lapozóállomány méretének 256 megabyte-nak kell lennie. Az ennél kevesebb memóriát maguknak tudó rendszerek több lapozóállománnyal jobban teljesítenek. 256 megabyte-nál kevesebb lapozóállományt semmiképpen sem ajánlunk, és inkább a fizikai memóriát érdemes bõvítenünk. A rendszermag virtuális memóriát kezelõ lapozási algoritmusait úgy állították be, hogy abban az esetben teljesítsenek a legjobban, ha a lapozóállomány mérete legalább kétszerese a központi memória mennyiségének. A túl kicsi lapozóállomány beállítása rontja a virtuális memória lapkeresési rutinjának hatékonyságát és a memória bõvítése esetén még további gondokat is okozhat.

A több SCSI-lemezzel (vagy a különbözõ vezérlõkre csatlakoztatott több IDE-lemezzel) bíró nagyobb rendszerek esetében érdemes minden egyes (de legfeljebb négy) meghajtóra beállítani lapozóállományt. A lapozóállományoknak közel azonos méretûnek kell lenniük. A rendszermag tetszõleges méretûeket képes kezelni, azonban a belsejében alkalmazott adatszerkezetek a legnagyobb lapozóállomány méretének négyszereséig képesek növekedni. Ha a lapozóállományokat nagyjából ugyanazon a méreten tartjuk, akkor a rendszermag képes lesz a lapozáshoz felhasznált területet optimálisan elosztani a lemezek között. A nagyobb lapozóállományok használata még akkor is jól jön, ha nem is használjuk annyira. Segítségével sokkal könnyebben talpra tudunk állni egy elszabadult program tombolásából, és nem kell rögtön újraindítanunk a rendszert.

Miért particionáljunk?

Egyes felhasználók úgy gondolják, hogy egyetlen nagyobb méretû partíció mindenre megfelel, ám ez a gondolat több okból is helytelennek tekinthetõ. Elõször is, minden egyes partíciónak eltér a mûködési jellemzõje, és különválasztásukkal lehetõvé válik az állományrendszerek megfelelõ behangolása. Például a rendszerindításhoz használt és a /usr partíciókat többségében csak

olvasásra használják, és nem sokat írnak rájuk. Eközben a /var és /var/tmp könyvtárakban zajlik az írások és olvasások túlnyomó része.

A rendszer megfelelõ felosztásával a kisebb, intenzívebben írt partíciókon megjelenõ töredezettség nem szivárog át a többségében csak olvasásra használt partíciókra. Ha a sokat írt partíciókat közel tartjuk a lemez széléhez, akkor azokon a partíciókon növekszik az I/O teljesítménye, ahol az a leggyakrabban megjelenik. Mivel mostanság az I/O teljesítményére inkább a nagyobb partíciók esetén van szükség, azzal nem érünk el ebben különösebb mértékû növekedést, ha a /var partíciót a lemez szélére toljuk. Befejezésképpen hozzátesszük, hogy ennek vannak biztonsági megfontolásai is. Egy kisebb és takarosabb rendszerindító partíció, ami többnyire írásvédett, nagyobb eséllyel él túl egy csúfos rendszerösszeomlást.

11.3. A mag beállítása

A rendszer beállításaira vonatkozó információk központi lelõhelye az /etc/rc.conf állomány. Ez az állomány tartalmazza a beállításokra vonatkozó adatok széles körét, amelyet elsõsorban a rendszer indulása során a rendszer beállítására használnak. Erre a neve is utal: ez az rc* állományok konfigurációs állománya.

A rendszergazda az rc.conf állományban tudja felülbírálni az /etc/defaults/rc.conf állományban szereplõ alapértelmezett beállításokat. Az alapértelmezéseket tartalmazó állományt nem szabad közvetlenül átmásolni az /etc könyvtárba, hiszen alapértelmezett értékeket tartalmaz, nem pedig mintákat. Minden rendszerfüggõ beállítást magában az rc.conf állományban kell elvégezni.

Számos stratégia létezik a tömegesen adminisztrált számítógépeknél a közös és rendszerfüggõ beállítások különválasztására, ezáltal a karbantartási költségek csökkentésére. A közös beállításokat ajánlott egy másik helyre, például az /etc/rc.conf.site állományba rakni, majd hivatkozni erre a kizárólag csak rendszerfüggõ információkat tartalmazó /etc/rc.conf állományból.

Mivel az rc.conf állományt az sh(1) dolgozza fel, ezt elég könnyen el tudjuk érni. Például:

- rc.conf:

```
. /etc/rc.conf.site
hostname="node15.example.com"
network_interfaces="fxp0 lo0"
ifconfig_fxp0="inet 10.1.1.1"
```

- rc.conf.site:

```
defaultrouter="10.1.1.254"
saver="daemon"
blanktime="100"
```

Az rc.conf.site állomány ezt követõen az `rsync` parancs használatával már szétszórható a rendszerben, miközben az rc.conf állomány mindenkinél egyedi marad.

Ha a rendszert a sysinstall(8) vagy a `make world` használatával frissítjük, akkor az rc.conf tartalma nem íródik felül, így a rendszer beállításairól szóló adatok nem vesznek el.

11.4. Az alkalmazások beállítása

A telepített alkalmazások általában saját konfigurációs állományokkal, azok pedig saját formátummal stb. rendelkeznek. Fontos, hogy ezeket az állományokat az alaprendszertõl elkülönítve tároljuk, ezáltal a csomagkezelõ eszközök könnyen rájuk tudjanak találni és dolgozni velük.

Ezeket az állományokat általában a /usr/local/etc könyvtárban találjuk meg. Amennyiben egy alkalmazáshoz több konfigurációs állomány is tartozik, akkor ahhoz ezen belül egy külön alkönyvtár jön létre.

Normális esetben, amikor egy portot vagy csomagot telepítünk, minta konfigurációs állományokat is kapunk. Ezek nevében többnyire a .default utótag szerepel. Ha még nincs konfigurációs állomány az adott alkalmazáshoz, akkor a .default jelzésû állományokból ez létrehozható.

Példaképpen most tekintsük a /usr/local/etc/apache könyvtár tartalmát:

```
-rw-r--r--  1 root  wheel   2184 May 20  1998 access.conf
-rw-r--r--  1 root  wheel   2184 May 20  1998 access.conf.default
-rw-r--r--  1 root  wheel   9555 May 20  1998 httpd.conf
-rw-r--r--  1 root  wheel   9555 May 20  1998 httpd.conf.default
-rw-r--r--  1 root  wheel  12205 May 20  1998 magic
-rw-r--r--  1 root  wheel  12205 May 20  1998 magic.default
-rw-r--r--  1 root  wheel   2700 May 20  1998 mime.types
-rw-r--r--  1 root  wheel   2700 May 20  1998 mime.types.default
-rw-r--r--  1 root  wheel   7980 May 20  1998 srm.conf
-rw-r--r--  1 root  wheel   7933 May 20  1998 srm.conf.default
```

Az állományok mérete jól mutatja, hogy csak az srm.conf változott meg. Az Apache késõbbi frissítései ezt az állományt nem fogják felülírni.

11.5. Szolgáltatások indítása

A felhasználók közül sokan választják a FreeBSD Portgyûjteményében található külsõ szoftverek telepítését. A telepített szoftvert ilyenkor gyakran úgy kell beállítani, hogy a rendszer indulásával együtt induljon. Az olyan szolgáltatások, mint például a mail/postfix vagy a www/apache13 csupán két olyan szoftvercsomag, amelyet a rendszerrel együtt kell elindítani. Ebben a szakaszban a külsõ szoftverek indítására használatos eljárásokkal foglalkozunk.

A FreeBSD-ben megjelenõ legtöbb szolgáltatás, mint például a cron(8), a rendszerindító szkripteken keresztül kel életre. Habár ezek a szkriptek a FreeBSD egyes verziói vagy az egyes gyártók esetén különbözhetnek, azonban az mindegyikükben közös, hogy az elindításukra vonatkozó beállítások egyszerû indítószkriptekkel adhatóak meg.

11.5.1. Az alkalmazások részletesebb beállítása

Most miután a FreeBSD rendelkezik egy rc.d könyvtárral, az alkalmazások indításának beállítása is könnyebbé és ügyesebbé vált. Az rc.d mûködésérõl szóló szakaszban megismert kulcsszavak segítségével az alkalmazások mostantól kezdve a többi szolgáltatás, például a DNS után indulnak el, és az rc.conf állományon keresztül a szkriptekbe huzalozottak helyett most már tetszõleges paramétereket is átadhatunk stb. Egy egyszerû szkript ehhez hasonlóan néz ki:

```
#!/bin/sh
#
# PROVIDE: utility
# REQUIRE: DAEMON
# KEYWORD: shutdown

. /etc/rc.subr

name=utility
rcvar=utility_enable

command="/usr/local/sbin/utility"

load_rc_config $name

#
# NE VÁLTOZTASSUK MEG AZ ITT LÉVõ ALAPÉRTELMEZÉSEKET,
# INKÁBB AZ /etc/rc.conf ÁLLOMÁNYBAN ÁLLÍTSUK BE EZEKET
#
utility_enable=${utility_enable-"NO"}
pidfile=${utility_pidfile-"/var/run/utility.pid"}

run_rc_command "$1"
```

Ez a szkript gondoskodik arról, hogy a utility nevû alkalmazás a `DAEMON` szolgáltatás után induljon el. Emellett még felkínál egy módszert a PID avagy futó programok azonosítójának beállítására és nyomonkövetésére is.

Ezt követõen az /etc/rc.conf állományból az alkalmazás elindítható az alábbi sor hozzáadásával:

```
utility_enable="YES"
```

Ez a módszer megkönnyíti a parancssorban átadott paraméterek módosítását, az /etc/rc.subr állományban szereplõ alapértelmezett függvények használatát, az rcorder(8) segédprogrammal szembeni kompatibilitást és az rc.conf állomány könnyebb beállítását.

11.5.2. Szolgáltatások indítása szolgáltatásokkal

Más szolgáltatások, mint például a POP3 vagy IMAP szerverek démonai stb. az inetd(8) segítségével indíthatóak el. Ez a Portgyûjteménybõl telepített szolgáltatások esetén magával vonja az adott

segédprogram felvételét vagy a hozzá tartozó sor engedélyezését az /etc/inetd.conf állományban. Az inetd mûködésével és annak beállításával mélyrehatóbban az inetd szakasza foglalkozik.

A legtöbb esetben a cron(8) démon használata kézenfekvõ a rendszerszintû szolgáltatások elindításában. Ez a megközelítés számos elõnyt tartogat, mivel a `cron` ezeket a programokat a felhasználó crontab állománya alapján futtatja. Ezzel a mezei felhasználók számára is lehetõvé válik, hogy elindítsanak és karbantartsanak alkalmazásokat.

A `cron` segédprogramnak van egy olyan speciális lehetõsége, hogy az idõ helyett a `@reboot` értéket adhatjuk meg. Ennek hatására a feladat a cron(8) indításával együtt fut le, tehát megszokott esetben a rendszer indítása során.

11.6. A `cron` segédprogram beállítása

A cron(8) a FreeBSD egyik leghasznosabb segédprogramja. A `cron` segédprogram a háttérben fut és folyamatosan figyeli az /etc/crontab állományt. Emellett a `cron` új crontab állományok után kutatva folyamatosan ellenõrzi a /var/cron/tabs könyvtárat. Ezek a crontab állományok olyan feladatokról tárolnak adatokat, amelyeket a `cron` programnak egy adott pillanatban el kell végeznie.

A `cron` a konfigurációs állományok két külön fajtáját, a rendszer- és felhasználói crontabokat használja. A két típus között levõ egyetlen különbség a hatodik mezõben található. A rendszerszintû crontabok esetében a hatodik mezõ annak a felhasználónak a nevét tartalmazza, amivel a program fut. Ezzel a rendszer szintjén mûködõ crontaboknak megadatott az a képesség, hogy tetszõleges felhasználó nevében futtassanak programokat. A felhasználók crontabjaiban a hatodik mezõ a futtatandó parancsot tartalmazza, és ilyenkor az összes parancs a crontabot létrehozó felhasználó nevében hajtódik végre. Ez utóbbi egy fontos biztonsági jellemzõ.

A felhasználói crontabok lehetõvé teszik az egyes felhasználók számára, hogy a `root` felhasználó jogosultságai nélkül képesek legyenek feladatokat ütemezni, ugyanis a felhasználóhoz tartozó crontabban szereplõ parancsok mindegyike a tulajdonosának engedélyeivel fut.

Az átlagos felhasználókhoz hasonlóan a `root` felhasználónak is lehet crontabja, ami nem ugyanaz, mint az /etc/crontab (a rendszer saját crontab állománya). De mivel a rendszernek külön crontabja van, ezért a `root` felhasználónak nem kell külön crontabot létrehozni.

Vessünk egy pillanatást az /etc/crontab (a rendszer crontabjának) tartalmára:

```
# /etc/crontab - a root crontabja FreeBSD alatt
#
# $FreeBSD: src/etc/crontab,v 1.32 2002/11/22 16:13:39 tom Exp $
#①
#
SHELL=/bin/sh
PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin ②
HOME=/var/log
#
```

```
#
#minute hour    day month   wday    who command ③
#
#
*/5 *   *   *   *   root    /usr/libexec/atrun ④
```

① A FreeBSD legtöbb konfigurációs állományához hasonlóan itt is a # jelöli a megjegyzéseket. Az ilyen megjegyzések remekül használhatóak annak feljegyzésére, hogy mit és miért akarunk futtatni. A megjegyzések azonban nem szerepelhetnek a paranccsal egy sorban, mivel máskülönben a parancs részeként kerülnek értelmezésre. Tehát mindig új sorba kell raknunk ezeket. Az üres sorokat a program nem veszi figyelembe.

② Elõször is meg kell adnunk egy környezetet. Az egyenlõség (=) karakter használatos a környezeti beállítások meghatározására, ahogy mindezt az itteni példában is tapasztalhatjuk a SHELL, PATH és HOME értékek esetében. Ha nem adunk meg mást, akkor a cron az alapértelmezés szerinti sh parancsértelmezõt használja. Ha nem adjuk meg a PATH változó értékét, akkor minden állományra abszolút elérési úttal kell hivatkoznunk, mivel ennek nincs alapértelmezett értéke. Ha nem definiáljuk a HOME változó értékét, akkor a cron a parancshoz tartozó felhasználó könyvtárából fog dolgozni.

③ Ez a sor írja le a megadható hét mezõt. Az itt szereplõ értékek a minute (perc), hour (óra), mday (a hónap napja), month (hónap), wday (a hét napja), who (ki) és command (mit). A mezõk szinte maguktól értetõdnek. A minute egy órán belül adja meg azokat a perceket, amikor az adott parancsot le kell futtatni. A hour hasonló a minute beállításhoz, csak az itt szereplõ értékét órákban kell értelmezni. Az mday a hónap napjaiban számol. A month hasonló a minute és hour opciókhoz, de ez hónapot jelöl. A wday a hét egy napját jelzi. Ezeknek a mezõknek numerikus, valamint a huszonnégy órás idõformátumnak megfelelõ értékeket kell tartalmazniuk. A who mezõ, a többiektõl eltérõ módon, csak az /etc/crontab állományban jelenik meg. Ez a mezõ adja meg, hogy a parancsot milyen felhasználóval kell futtatni. Ez az opció nem jelenik meg a felhasználók saját crontab állományainak telepítésekor. A sor végén láthatjuk még a command oszlopot is. Ez az utolsó mezõ, és ide kerül a végrehajtandó parancs.

④ Ez az utolsó sor a fentebb tárgyalt értékeket határozza meg. Észrevehetjük, hogy a sor egy */5 alakú felírással kezdõdik, amelyet további karakterek követnek. A * karakterek jelentése "elsõ-utolsó", ami arra utal, hogy *mindig*. Ennek megfelelõen úgy értelmezhetjük ezt a sort, hogy a root felhasználóval le kell futtatni az atrun parancsot minden ötödik percben, függetlenül attól, hogy milyen nap vagy hónap van. Az atrun parancsról részletesebban az atrun(8) man oldalán kapunk felvilágosítást.Az itt szereplõ parancsoknak tetszõleges mennyiségû paraméter adható át, azonban a több soron keresztül átívelõ parancsok tördelését a sor végén a "\" karakterrel kell jelezni.

Ez mindegyik crontab állomány alapbeállítása, habár ettõl általában egy dologban eltérnek. A hatodik mezõ, ahol a felhasználót adtuk meg, csak a rendszer /etc/crontab állományában jelenik meg. Ez a mezõ a felhasználók crontab állományaiból kimarad.

11.6.1. Egy crontab telepítése

Nem kötelezõ az itt ismertetésre kerülõ módon szerkeszteni vagy telepíteni a rendszer crontabját. Egyszerûen nyissuk meg a kedvenc szövegszerkesztõnkkel, és a cron segédprogram majd észreveszi, hogy az állomány megváltozott, majd ennek

> megfelelõen neki is lát a módosított változat használatának. Errõl a GYIK-ban (angolul) többet is megtudhatunk.

Egy frissen készített felhasználói crontab telepítéséhez elõször a kedvenc szövegszerkesztõnk segítségével létre kell hoznunk a megfelelõ formátumú állományt, majd használnunk a `crontab` segédprogramot. Ennek általános alakja:

```
% crontab crontab_állomány
```

Ebben a példában a crontab_állomány a korábban létrehozott crontab neve lesz.

Lehetõségünk van lekérdezni a telepített crontab állományokat: egyszerûen adjuk át a `-l` kapcsolót a `crontab` parancsnak, és nézzük meg, mit ad vissza.

A `crontab -e` használata olyan felhasználók számára ajánlott, akik sablon alkalmazása nélkül szeretnének teljesen maguktól megírni egy crontab állományt. Ennek hatására a kiválasztott szövegszerkesztõ egy üres állományt kap. Miután ezt az állományt elmentettük, a `crontab` programmal magától telepítésre kerül.

Ha a késõbbiekben törölni akarjuk a felhasználónkhoz tartozó crontab állományt, akkor erre a célra használjuk a `crontab -r` kapcsolóját.

11.7. Az rc használata FreeBSD alatt

A rendszer indítására a FreeBSD 2002-ben átvette a NetBSD rc.d rendszerét. Ezt a felhasználók könnyen felismerhetik az /etc/rc.d könyvtárban található állományokról. A legtöbbjük olyan alapvetõ szolgáltatás, amelyet a `start`, `stop` és `restart` paraméterekkel lehet vezérelni. Például az sshd(8) az alábbi paranccsal indítható újra:

```
# /etc/rc.d/sshd restart
```

Ez az eljárás hasonló a többi szolgáltatás esetén is. Természetesen ezek a szolgáltatások általában maguktól indulnak el a rendszer indítása során az rc.conf(5) állományban megadottak szerint. Például ha a rendszerünk indulásakor szeretnénk aktiválni a hálózati címfordítással foglalatoskodó démont, akkor csak adjuk hozzá az /etc/rc.conf állományhoz a következõ sort:

```
natd_enable="YES"
```

Amennyiben a `natd_enable="NO"` sor már szerepel benne, akkor egyszerûen írjuk át a `NO` értéket `YES` -re. Ezután az rc szkriptek a rendszer következõ indításakor a lentieknek megfelelõen automatikusan elindítják a hozzá tartozó szolgáltatásokat is.

Mivel az rc.d rendszert elsõsorban arra használják, hogy szolgáltatásokat indítsanak el vagy állítsanak le az operációs rendszerrel együtt, a szabványos `start`, `stop` és `restart` paraméterek csak abban az esetben látják el a feladatukat, ha a nekik megfelelõ változókat beállítottuk az /etc/rc.conf állományban. Tehát például az `sshd restart` csak abban az esetben fog bármit is csinálni, ha az

/etc/rc.conf állományban az `sshd_enable` változót a `YES` értékre állítottuk. Ha az /etc/rc.conf beállításaitól függetlenül kívánunk egy szolgáltatásnak `start`, `stop` vagy `restart` parancsot adni, akkor elé kell tennünk egy "one" szót. Például ha az `sshd` szolgáltatás újraindításához az /etc/rc.conf tartalmát figyelmen kívül akarjuk hagyni, akkor ezt a parancsot kell kiadnunk:

```
# /etc/rc.d/sshd onerestart
```

Könnyen ellenõrizni tudjuk, hogy az adott szolgáltatás az /etc/rc.conf részérõl engedélyezett-e, ha a neki megfelelõ rc.d szkriptnek megadjuk az `rcvar` paramétert. Ennek segítségével például a rendszergazda így képes ellenõrizni, hogy az `sshd` szolgáltatást engedélyezi-e az /etc/rc.conf:

```
# /etc/rc.d/sshd rcvar
# sshd
$sshd_enable=YES
```

A második sor (`# sshd`) az `sshd` parancs kimenete, nem pedig a `root` parancssora.

A `status` paraméterrel kideríthetjük, hogy egy szolgáltatás aktív-e. Ezzel például így tudjuk ellenõrizni az `sshd` szolgáltatás mûködését:

```
# /etc/rc.d/sshd status
sshd is running as pid 433.
```

Az üzenet:

```
Az sshd a 433-as azonosítóval fut.
```

Bizonyos esetekben a `reload` paraméter használatával lehetõségünk van a szolgáltatások újraindítására is. Ilyenkor a rendszer megpróbál egy olyan jelzést küldeni a szolgáltatásnak, amivel a konfigurációs állományainak újraolvasását kéri. A legtöbbször lényegében ez a `SIGHUP` jelzés kiküldését rejti magában. Ez a lehetõség azonban nem mindegyik szolgáltatás esetén érhetõ el.

Az rc.d rendszer nem csupán hálózati szolgáltatások esetén használatos, hanem nagyrészben hozzájárul a rendszer indításához is. Erre vegyük példának a bgfsck állományt. Amikor ez a szkript lefut, a következõ üzenetet jeleníti meg:

```
Starting background file system checks in 60 seconds.
```

Az üzenet fordítása:

```
A háttérben 60 másodperc múlva megkezdõdik az állományrendszerek ellenõrzése.
```

Ennek megfelelõen tehát ezt az állományt az állományrendszerek háttérben folyó ellenõrzésére

használják, ami pedig a rendszer indítása során fut le.

Számos rendszerszolgáltatás igényel a mûködéséhez további szolgáltatásokat. Például a NIS és más egyéb távoli eljáráshíváson alapú szolgáltatások egészen addig nem képesek elindulni, amíg az `rpcbind` (portmapper) szolgáltatást el nem indítjuk. Az ilyen jellegû gondok feloldására az indítószkriptek elején levõ megjegyzésekben található egy kevés metainformáció a szkript mûködéséhez szükséges elemekre (függõségeire) vonatkozóan. A rendszer indítása közben az rcorder(8) nevû program képes a megjegyzések közt ezeket az információkat feldolgozni és ebbõl megállapítani, hogy a függõségi viszonyok betartásával milyen sorrendben kell elindítani a rendszer által felkínált szolgáltatásokat.

Ehhez a következõ kulcsszavakat kell megadni az egyes indító szkriptek elején (az rc.subr(8) így tudja "engedélyezni" az indító szkriptet):

- `PROVIDE`: segítségével megmondjuk, hogy ez az állomány milyen szolgáltatásokat nyújt.

A következõ kulcsszavak az egyes indítóállományok elején szerepelhetnek. Nem kell feltétlenül használnunk ezeket, de velük az rcorder(8) munkáját segíthetjük:

- `REQUIRE`: felsoroljuk azokat a szolgáltatásokat, amelyek a futásához kellenek. Az állomány tehát az itt megadott szolgáltatások *után* fog lefutni.
- `BEFORE`: felsoroljuk azokat a szolgáltatásokat, amelyek *elõtt* futtatni kell ezt az állományt.

Az indító szkriptekben a kulcsszavak ügyes megválasztásával a rendszergazda nagyon finoman képes az indításkor végrehajtódó szkriptek sorrendjét szabályozni és a többi UNIX® alapú operációs rendszerbõl ismert "futtatási szintek" használata nélkül vezérelni a rendszerben megjelenõ szolgáltatásokat.

Az rc.d rendszerrõl bõvebben az rc(8) és rc.subr(8) man oldalakon olvashatunk. Ha szeretnénk saját rc.d szkripteket írni vagy javítani a már meglévõkön, akkor ez a cikk (angolul) segítségünkre lehet.

11.8. A hálózati kártyák beállítása

Manapság már el sem tudunk képzelni számítógépet hálózati csatlakozás nélkül. A hálózati csatolókártyák hozzáadása és beállítása egy FreeBSD rendszergazda mindennapos feladata.

11.8.1. A megfelelõ meghajtóprogram felderítése

Mielõtt bárminek is nekikezdenénk, érdemes tisztában lennünk azzal, hogy a rendelkezésünkre álló kártya milyen típusú, milyen chipet használ és hogy PCI vagy ISA buszon csatlakozik-e. A FreeBSD a PCI és ISA csatolós kártyák széles spektrumát ismeri. Az egyes kiadásokhoz mellékelt "Hardware Compatibility List" (Hardverkompatibilitási lista) dokumentumokban tudjuk ellenõrizni, hogy a kártyákat ismeri a rendszer.

Miután meggyõzõdtünk róla, hogy a kártyánkat ismeri a rendszer, meg kell keresnünk a hozzá tartozó meghajtót. A /usr/src/sys/conf/NOTES és a /usr/src/sys/arch/conf/NOTES állományok tartalmazzák a hálózati kártyák meghajtóinak rövid leírását, benne a támogatott chipsetek és kártyák típusaival. Ha ez alapján nem tudjuk teljes biztossággal eldönteni, hogy melyik a számunkra megfelelõ meghajtó, nézzük meg a saját man oldalát. Ezen a man oldalon megtaláljuk az általa

ismert összes eszközt és a velük kapcsolatban elõforduló jellemzõ problémákat.

Ha egy elterjedt típust sikerült beszereznünk, akkor nem kell különösebben sokáig keresnünk a neki megfelelõ meghajtót. Az ismertebb hálózati kártyák meghajtói ugyanis alapból benne vannak a GENERIC rendszermagban, ezért a rendszer indítása során ehhez hasonlóan meg is jelennek a kártyák:

```
dc0: <82c169 PNIC 10/100BaseTX> port 0xa000-0xa0ff mem 0xd3800000-0xd38
000ff irq 15 at device 11.0 on pci0
miibus0: <MII bus> on dc0
bmtphy0: <BCM5201 10/100baseTX PHY> PHY 1 on miibus0
bmtphy0:  10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
dc0: Ethernet address: 00:a0:cc:da:da:da
dc0: [ITHREAD]
dc1: <82c169 PNIC 10/100BaseTX> port 0x9800-0x98ff mem 0xd3000000-0xd30
000ff irq 11 at device 12.0 on pci0
miibus1: <MII bus> on dc1
bmtphy1: <BCM5201 10/100baseTX PHY> PHY 1 on miibus1
bmtphy1:  10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
dc1: Ethernet address: 00:a0:cc:da:da:db
dc1: [ITHREAD]
```

Ebben a példában láthatunk is két olyan kártyát, amelyek a dc(4) meghajtót használják.

Ha a hálózati kártyánk meghajtója nem szerepel a GENERIC konfigurációban, akkor a mûködéséhez be kell tölteni a megfelelõ meghajtót. Ezt alapvetõen kétféleképpen érhetjük el:

- Ennek legegyszerûbb módja, ha a kldload(8) használatával alkalmanként vagy a /boot/loader.conf állományban a megfelelõ sor hozzáadásával a rendszer indításával együtt betöltjük a hálózati kártya meghajtójához tartozó modult. Nem mindegyik hálózati kártya meghajtója érhetõ el modul formájában. Erre konkrét például szolgálnak az ISA kártyákhoz tartozó modulok.
- Másik lehetõségünk, ha statikusan beépítjük a kártyánk támogatását a rendszermagba. A /usr/src/sys/conf/NOTES és az /usr/src/sys/arch/conf/NOTES állományok, valamint a meghajtóhoz tartozó man oldal elolvasásából megtudhatjuk a rendszermag beállításait tartalmazó állományban megadandó paramétereket. A rendszermag újrafordítását lásd A FreeBSD rendszermag testreszabása. Ha a rendszermag (GENERIC) az indulás során észlelte a kártyánkat, nem kell újat készítenünk.

A Windows® NDIS meghajtóinak használata

Sajnos még mindig sok olyan gyártó akad, akik a nyílt forrású közösség számára nem adják ki a meghajtóik mûködésének alapjait, mivel az ilyen adatokat szakmai titoknak tekintik. Ebbõl következik, hogy a FreeBSD és más operációs rendszerek fejlesztõi számára két választás marad: vagy a gyári meghajtók visszafejtésének hosszú és fájdalmas útján haladva fejlesztik ki a saját meghajtójukat, vagy pedig a Microsoft® Windows® platformra kiadott meghajtók binárisait hasznosítják. A legtöbb fejlesztõ, köztük a FreeBSD fejlesztõi is, ez utóbbi megközelítést választották.

Bill Paul (wpaul) jóvoltából a FreeBSD 5.3-RELEASE változatában megjelent a "Network Driver Interface Specification" (NDIS, avagy hálózati meghajtók szabványos felülete) "natív" támogatása. A FreeBSD NDISulator (másnéven Project Evil, a Gonosz terve) nevû komponense fog egy Windows®-os meghajtót és elhiteti vele, hogy a Windows® operációs rendszerrel kommunikál. Mivel az ndis(4) meghajtó Windows® binárisokat használ fel, ezért csak i386 és amd64 rendszerek esetén érhetõ el.

Az ndis(4) meghajtó leginkább a PCI, CardBus és PCMCIA csatolójú eszközök támogatására lett kitalálva, az USB eszközöket még nem ismeri.

Az NDISulator használatához három tényezõre van szükségünk:

1. A rendszermag forrása
2. a Windows® XP meghajtó binárisa (.SYS a kiterjesztése)
3. a Windows® XP meghajtó konfigurációs állománya (.INF a kiterjesztése)

Keressük meg az említett állományokat az adott kártyához. Ezeket általában a mellékelt CD-n vagy a gyártó honlapján találjuk meg. A most következõ példákban a W32DRIVER.SYS és a W32DRIVER.INF neveket fogjuk használni.

A Windows® i386 architektúrájú verziójához készült meghajtóprogramokat nem tudjuk a FreeBSD/amd64 verziójával használni. A mûködéshez amd64-re készült Windows®-os meghajtókra van szükség.

A következõ lépés a meghajtó binárisainak betölthetõ modulba fordítása. Ennek eléréséhez használjuk az ndisgen(8) parancsot a `root` felhasználóval:

```
# ndisgen /windowsos/meghajtó/W32DRIVER.INF /windowsos/meghajtó/W32DRIVER.SYS
```

Az ndisgen(8) egy interaktív segédprogram, amely mûködése közben még rákérdez néhány szükséges információra. Az aktuális könyvtárban létrehoz egy rendszermagmodult, amelyet az alábbi módon tudunk betölteni:

```
# kldload ./W32DRIVER_SYS.ko
```

Az elõállított modul mellé be kell töltenünk még az ndis.ko és az if_ndis.ko modulokat is. Ez általában minden olyan modul esetén megtörténik magától, amely függ az ndis(4) használatától. Kézileg a következõ parancsokkal tudjuk ezeket betölteni:

```
# kldload ndis
# kldload if_ndis
```

Itt az elsõ parancs betölti az NDIS miniport meghajtó burkolására szánt kódot, valamint a második a tényleges hálózati csatolófelületet.

Most pedig a dmesg(8) kimenetében ellenõrizzük, hogy történt-e valamilyen hiba a betöltés során.

Ha minden jól ment, akkor az alábbiakhoz hasonló kimenetet produkált:

```
ndis0: <Wireless-G PCI Adapter> mem 0xf4100000-0xf4101fff irq 3 at device 8.0 on pci1
ndis0: NDIS API version: 5.0
ndis0: Ethernet address: 0a:b1:2c:d3:4e:f5
ndis0: 11b rates: 1Mbps 2Mbps 5.5Mbps 11Mbps
ndis0: 11g rates: 6Mbps 9Mbps 12Mbps 18Mbps 36Mbps 48Mbps 54Mbps
```

Innentõl kezdve az ndis0 nevû eszközt úgy tudjuk használni, mint bármelyik más hálózati felületet (például dc0).

A többi modulhoz hasonló módon be tudjuk állítani, hogy a rendszer indulásával együtt betöltõdjenek az NDIS modulok. Ehhez elõször másoljuk az imént létrehozott modult, az W32DRIVER_SYS.ko állományt a /boot/modules könyvtárba. Ezután adjuk hozzá a következõ sort a /boot/loader.conf állomány tartalmához:

```
W32DRIVER_SYS_load="YES"
```

11.8.2. A hálózati kártya beállítása

Ahogy betöltõdött a megfelelõ meghajtó a hálózati kártyánkhoz, be is kell állítanunk a kártyát. A hálózati kártyák sok más dologgal együtt beállíthatóak a telepítés során a sysinstall segítségével.

A rendszerünkben beállított hálózati csatolófelületek megjelenítéséhez gépeljük be a következõ parancsot:

```
% ifconfig
dc0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=80008<VLAN_MTU,LINKSTATE>
        ether 00:a0:cc:da:da:da
        inet 192.168.1.3 netmask 0xffffff00 broadcast 192.168.1.255
        media: Ethernet autoselect (100baseTX <full-duplex>)
        status: active
dc1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=80008<VLAN_MTU,LINKSTATE>
        ether 00:a0:cc:da:da:db
        inet 10.0.0.1 netmask 0xffffff00 broadcast 10.0.0.255
        media: Ethernet 10baseT/UTP
        status: no carrier
plip0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> mtu 1500
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
        options=3<RXCSUM,TXCSUM>
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x4
        inet6 ::1 prefixlen 128
        inet 127.0.0.1 netmask 0xff000000
        nd6 options=8010<POINTOPOINT,MULTICAST> mtu 1500
```

Az elõbbi parancs kimenetében a következõ eszközök jelentek meg:

- dc0: az elsõ Ethernet felület
- dc1: a második Ethernet felület
- plilp0: a párhuzamos port felülete (amennyiben található párhuzamos port a számítógépben)
- lo0: a loopback eszköz

A FreeBSD a kártyához tartozó meghajtó nevével és egy sorszámmal azonosítja a rendszermag indulása során talált eszközöket. Például az sis2 a rendszerben található harmadik olyan eszköz, amely a sis(4) meghajtót használja.

A példában a dc0 eszköz aktív és mûködõképes. Ennek legfontosabb jelei:

1. Az `UP` szó mutatja, hogy a kártyát sikerült beállítani és készen áll a használatra.
2. A kártya internet (`inet`) címe (jelen esetünkben ez `192.168.1.3`).
3. Érvényes hálózati maszkkal rendelkezik (`netmask`, ahol a `0xffffff00` a `255.255.255.0` címnek felel meg).
4. Érvényes broadcast (üzenetszóró) címmel rendelkezik (ami itt most `192.168.1.255`).
5. A kártya MAC-címe (`ether`) `00:a0:cc:da:da:da`.
6. A hozzá tartozó fizikai eszköz kiválasztása automatikus (`media: Ethernet autoselect (100baseTX <full-duplex>)`). Láthatjuk, hogy a dc1 eszközt egy `10baseT/UTP` típusú fizikai eszközhöz állítottuk be. Az egyes meghajtókhoz tartozó fizikai módokról a nekik megfelelõ man oldalakon olvashatunk.
7. A kapcsolat állapota (`status`) `active` értékû, tehát van vonal. A dc1 esetén láthatjuk, hogy a `status: no carrier` (nincs vonal). Ez teljesen normálisnak tekinthetõ minden olyan esetben, amikor a kártyába még nem dugtunk Ethernet-kábelt.

Amennyiben az ifconfig(8) kimenete valami ilyesmi:

```
dc0: flags=8843<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500
            options=80008<VLAN_MTU,LINKSTATE>
            ether 00:a0:cc:da:da:da
            media: Ethernet autoselect (100baseTX <full-duplex>)
            status: active
```

akkor az arra utal, hogy a kártyát nem állítottuk be.

A kártya beállításához a `root` felhasználó jogosultságaira van szükségünk. A hálózati kártyák beállítása az ifconfig(8) segítségével elvégezhetõ parancssorból is, de a gép újraindításakor az így megadott értékek elvesznek. Ezért az /etc/rc.conf állományba kell felvennünk a hálózati kártyák érvényes beállításait.

A kedvenc szövegszerkesztõnkben nyissuk meg az /etc/rc.conf állományt. Minden egyes hálózati csatolóhoz fel kell vennünk benne egy sort, ennek megfelelõen most a példához tartozó módon az alábbiakat:

```
ifconfig_dc0="inet 192.168.1.3 netmask 255.255.255.0"
ifconfig_dc1="inet 10.0.0.1 netmask 255.255.255.0 media 10baseT/UTP"
```

A dc0 és dc1 neveket kell a rendszerünkben ténylegesen megtalálható eszközök neveire kicserélni, valamint megadni a nekik megfelelõ címeket. A kártya meghajtójának és az ifconfig(8) man oldalának elolvasásával kideríthetjük az itt megadható további beállításokat, valamint az rc.conf(5) man oldalán részletesebben megismerhetjük az /etc/rc.conf formai követelményeit.

Ha a telepítés során beállítottuk volna a hálózati kapcsolatokat, akkor tapasztalhatjuk, hogy egyes hálózati kártyák sorai itt már szerepelnek. Ellenõrizzük az /etc/rc.conf tartalmát, mielõtt bõvítenénk!

Mindezek mellett az /etc/hosts állományba is be kell írnunk a helyi hálózatunkon található különféle gépek neveit és IP-címeit, ha még nem szerepelnének ott. Errõl további részleteket a hosts(5) man oldalról és az /usr/shared/examples/etc/hosts állományból tudhatunk meg.

Ha a géppel szeretnénk majd csatlakozni az internetre, akkor ne felejtsük el manuálisan beállítani az alapértelmezett átjárót és a névfeloldáshoz szükséges kiszolgálót:

```
# echo 'defaultrouter="alapertelmezett_atjaro"' >> /etc/rc.conf
# echo 'nameserver DNS_kiszolgalo' >> /etc/resolv.conf
```

11.8.3. Tesztelés és hibaelhárítás

Miután az /etc/rc.conf állományban elvégeztük a szükséges változtatásokat, érdemes újraindítanunk a rendszerünket. Ennek révén érvényesítjük a csatolófelületekkel kapcsolatos változtatásainkat és ellenõrizzük, hogy így a rendszer mindenféle hibaüzenet nélkül képes elindulni. A másik lehetõség, ha csak magát a hálózati alrendszer konfigurációját indítjuk el újra:

```
# /etc/rc.d/netif restart
```

Ha az /etc/rc.conf állományban már beállítottuk az alapértelmezett átjárót, akkor elegendõ csupán ez a parancs:

```
# /etc/rc.d/routing restart
```

Ahogy újrakonfiguráltuk a hálózati alrendszert, ki is tudjuk próbálni a hálózati felületeket.

Az Ethernet kártyák tesztelése

Az Ethernet kártyák helyes beállításának vizsgálatához két dolgot kell kipróbálnunk. Elõször is pingeljük magát a felületet, majd ezután pingeljünk meg a helyi hálózaton egy másik számítógépet.

Elsõként tehát próbáljuk meg a helyi felületet:

```
% ping -c5 192.168.1.3
PING 192.168.1.3 (192.168.1.3): 56 data bytes
64 bytes from 192.168.1.3: icmp_seq=0 ttl=64 time=0.082 ms
64 bytes from 192.168.1.3: icmp_seq=1 ttl=64 time=0.074 ms
64 bytes from 192.168.1.3: icmp_seq=2 ttl=64 time=0.076 ms
64 bytes from 192.168.1.3: icmp_seq=3 ttl=64 time=0.108 ms
64 bytes from 192.168.1.3: icmp_seq=4 ttl=64 time=0.076 ms

--- 192.168.1.3 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.074/0.083/0.108/0.013 ms
```

Most pedig pingeljünk meg egy másik számítógépet a helyi hálózaton:

```
% ping -c5 192.168.1.2
PING 192.168.1.2 (192.168.1.2): 56 data bytes
64 bytes from 192.168.1.2: icmp_seq=0 ttl=64 time=0.726 ms
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.766 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.700 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=0.747 ms
64 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=0.704 ms

--- 192.168.1.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.700/0.729/0.766/0.025 ms
```

Ha beállítottuk az /etc/hosts állományt, akkor a `192.168.1.2` helyett a gép nevét is megadhatjuk.

A hibák elhárítása

A hardverek és szoftverek beállításaiban mindig is valódi kín megtalálni a hibákat, és ezeket a kínokat többnyire úgy tudjuk enyhíteni, ha elõször az egyszerû hibaforrásokat szûrjük ki. Csatlakoztattuk a hálózati kábelt? Tisztességesen beállítottuk a hálózati szolgáltatásokat? Jól állítottuk be a tûzfalat? A FreeBSD képes kezelni a kártyát? A hibajelentések elküldése elõtt mindig bújjuk át a támogatott hardvereszközök listáját. A FreeBSD verziókat frissítsük a legújabb STABLE változatra. Olvassuk át a levelezési listák archívumait vagy legalább keressünk rá a témára az interneten.

Ha a kártya mûködik, de a teljesítménye nem kielégítõ, érdemes ennek utánanézni a tuning(7) man oldalon. Ilyenkor érdemes ellenõrizni a hálózati beállításainkat is, mivel a helytelen beállítások gyakran okoznak teljesítményvesztést.

Bizonyos esetekben láthatunk egy vagy két `device timeout` típusú hibát is, ami a kártyák egyes fajtáinál elfogadható. Ha azonban folyamatosan megjelennek vagy zavaróvá válnak, érdemes utánanéznünk, hogy az eszköz nem ütközik-e valamelyik másikkal. Mindenképpen gyõzõdjünk meg a kábelek épségérõl és csatlakoztatásáról. Még az is elképzelhetõ, hogy egyszerûen csak egy másik

hálózati kártyára van szükségünk.

Néha felbukkanak `watchdog timeout` jellegû hibák is. Ilyenkor elsõként mindig a hálózati kábelt ellenõrizzük. Egyes kártyáknak olyan PCI foglalatra van szükségük, ami támogatja a Bus Mastering opciót. Néhány régebbi alaplapon csak ilyen PCI bõvítõhely található (ami általában a 0. foglalat). Olvassunk utána a hálózati kártya és az alaplap dokumentációjában, hátha ezek okozzák a problémát.

A `No route to host` üzenet akkor jelenik meg, ha a rendszer képtelen megállapítani, milyen úton juttassa el a csomagokat a megadott célhoz. Ez többnyire olyankor történik meg, amikor nem adtunk meg alapértelmezett kézbesítési irányt (default route) vagy nem dugtuk be a hálózati kábelt. A `netstat -rn` kimenetébõl meg tudjuk állapítani, hogy létezik-e érvényes út az elérni kívánt cél felé. Ha nincs, akkor haladjunk tovább a Egyéb haladó hálózati témákre.

A `ping: sendto: Permission denied` jellegû üzeneteket többségében egy helytelenül beállított tûzfal okozza. Ha az `ipfw` mûködését engedélyeztük a rendszermagban, de nem adtunk meg hozzá szabályokat, akkor az alapértelmezett házirend szerint minden forgalmat blokkolni fog, tehát még a pingeket is! Ezzel kapcsolatban a Tűzfalak elolvasását ajánljuk.

Elõfordulhat, hogy a kártya teljesítménye igen gyenge vagy az átlagos alatt van. Ilyenkor a fizikai eszköz `autoselect` (automatikus) típusú kiválasztása helyett érdemes megadnunk a konkrét eszköznek megfelelõ típust. Habár ez a legtöbb hardver esetén beválik, nem mindenki számára jelent megoldást. Ismételten csak annyit tudunk ehhez hozzátenni, hogy ellenõrizzük a hálózati beállításainkat és olvassuk el a tuning(7) man oldalt.

11.9. Virtuális címek

A FreeBSD alkalmazása során igen gyakori a virtuális címek használata, aminek segítségével egyetlen szerver több szerverként képes látszódni a hálózaton. Ezt úgy érik el, hogy egyetlen felülethez több hálózati címet rendelnek hozzá.

Az adott hálózati csatolófelületnek van egy "valódi címe" és tetszõleges számú "álcíme". Ezeket az álcímeket általában az /etc/rc.conf állományban kell feltüntetni.

Az fxp0 felület esetén az álcímek megadása valahogy így néz ki:

```
ifconfig_fxp0_alias0="inet xxx.xxx.xxx.xxx netmask xxx.xxx.xxx.xxx"
```

Figyeljük meg, hogy az álcímekhez tartozó bejegyzések az `alias0` névvel kezdõdnek és szám szerint növekvõleg következnek egymás után (például, `_alias1`, `_alias2` és így tovább). A beállítás a sorozat elsõ kimaradó tagjánál megszakad.

Az álcímek hálózati maszkjának pontos meghatározása nagyon fontos, de szerencsére nem különösebben bonyolult. Minden felület esetén lennie kell egy olyan címnek, amely helyesen reprezentálja a hálózat hálózati maszkját. Minden egyéb olyan címnek, ami ugyanabba az alhálózatba esik, végig `1`-esekbõl álló hálózati maszkkal kell rendelkezniük (ami felírható `255.255.255.255` vagy `0xffffffff` formájában is).

Például vegyük azt, hogy az fxp0 felületen keresztül két hálózathoz csatlakozunk, melyek közül az egyik a 10.1.1.0, amelynek hálózati maszkja 255.255.255.0, és a 202.0.75.16, amelynek hálózati maszkja 255.255.255.240. Azt szeretnénk elérni, hogy a rendszerünk a 10.1.1.1 címtõl a 10.1.1.5 címig, valamint a 202.0.75.17 címtõl a 202.0.75.20 címig jelenjen meg a nekik megfelelõ hálózatokon. Ahogy arra már fentebb is utaltunk, az adott hálózati tartományban csak az elsõ címnek (ebben az esetben ez a 10.0.1.1 és a 202.0.75.17) kell valódi hálózati maszkkal rendelkeznie. Minden további címnek (a 10.1.1.2 és 10.1.1.5 között, valamint a 202.0.75.18 és 202.0.75.20 között) legyen 255.255.255.255 a hálózati maszkja.

Az alábbi /etc/rc.conf bejegyzések ennek az elrendezésnek megfelelõen állítják be a kártyát:

```
ifconfig_fxp0="inet 10.1.1.1 netmask 255.255.255.0"
ifconfig_fxp0_alias0="inet 10.1.1.2 netmask 255.255.255.255"
ifconfig_fxp0_alias1="inet 10.1.1.3 netmask 255.255.255.255"
ifconfig_fxp0_alias2="inet 10.1.1.4 netmask 255.255.255.255"
ifconfig_fxp0_alias3="inet 10.1.1.5 netmask 255.255.255.255"
ifconfig_fxp0_alias4="inet 202.0.75.17 netmask 255.255.255.240"
ifconfig_fxp0_alias5="inet 202.0.75.18 netmask 255.255.255.255"
ifconfig_fxp0_alias6="inet 202.0.75.19 netmask 255.255.255.255"
ifconfig_fxp0_alias7="inet 202.0.75.20 netmask 255.255.255.255"
```

11.10. Konfigurációs állományok

11.10.1. Az /etc felépítése

A beállításokkal kapcsolatos információk számos könyvtárban tárolódnak. Többek közt:

/etc	Általános rendszerszintû beállítások. Az itt levõ adatok a rendszer egészére vonatkoznak.
/etc/defaults	A rendszer konfigurációs állományainak alapértelmezett változatai.
/etc/mail	A sendmail(8) beállításához tartozó további állományok, egyéb levélküldéshez használt adatok.
/etc/ppp	A felhasználói és rendszermag szintû ppp programok beállításai.
/etc/namedb	A named(8) mûködéséhez szükséges adatok alapértelmezett helye. Általában a named.conf és a zónák leírását tároló állományok kerülnek ide.
/usr/local/etc	A telepített alkalmazások konfigurációs állományai. Néha alkalmazásonként külön könyvtárakba kerülnek a benne található állományok.

/usr/local/etc/rc.d	A telepített alkalmazások indításával és leállításával kapcsolatos szkriptek.
/var/db	Automatikusan generált rendszerszintû adatbázisok a csomagokkal, a programok helyével stb. kapcsolatosan.

11.10.2. Hálózati nevek

/etc/resolv.conf

Az /etc/resolv.conf határozza meg, hogy a FreeBSD névfeloldója miként fér hozzá az internet tartománynév rendszeréhez (a DNS-hez).

Az resolv.conf állományban leggyakrabban a következõ bejegyzések fordulnak elõ:

`nameserver`	Annak a névszernek az IP-címe, ahova a névfeloldó küldi a kéréseit. A névszervereket a felírás sorrendjében kérdezi meg, maximum hármat.
`search`	A hálózati nevek keresõlistája. Ezt általában a helyi hálózati nevek tartománya határozza meg.
`domain`	A helyi tartomány neve.

Egy átlagos resolv.conf tartalma:

```
search example.com
nameserver 147.11.1.11
nameserver 147.11.100.30
```

Csak egy `search` és `domain` opciót szabad megadni.

A DHCP használatakor a dhclient(8) felül szokta írni a resolv.conf tartalmát a DHCP szervertõl kapott információkkal.

/etc/hosts

Az /etc/hosts az internet kezdeti napjaira emlékeztetõ egyszerû szöveges adatbázis. A nevek és IP-címek közti leképzéseket a DNS és NIS rendszerekkel karöltve oldja fel. Ide a helyi hálózaton csatlakozó számítógépek neveit lehet beírni ahelyett, hogy erre a célra beállítanánk egy külön named(8) szervert. Ezenkívül még az /etc/hosts állományba internetes nevek rekordját is felvehetjük, amivel így csökkenthetjük a gyakran használt nevek feloldására irányuló külsõ kéréseket.

```
# $FreeBSD$
#
#
```

```
# A hálózati nevek adatbázisa
#
# Ebbe az állományba rakjuk a helyi hálózaton található címeket és
# a hozzájuk tartozó hálózati neveket, ahol szinte ugyanez az
# adatbázis megtalálható.  A 'my.domain' helyére a saját gépünk
# nevét írjuk be.
#
# A DNS vagy NIS alkalmazása esetén ez az állomány nem feltétlenül kerül
# felhasználásra. A névfeloldás sorrendjét az /etc/nsswitch.conf
# állományban adhatjuk meg.
#
::1                     localhost localhost.my.domain
127.0.0.1               localhost localhost.my.domain

#
# Egy képzeletbeli hálózat.
#10.0.0.2               myname.my.domain myname
#10.0.0.3               myfriend.my.domain myfriend
#
# Az RFC 1918-nak megfelelõen a következõ IP-címekkel rendelkezõ
# alhálózatok sosem csatlakozhatnak közvetlenül az internetre:
#
#       10.0.0.0        -   10.255.255.255
#       172.16.0.0      -   172.31.255.255
#       192.168.0.0     -   192.168.255.255
#
# Amikor csatlakozunk az internethez, egy valódi, hivatalosan
# kiosztott számra lesz szükségünk.  Ne találjunk ki magunknak
# hálózati címeket, hanem használjuk az internetszolgáltatótól
# kapott címet (amennyiben rendelkezünk # ilyennel) vagy az
# regionális internetes nyilvántartásban szereplõ címek közül
# valamelyiket (ARIN, APNIC, LACNIC, RIPE NCC vagy AfriNIC).
```

Az /etc/hosts formai felépítése igen egyszerû:

```
[internetes cím] [hivatalos hálózati név] [álnév1] [álnév2] ...
```

Tehát például:

```
10.0.0.1 azEnValodiNevem.aHalozaton.hu azEnValodiNevem izemize1 izemize2
```

A részletekért keressük fel a hosts(5) man oldalt.

11.10.3. A naplóállományok beállítása

syslog.conf

A syslog.conf állomány a syslogd(8) program beállításait tartalmazza. Segítségével megadhatjuk,

hogy a `syslog` által generált üzenetek egyes típusait milyen naplóállományokba mentsük.

```
# $FreeBSD$
#
# Ebben az állományban HASZNÁLHATÓAK szóközök a mezõk elválasztására,
# habár a többi *nix-típusú rendszer inkább tabulátorokat használ
# erre a célra. Ha több rendszeren is használni akarjuk ezt az
# állományt, akkor ne használjunk szóközöket.
#
# A többit lásd a syslog.conf(5) man oldalon.
#
.err;kern.debug;auth.notice;mail.crit           /dev/console
*.notice;kern.debug;lpr.info;mail.crit;news.err /var/log/messages
security.*                                      /var/log/security
mail.info                                       /var/log/maillog
lpr.info                                        /var/log/lpd-errs
cron.*                                          /var/log/cron
*.err                                           root
*.notice;news.err                               root
*.alert                                         root
*.emerg                                         *
# Tegyük vissza ezt a sort, ha a /dev/console eszközre kiírt
# üzeneteket át akarjuk irányítani az /var/log/console.log állományba.
#console.info                                   /var/log/console.log
# Ha az összes üzenetet a /var/log/all.log állományba akarjuk menteni,
# akkor tegyük vissza ezt a sort.
#*.*                                            /var/log/all.log
# Ha egy "loghost" nevû gépre szeretnénk naplózni, akkor tegyük vissza
# ezt a sort.
#*.*                                            @loghost
# Az inn használatakor tegyük vissza ezeket a sorokat.
# news.crit                                     /var/log/news/news.crit
# news.err                                      /var/log/news/news.err
# news.notice                                   /var/log/news/news.notice
!startslip
*.*                                             /var/log/slip.log
!ppp
*.*                                             /var/log/ppp.log
```

A syslog.conf(5) man oldalának elolvasásával tudhatunk meg többet ezekrõl.

newsyslog.conf

A newsyslog.conf a newsyslog(8) beállításait tároló állomány. Ez egy olyan program, amelyet általában a cron(8) futtat le. A newsyslog(8) dönti el, hogy mikor van szükség a naplók archiválására és átrendezésére. Ennek során a logfile állományból logfile.0 lesz, a logfile.0 állományból pedig logfile.1 és így tovább. Beállíthatjuk úgy is, hogy a naplóállományokat archiválja gzip(1) formátumban, aminek megfelelõen ezek logfile.0.gz, logfile.1.gz és ehhez hasonló névvel jönnek létre.

A newsyslog.conf megadja, hogy melyik naplóállományokat kell felügyelni, mennyi példányt tartsunk meg belõlük és mikor kell velük foglalkozni. A naplóállományok átrendezhetõek és/vagy archiválhatóak egy adott méret elérésekor vagy egy adott idõ eltelte után.

```
# A newsyslog konfigurációs állománya
# $FreeBSD$
#
# állománynév     [tulajdonos:csoport]  mód  darab méret mikor [ZB] [/pid_állomány]
[jelzés]
/var/log/cron                           600  3     100  *     Z
/var/log/amd.log                        644  7     100  *     Z
/var/log/kerberos.log                   644  7     100  *     Z
/var/log/lpd-errs                       644  7     100  *     Z
/var/log/maillog                        644  7     *    @T00  Z
/var/log/sendmail.st                    644  10    *    168   B
/var/log/messages                       644  5     100  *     Z
/var/log/all.log                        600  7     *    @T00  Z
/var/log/slip.log                       600  3     100  *     Z
/var/log/ppp.log                        600  3     100  *     Z
/var/log/security                       600  10    100  *     Z
/var/log/wtmp                           644  3     *    @01T05 B
/var/log/daily.log                      640  7     *    @T00  Z
/var/log/weekly.log                     640  5     1    $W6D0 Z
/var/log/monthly.log                    640  12    *    $M1D0 Z
/var/log/console.log                    640  5     100  *     Z
```

További információkat a newsyslog(8) man oldaláról nyerhetünk.

11.10.4. sysctl.conf

A sysctl.conf állomány leginkább az rc.conf állományhoz hasonlít, benne az értékeket `változó=érték` párokban adhatjuk meg. Az itt definiált értékek akkor kerülnek ténylegesen beállításra, amikor a rendszer többfelhasználós módba vált. Ezen a módon nem mindegyik változó értékét tudjuk átállítani.

A sysctl.conf állományban az alábbi érték beállításával tudjuk beállítani, hogy a rendszer ne naplózza, amikor a programok végzetes jelzéssel fejezõdnek be, valamint azt, hogy a felhasználók láthassák egymás futó programjait:

```
# Ne naplózzuk a végzetes jelzésekhez (például sig 11) tartozó kilépéseket.
kern.logsigexit=0

# Ne engedjük a felhasználóknak, hogy lássák egy másik felhasználó
# azonosítójával futó programokat.
security.bsd.see_other_uids=0
```

11.11. Finomhangolás a sysctl használatával

A sysctl(8) egy olyan felület, amely lehetõséget biztosít egy mûködõ FreeBSD rendszer megváltoztatására. Segítségével többek közt hozzáférhetünk a TCP/IP protokollkészlet és a virtuális memóriát kezelõ alrendszer rengeteg apró opciójához, melyek megfelelõ beállításával egy tapasztalt rendszergazda kezében drasztikusan növelhetõ a rendszer teljesítménye. A sysctl(8) alkalmazásával több mint ötszáz rendszerszintû változó kérdezhetõ le és állítható be.

A sysctl(8) két funkciót rejt magában: a rendszer beállításainak lekérdezését és módosítását.

Így nézhetjük meg az összes lekérdezhető változót:

```
% sysctl -a
```

Így kérhetjük egy konkrét változó, például a `kern.maxproc` értékét:

```
% sysctl kern.maxproc
kern.maxproc: 1044
```

Egy adott változó értékének módosításához pedig használjuk a *változó=érték* felírást:

```
# sysctl kern.maxfiles=5000
kern.maxfiles: 2088 -> 5000
```

A sysctl változók értékei lehetnek karakterláncok, számok és logikai értékek (ahol az `1` az igennek, a `0` a nemnek felel meg).

Ha a számítógép indításakor automatikusan be akarunk állítani bizonyos változókat, akkor vegyük fel ezeket az /etc/sysctl.conf állományba. Ennek pontosabb részleteit a sysctl.conf(5) man oldalon és a sysctl.confban találhatjuk meg.

11.11.1. A sysctl(8) írásvédett értékei

Egyes esetekben szükséges lehet a sysctl(8) írásvédett változóinak módosítása. Habár gyakran elengedhetetlen, ezt kizárólag csak a rendszer (újra)indításakor tudjuk megtenni.

Például egyes laptopoknál a cardbus(4) eszköz nem próbálkozik több memóriaterület használatával, ezért egy ehhez hasonló hibával leáll:

```
cbb0: Could not map register memory
device_probe_and_attach: cbb0 attach returned 12
```

Az ilyen és ehhez hasonló esetekben gyakran olyan sysctl(8) változók alapértelmezett értékeit kellene megváltoztatnunk, amelyek írásvédettek. Ilyenkor tegyük az érintett sysctl(8) változó "objektumazonosítóját" (OID) és a hozzá tartozó értéket a /boot/loader.conf állományunkba. Az

alapértelmezéseket a /boot/defaults/loader.conf állományban találjuk meg.

A fentebb tárgyalt probléma megoldásához a felhasználónak a `hw.pci.allow_unsupported_io_range=1` értéket kell beállítania az elõbb említett állományban. Ezután már a cardbus(4) megfelelõen fog mûködni.

11.12. A lemezek finomhangolása

11.12.1. Sysctl változók

`vfs.vmiodirenable`

A `vfs.vmiodirenable` sysctl változó értéke lehet 0 (ki) vagy 1 (be, és ez az alapértelmezés is). Ez a változó vezérli a könyvtárak gyorsítótárazását a rendszerben. A könyvtárak többsége kis méretû, így az állományrendszerbõl csak egyetlen (általában 1 KB méretû) darabkát használnak és még ennél is kevesebbet (általában 512 byte-ot) a pufferben. A változó kikapcsolt (avagy 0) értéke mellett a puffer csak rögzített számú könyvtárat táraz be még abban az esetben is, amikor temérdek mennyiségû memória áll a rendelkezésére. Ha viszont (az 1 értékkel) engedélyezzük, akkor a rendszer a könyvtárak tárazására felhasználja a virtuális memóriában pufferelt lapokat is, amivel lényegében az összes elérhetõ memóriát a könyvtárak tárazására fordítja. Ilyenkor azonban az egyes könyvtárak tárazására használt legkisebb memóriaterület a fizikai lapmérettel egyezik meg (ami általában 4 KB) és nem 512 byte. Abban az esetben javasoljuk ennek a beállításnak a használatát, ha olyan szolgáltatásokkal dolgozunk, amelyek nagy számú állománnyal dolgoznak egyszerre. Ilyen szolgáltatások többek közt a webes gyorsítótárak, nagyobb levelezõrendszerek és hírrendszerek. Az opció engedélyezése alapvetõen nem veti vissza a rendszer teljesítményét még akkor sem, ha ezzel memóriát pazarlunk el, de ezt igazából érdemes kikísérletezni.

`vfs.write_behind`

A `vfs.write_behind` sysctl változó alapértelmezett értéke `1` (bekapcsolt). Ez arra utasítja az állományrendszert, hogy csak akkor küldje ki az adatokat az eszközre, ha belõlük teljes fürtök gyûltek össze. Ez jellemzõ módon nagyobb szekvenciális állományok írása esetén kedvezõ. Arra szolgál, hogy segítségével el lehessen kerülni az I/O túlságosan gyakori módosítások okozta terhelését. Bizonyos körülmények közt ez azonban lassíthatja a futó programok mûködését, ezért ilyenkor érdemes megfontolni a kikapcsolását.

`vfs.hirunningspace`

A `vfs.hirunningspace` sysctl változó értéke azt adja meg, hogy tetszõleges számú példánynál rendszerszinten mekkora mértékû írási mûvelet irányítható át a lemezvezérlõk soraiba. Az alapértelmezés többnyire elegendõ, de olyan gépeken, ahol sok lemez dolgozik egyszerre, ez az érték négy vagy öt *megabyte-ra* is felszökhet! Hozzátennénk, hogy ha ezt az értéket túlságosan nagyra állítjuk (és így túllépjük a puffer írási küszöbértékét), akkor ezzel hihetetlenül gyenge fürtözési teljesítményt nyerünk. Semmiképp se állítsuk túlzottan nagy értékre! A nagyobb írási értékek a velük párhuzamos olvasások számára késleltetést is jelentenek.

Találhatunk még más egyéb pufferelési és gyorsítótárazási sysctl változókat, azonban ezek megváltoztatását egyáltalán nem javasoljuk, mivel a virtuális memória alrendszer kiválóan tudja önállóan állítani ezeket a paramétereit.

vm.swap_idle_enabled

A vm.swap_idle_enabled sysctl változó módosítása olyan nagyobb többfelhasználós rendszerekben bizonyulhat hasznosnak, ahol sok felhasználó lép be és lép ki a rendszerbe és sok az üresjáratban futó program. Az ilyen jellegû rendszerek hajlamosak nagy mennyiségû folyamatos terhelést mérni a tartalékolt szabad memóriára. A beállítás engedélyezésével, valamint a vm.swap_idle_threshold1 és a vm.swap_idle_threshold2 változókon keresztül a kilapozás "reakcióidejének" alkalmas behangolásával a megszokottnál gyorsabban lenyomhatjuk az üresjáratban dolgozó programokhoz tartozó memórialapok prioritását, amivel a kilapozásokat vezérlõ démon kezére játszunk. Azonban tényleg csak akkor engedélyezzük ezt a lehetõséget, ha valóban szükségünk van rá, mivel így a memóriát jóval elõbb lapozzuk ki és ezzel több lapozóállományt és lemezteljesítményt emésztünk fel. Kisebb rendszerekben jól behatárolható a hatása, azonban a nagyobb rendszerekben, ahol már eleve visszafogott mértékû lapozás történik, ez a beállítás lehetõvé teszi a virtuális memóriát kezelõ alrendszer számára, hogy könnyedén ki- és be rakosgasson komplett futó programokat a memóriába.

hw.ata.wc

A FreeBSD 4.3 egyszer már kacérkodott az IDE-lemezek írási pufferének kikapcsolásával. Ez ugyan csökkentette az IDE-lemezek írási sávszélességét, azonban bizonyos merevlemezgyártók gondatlanságából eredõ súlyos adatvesztések miatt szükséges volt a használata. A gond ezzel kapcsolatban ott van, hogy egyes IDE-meghajtók hazudnak az írások teljesítésérõl. A lemezek írási gyorsítótárazásának bekapcsolásával az IDE-meghajtók nem csak az írások sorrendjét rendezik át, hanem nagyobb terhelés esetén egyes blokkokat jóval késõbb is rögzítenek. Ezért a rendszer esetleges összeomlása vagy egy áramkimaradás súlyos károkat okozhat az állományrendszerben. A FreeBSD úgy döntött, hogy a megbízhatóságot választja. Sajnos ez olyan nagyságú teljesítményvesztést okozott, hogy a következõ kiadásban már kénytelenek voltunk alapértelmezés szerint is visszakapcsolni ezt a lehetõséget. A hw.ata.wc nevû sysctl változó vizsgálatával ellenõrizhetjük a rendszerünkön érvényes alapértelmezett beállítást. Amennyiben az IDE írások gyorsítótárazása nem engedélyezett, akkor ezt a változó értékének 1-re állításával állíthatjuk vissza. Ezt a rendszer indításakor a rendszerbetöltõben tehetjük meg. A rendszermag indítása után ennek már nincs hatása.

A részleteket a ata(4) man oldalon tudhatjuk meg.

SCSI_DELAY (kern.cam.scsi_delay)

A rendszermag SCSI_DELAY nevû beállítása a rendszer indulásának idejét hivatott mérsékelni. Az alapértelmezett értéke viszonylag magas, innen származik a rendszer indítása során keletkezõ 15 másodperces csúszás. Általában az is megfelelõ, ha ezt visszavesszük az 5 értékre (fõleg a modernebb meghajtók számára). A FreeBSD újabb (5.0 vagy késõbbi) változataiban ez az érték már a kern.cam.scsi_delay sysctl változó értékével is megadható a rendszer indításakor. Azonban ügyeljünk rá, hogy mind a finomhangoláshoz használt változó, mind pedig rendszermag beállítása *ezredmásodpercben* és *nemmásodpercben* értelmezi ezt az értéket.

11.12.2. Soft Updates

A tunefs(8) nevû program használható az állományrendszerek finomhangolására. Nagyon sok opciót találhatunk benne, de itt most csak a "Soft Updates" ki- és bekapcsolásával foglalkozunk, amit a következõ módon tehetünk meg:

```
# tunefs -n enable /allomanyrendszer
# tunefs -n disable /allomanyrendszer
```

Amíg egy állományrendszer csatlakoztatott állapotban van, addig nem módosítható a tunefs(8) paranccsal. A Soft Updates bekapcsolására ezért az a legalkalmasabb idõpont, amikor egyfelhasználós módban vagyunk és még egyetlen partíciót sem csatlakoztattunk.

A Soft Updates beállítás engedélyezése a memóriában pufferelt gyorsítótáron keresztül jelentõs mértékben fokozza a metaadatok teljesítményét, elsõsorban az állományok létrehozását és törlését. A Soft Updates használatát ezért minden állományrendszer esetén ajánljuk. A Soft Updates alkalmazásának két rossz oldalára kell tekintettel lennünk. Elõször is a Soft Updates a rendszer összeomlása esetén ugyan garantálja az állományrendszer konzisztenciáját, de könnyen elképzelhetõ, hogy több másodperccel (vagy akár egy egész perccel!) hátrébb jár a fizikai lemez frissítésében. Másodszor a Soft Updates késlelteti az állományrendszer blokkjainak felszabadítását. Ha van egy olyan állományrendszerünk (mint például a rendszer indításához használt gyökér partíció), ami már majdnem betelt, akkor egy nagyobb frissítés, például a `make installworld` parancs kiadása, során az állományrendszer egyszerûen kifogy a helybõl és így a frissítés meghiúsul.

Bõvebben a Soft Updates mûködésérõl

Két hagyományos megközelítés létezik az állományrendszerek metaadatainak visszaírására. (A metaadatok módosításakor olyan nem adatot tartalmazó blokkok változnak meg, mint például az állományokra vonatkozó információk vagy a könyvtárak.)

Eredetileg alapértelmezés szerint a metaadatok változásait szinkron módon írták ki. Amikor egy könyvtár megváltozott, a rendszer egészen addig várt, amíg ez a változás a lemezre nem íródott. Ugyanekkor az állományok adatait tartalmazó pufferek (az állományok tartalma) átkerültek a pufferelt gyorsítótárba, hogy majd késõbb, aszinkron módon kerüljenek kiírásra. Ennek az implementációnak a biztonságos mûködés volt az elõnye, mivel így a metaadatok még akkor is konzisztens állapotban maradtak, amikor valamilyen hiba következett be. Tehát egy állomány vagy teljesen létrejött vagy egyáltalán nem. Ha az állományhoz tartozó blokkok már nem tudtak kijutni a gyorsítótárból az összeomlás ideje elõtt, akkor az fsck(8) felismerte ezt a helyzetet és az állományrendszer ilyen jellegû hibáját úgy orvosolta, hogy az adott állomány méretét nullára állította. Ezenkívül még az implementációs részletek is tiszták és egyszerûek maradtak. Ennek viszont hátránya, hogy a metaadatok kezelése lassú. Ha például kiadunk egy `rm -r` parancsot, akkor az a könyvtárban levõ állományokat szekvenciálisan dolgozza fel, de minden egyes változtatást (az állományok törlését) csak szinkron módon rögzíti a lemezre. Ezek a frissítések érintik magát a könyvtárat, az állományokkal kapcsolatos információkat tároló táblázatot (az ún. inode táblát) és minden valószínûség szerint az állományok által lefoglalt blokkokat is közvetve. Hasonló megfontolások élnek a nagyobb könyvtárszerkezetek kibontása esetén is (`tar -x`).

A második lehetõség a metaadatok aszinkron frissítése. Ez az alapértelmezés a Linux ext2fs és BSD-k `mount -o async` opcióval csatlakoztatott UFS állományrendszerei esetén. Ilyenkor minden metaadattal kapcsolatos aktualizálás egyszerûen bekerült a pufferelt gyorsítótárba, tehát az állományok adatai és ezek a típusú frissítések keverednek. Ennek a megvalósításnak az az elõnye, hogy nem kell megvárni, amíg a metaadatok is kiíródnak a lemezre, ezért a metaadatok óriási mennyiségû változásával járó mûveletek sokkal gyorsabban hajtódnak végre, mint a szinkron

esetben. Sõt, maga az implementáció is tiszta és egyszerû marad, ezért a kódban megjelenõ hibák beszivárgásának kockázata alacsony. A módszer hátránya, hogy egyáltalán semmilyen garanciát nem kapunk az állományrendszer konzisztenciájára. Ha tehát egy rengeteg metaadat megváltozásával együttjáró mûvelet közben történik valamilyen probléma (áramkimaradás, vagy valaki egyszerûen megnyomja a reset gombot), akkor az állományrendszer elõre kiszámíthatatlan állapotba kerül. A rendszer újbóli indításakor ezért nincs lehetõségünk megvizsgálni az állományrendszer állapotát. Elképzelhetõ, hogy az állományokhoz tartozó adatok már kikerültek a lemezre, miközben a rá vonatkozó inode- vagy könyvtári bejegyzések még nem. Így lényegében lehetetlen olyan `fsck` implementációt készíteni, ami képes lenne eltüntetni ezt a káoszt (hiszen az ehhez szükséges adatok nem állnak rendelkezésre). Ha az állományrendszer helyrehozhatatlanul károsodott, akkor csak a newfs(8) és a biztonsági mentés visszaállítása segíthet rajta.

Ezt általában úgy küszöbölik ki, hogy az egészhez hozzáteszik még a *módosított területek feljegyzését*, amit gyakran csak *naplózásnak* (journaling) neveznek, habár ezt az elnevezést nem mindenhol ilyen értelemben használják, ezért a tranzakciók naplózásának más formáira is utalhat. A metaadatok frissítése ebben az esetben is csak szinkron módon történik, de csak a lemez egy kisebb területére. Késõbb ez a megfelelõ helyére kerül. Mivel a lemez naplózásra fordított része egy viszonylag kis méretû, folytonos terület, a lemez fejének még a megterhelõbb mûveletek esetén sem kell sokat mozognia, ezért valójában ez a megoldás gyorsabb, mint a mezei szinkron frissítések. Az implementáció bonyolultsága továbbra is jól behatárolható, a velejáró hibalehetõségek kockázata alacsony. Hátránya, hogy minden metaadat kétszer íródik ki (egyszer a naplózási területre, aztán a megfelelõ helyre), ezért a hétköznapi használat során "visszaesés" tapasztalható a teljesítményben. Másrészrõl azonban egy összeomlás esetén a naplózási terület segítségével minden függõben levõ metaadattal kapcsolatos mûvelet könnyen visszafordítható vagy lezárható a rendszer következõ indításakor, így ezzel egy gyors helyreállítást nyerünk.

Kirk McKusick, a Berkeley FFS fejlesztõje ezt a problémát a Soft Updates segítségével hidalta át: a metaadatokkal kapcsolatos minden függõben levõ frissítést a memóriában tart, majd ezeket rendezett sorrendben írja ki a lemezre ("a metaadatok rendezett frissítése"). Ennek következményeképpen a metaadatok komolyabb frissítése során a késõbb érkezõ módosításoknak lehetõségük van "elkapni" a memóriában levõ korábbi változataikat, ha azok még nem kerültek ki a lemezre. Így az összes, például könyvtárakon végzett, mûvelet a lemezre írás elõtt általában elõször a memóriában játszódik le (az adatblokkok a pozíciójuknak megfelelõen kerülnek rendezésre, ezért a rájuk vonatkozó metaadatok elõtt nem jutnak ki a lemezre). Ha eközben a rendszer összeomlik, akkor így implicit módon a "napló visszalapozását" eredményezi: minden olyan mûvelet, ami már nem tudott kijutni a lemezre, meg nem történtnek számít. Ezen a módon az állományrendszernek egy 30 és 60 másodperc közti korábbi állapota marad fenn. Az algoritmus garantálja, hogy az összes használt erõforrás a nekik megfelelõ bittérképekben helyesen jelölõdik, a blokkokban és az inode-okban. Az összeomlás után az erõforrások kiosztásával kapcsolatban csak egyetlen hiba léphet fel: amikor olyan erõforrások jelölõdnek "használtnak", amelyek igazából "szabadok". Az fsck(8) azonban képes felismerni ezeket a helyzeteket és felszabadítani a nem használt erõforrásokat. A `mount -f` parancs kiadásával minden további következmény nélkül figyelmen kívül hagyhatjuk az állományrendszer félkész állapotát és csatlakoztathatjuk az állományrendszereket. A használatban már nem levõ erõforrások felszabadításához az fsck(8) parancsot késõbb kell futtatni. Ez az alapötlet húzódik meg a *háttérben végzett lemezellenõrzés* mögött. A rendszer indításakor az állományrendszernek csupán egy *pillanatképét* rögzítjük, és az `fsck` tényleges lefuttatását késõbbre toljuk. Mivel mindegyik állományrendszer csatlakoztatható "félkész" állapotban, ezért a rendszer képes elindulni többfelhasználós módban. Eközben a háttérben az `fsck` beütemezhetõ minden

olyan állományrendszer számára, ahol arra szükség van, hogy szabadítsa fel az esetlegesen már nem használt erõforrásokat. (Így a Soft Updates opciót nem alkalmazó állományrendszerek esetén továbbra is szükség van az elõtérben elvégzett `fsck` parancsra.)

A módszer elõnye, hogy így a metaadatokkal kapcsolatos mûveletek közel olyan gyorsak, mint az aszinkron módon végzett frissítések (tehát gyorsabb, mintha *naplóznánk*, ami ugye minden metaadatot kétszer ír ki). A hátránya a bonyolultabb kód (ami miatt növekszik az olyan hibák lehetõsége, amelyek érzékenyen befolyásolhatják a felhasználói adatok elvesztését) és a nagyobb memóriaigény. Ezenkívül még van néhány olyan egyéni jellemzõje, amelyet meg kell szokni. A rendszer összeomlása után az állományrendszer valamivel "régebbi" lesz. Amikor pedig megszokott szinkron megközelítés szerint az `fsck` lefutása után nulla méretû állományok jönnének létre, ezek az állományok a Soft Updates esetén egyáltalán meg sem jelennek, mivel sem a rájuk vonatkozó metaadatok, sem pedig a tartalmuk nem került ki a lemezre. Egy `rm` lefuttatása után a lemezterület addig nem kerül felszabadításra, amíg a frissítések teljesen rá nem kerülnek a lemezre. Ez nagyobb mennyiségû adat telepítésekor gondokat okozhat egy olyan állományrendszeren, ahol nincs elegendõ hely az állományok kétszeri tárolására.

11.13. A rendszermag korlátainak finomhangolása

11.13.1. Az állományok és a futó programok korlátozásai

`kern.maxfiles`

A `kern.maxfiles` értéke a rendszerünk igényeinek megfelelõen növelhetõ vagy csökkenthetõ. Ez a változó adja meg a rendszerünkben levõ állományleírók maximális számát. Amikor az állományleírókat tároló táblázat megtelik, a rendszer üzenetpufferében egy `file: table is full` üzenet jelenik meg, amit a `dmesg` paranccsal tudunk megnézni.

Minden megnyitott állomány, csatlakozás vagy FIFO elhasznál egy állományleírót. Egy nagyméretû szerver könnyen felemészthet több ezernyi állományleírót attól függõen, hogy milyen és mennyi szolgáltatást futtat egymás mellett.

A FreeBSD korábbi kiadásaiban a `kern.maxfiles` a rendszermag beállításait tartalmazó állomány `maxusers` (a rendszerben egyszerre jelenlevõ felhasználók maximumának) értékébõl származott, tehát a `kern.maxfiles` a `maxusers` értékével arányosan növekszik. Amikor készítünk egy saját rendszermagot, mindig érdemes a rendszerünk használatának megfelelõen beállítani ezt az értéket, mivel a rendszermag ebbõl a számból határozza meg a legtöbb elõre meghatározott korlátait. Mivel még egy komoly szerveren sem jelentkeznek be egyszerre 256 felhasználónál többen, nagyjából ugyanannyi erõforrásra van szüksége, mint egy nagyobb webszervernek.

A `kern.maxusers` értéke a rendelkezésre álló memóriának megfelelõen magától méretezõdik a rendszer indításakor, és amit futás közben csak a `kern.maxusers` sysctl változó írásvédett értékének lekérdezésébõl tudhatunk meg. Egyes oldalak üzemeltetése a `kern.maxusers` így megállapított értékétõl nagyobbat vagy éppen kisebbet igényel, ezért a betöltéskor minden gond nélkül át lehet állítani 64, 128 vagy 256 értékûre. Senkinek sem ajánljuk, hogy 256 felé menjen, hacsak tényleg nincs szüksége ekkora mennyiségû állományleíróra. A `kern.maxusers` függvényében beállított alapértelmezett értékek tetszõleges módon átállíthatóak a rendszer indításakor vagy futás közben a /boot/loader.conf módosításával (az ide kapcsolódó javaslatokról bõvebben lásd a loader.conf(5)

man oldalt vagy a /boot/defaults/loader.conf állományt) illetve a leírás más részén megadott módok szerint.

A korábbi kiadásokban úgy lehet önszabályozóra állítani a `maxusers` beállítást, ha explicit módon `0` értéket adtunk meg neki . A `maxusers` paraméter beállításakor érdemes legalább 4-et megadni, különösen akkor, ha használjuk az X Window Systemet vagy szoftvereket fordítunk le. Azért van erre szükség, mert a `maxusers` értéke által szabályozott legfontosabb mennyiség az egyszerre futtatható programok táblázatának maximális mérete, amelyet így számolunk ki: `20 + 16 * maxusers`. Tehát ha a `maxusers` értékét 1-re állítjuk be, akkor az elõbbi képlet értelmében csak 36 programunk futhat egymással párhuzamosan, beleértve mindazt a kb. 18 programot, amelyek a rendszerrel együtt indulnak, illetve még azt a további 15 programot, amelyeket az X Window System használatával indítunk el. Még egy olyan egyszerû dolog is, mint például egy man oldal megnézése, legalább kilenc programot indít el a szûréshez, kitömörítéshez és megnézéshez. Azonban ha a `maxusers` értékét 64-re állítjuk, akkor egyszerre akár már 1044 programot futtathatunk, ami szinte mindenre elegendõ. Ha persze egy új program indításakor kapunk egy típusú üzenetet vagy nagy számú konkurens felhasználóval futtatunk szervert (ilyen például az `ftp.FreeBSD.org`), akkor érdemes növelni ezt a számot és újrafordítani a rendszermagot.

A `maxusers` *nem* korlátozza a számítógépre egyszerre bejelentkezni képes felhasználók számát. Egyszerûen csak beállítja néhány táblázat méretét és az egyszerre futtatható programok mennyiségét a rendszert egyidejûleg használni kívánó felhasználók maximális számának figyelembevételével.

`kern.ipc.somaxconn`

Az `kern.ipc.somaxconn` sysctl változó a beérkezõ TCP kapcsolatokat fogadó sor hosszát határozza meg. Ennek az alapértelmezett értéke `128`, ami az új kapcsolatok megbízható kezeléséhez általában kevés egy erõsen leterhelt webszerver számára. Ilyen helyzetekben ezt az értéket javasolt `1024`-re vagy még annál is nagyobbra állítani. Az egyes szolgáltatások démonai ugyan szintén korlátozni szokták a fogadósoruk méretét (például a sendmail(8) vagy az Apache), de gyakran találunk a beállításai között olyat, amivel ennek a sornak a mérete növelhetõ. A nagyobb fogadósorok mellesleg jó szolgálatot tesznek a Denial of Service () típusú támadásokkal szemben is.

11.13.2. Hálózati korlátozások

A rendszermag `NMBCLUSTERS` nevû beállítása szab határt a rendszer részére elérhetõ memóriapufferek mennyiségének. Egy nagyobb forgalmú szerver esetén a pufferek alacsony száma gátat szabhat a FreeBSD képességeinek. Minden klaszter nagyjából 2 KB memóriát takar, így az 1024-es érték azt jelenti, hogy a rendszermag memóriájából 2 megabyte-ot fordítunk a hálózati pufferelésre. Egyszerûen kiszámítható, mennyire is van szükségünk: ha van egy webszerverünk, amely egyszerre legfeljebb 1000 párhuzamos kapcsolatot fogad, és minden kapcsolat lefoglal 16 KB-ot a fogadó-, valamint újabb 16 KB-ot a küldõpuffer számára, akkor megközelítõleg 32 MB-nyi hálózati pufferre lesz szükségünk a webszerver hatékony mûködéséhez. Ezt az értéket gyakran még érdemes megszorozni kettõvel, így 2 x 32 MB / 2 KB = 64 MB / 2 KB = 32768. Több memóriával rendelkezõ számítógépek esetén egy 4096 és 32768 közti értéket javaslunk. Semmilyen körülmények között ne adjunk meg ennél nagyobb értéket, mert ezzel a rendszer már az indítása során összeomolhat. A netstat(1) `-m` beállításával ellenõrizhetjük a hálózati klaszterek kihasználtságát.

A kern.ipc.nmbclusters változó értékét a rendszer indításakor érdemes megváltoztatni. A FreeBSD korábbi változataiban ehhez a rendszermag NMBCLUSTERS nevû config(8) paraméterének módosítására van szükségünk.

Az olyan forgalmasabb szervereken, ahol sokat használják a sendfile(2) rendszerhívást, szükségünk lehet a sendfile(2) által használt pufferek számának növelésére a rendszermag NFSBUFS nevû konfigurációs paraméterén vagy a /boot/loader.conf állományon keresztül (lásd loader(8)). Amikor a futó programok közül sokan vannak sfbufa állapotban, akkor az egyértelmûen annak a jele, hogy ezen a paraméteren állítanunk kell. A kern.ipc.nsfbufs egy írásvédett változót, amelyet a rendszermag állít be. Ez a paraméter névlegesen a kern.maxusers változó értékének megfelelõen változik, de bizonyos esetekben ettõl függetlenül önállóan kell behangolni.

Annak ellenére, hogy egy socketet blokkolásmentesnek jelöltünk meg, a sendfile(2) meghívása egy blokkolásmentes socketre blokkolódást eredményezhet egészen addig, amíg a használatához elegendõ struct sf_buf struktúra össze nem gyûlik.

net.inet.ip.portrange.*

A net.inet.ip.portrange.* sysctl változók vezérlik a TCP és UDP csatlakozásokhoz automatikusan hozzárendelt portszámok tartományát. Három ilyen tartomány létezik: az alsó, az alapértelmezett és a felsõ tartomány. A legtöbb hálózati program a net.inet.ip.portrange.first és net.inet.ip.portrange.last változók által rendre az 1024-tõl 5000-ig kijelölt alapértelmezett tartományt használja. A kimenõ kapcsolatok is rögzített porttartományokat követnek, és adott körülmények mellett be lehet állítani úgy a rendszerünket, hogy ezen kívül osszon ki portokat. Ez a legtöbbször akkor fordul elõ, amikor egy erõsen leterhelt webproxyt mûködtetünk. A porttartományok nem okoznak gondot olyan szervereknél, ahol általában bejövõ kapcsolatokra lehet számítani, tehát például webszerverek esetén, vagy ahol korlátozott a kimenõ kapcsolatok száma, mint például a levelek továbbításánál. Ha olyan helyzetbe keverednénk, ahol már kifutunk a felhasználható portokból, a net.inet.ip.portrange.last mérsékelt növelésével javasolt kitörni. Ilyenkor a 10000, 20000 vagy 30000 értékek elfogadhatóak. Amikor megváltoztatjuk a porttartományok határait, elõtte mindig gondoljuk át, milyen hatással lehet ez a tûzfalra. Egyes tûzfalak blokkolhatnak bizonyos tartományokat (általában az alacsonyabbakat) és arra számítanak, hogy a rendszerek a kimenõ kapcsolatokhoz a nagyobb számú portokat használják - ebbõl kifolyólag nem ajánlott csökkenteni a net.inet.ip.portrange.first értékét.

A TCP sávszélesség-késletetés szorzat

A TCP sávszélesség-késleltetés szorzat korlátozása hasonlít a NetBSD-ben megtalálható TCP/Vegas implementációhoz. A net.inet.tcp.inflight.enable sysctl változó 1-re állításával lehet engedélyezni. A rendszer ilyenkor minden egyes kapcsolathoz megpróbálja kiszámítani a sávszélesség-késleltetés szorzatot és az optimális átviteli sebesség fenntartásához illeszkedõen korlátozni a hálózat felé küldött adatok sorának hosszát.

Ez a lehetõség még olyankor bizonyulhat hasznosnak, amikor modemen, Gigabit Etherneten vagy nagysebességû WAN (vagy bármilyen más nagy sávszélesség-késleltetés szorzattal bíró) összeköttetéseken keresztül küldünk át adatokat, különösen abban az esetben, amikor ablakméretezést is használnunk vagy nagy küldési ablakot állítottunk be. Az engedélyezésekor ne felejtsük el net.inet.tcp.infligt.debug változót sem beállítani 0-ra (amivel így kikapcsoljuk a nyomkövetést),éles használat esetén pedig elõnyös lehet a net.inet.cp.inflight.min változót

legalább 6144-re állítani. Azonban hozzátesszük, hogy összeköttetéstõl függõen a nagy minimum értékek tulajdonképpen kikapcsolják a sávszélességkorlátozást. Ez a korlátozási lehetõség csökkenti a közbensõ út adatainak és csomagváltásokhoz tartozó soroknak a méretét, miközben csökkenti a helyi számítógép felületén felépülõ sorok méretét is. Ha kevesebb csomagot rakunk be a sorba, akkor az interaktív kapcsolatok, különösen a lassabb modemek esetében, kisebb *körbejárási idõvel* (Round Trip Time) mûködnek. Továbbá megemlítenénk, hogy ez a lehetõség csak az adatok küldésére (feltöltésére, szerveroldalra) van hatással. Semmilyen befolyása nincs az adatok fogadására (letöltésére).

A net.inet.tcp.inflight.stab állítgatása *nem* ajánlott. A paraméter értéke alapértelmezés szerint 20, ami legfeljebb 2 csomag hozzáadását jelenti a sávszélesség-késleltetés szorzat ablakának kiszámításakor. Erre a kiegészítõ ablakra azért van szükség, hogy stabilizálni tudjuk vele az algoritmust és javítani tudjuk a változó feltételekre adott reakciót, de lassabb összeköttetések esetében nagyobb ping idõket is eredményezhet (habár ezek még így kisebbek, mint ha nem használnánk az algoritmust). Ilyen esetekben megpróbálhatjuk 15-re, 10-re vagy esetleg 5-re visszavenni a paraméter értékét, de ekkor a kívánt hatás eléréséhez minden bizonnyal a net.inet.tcp.inflight.min értékét is redukálunk kell majd (például 3500-ra). Ezen paraméterek megváltoztatását csak végsõ esetben ajánljuk!

11.13.3. Virtuális memória

kern.maxvnodes

A vnode egy állomány vagy könyvtár belsõ ábrázolása. Ennek megfelelõen a vnode-ok számának növelésével az operációs rendszer spórolni tud a lemezmûveletekkel. Ezt általában maga az operációs rendszer szabályozza, és nincs szükség a finomhangolására. Néhány esetben, amikor a lemezmûveletek jelentik a rendszerben a szûk keresztmetszetet és kezdenek elfogyni a vnode-ok, szükség lehet ennek a számnak a növelésére. Ehhez az inaktív és szabad fizikai memória mennyiségét kell számításba vennünk.

Így kérhetjük le a pillanatnyilag használatban levõ vnode-ok mennyiségét:

```
# sysctl vfs.numvnodes
vfs.numvnodes: 91349
```

Így tudhatjuk meg a vnode-ok maximális számát:

```
# sysctl kern.maxvnodes
kern.maxvnodes: 100000
```

Ha a vnode-ok aktuális kihasználtsága megközelíti a csúcsértéket, nagyjából ezerrel javasolt megnövelni a kern.maxvnodes értékét. Ezután figyeljük továbbra is a vfs.numvnodes változását. Ha ismét felkúszik a maximális értékre, akkor növeljük megint egy keveset a kern.maxvnodes értékén. Eközben a top(1) használatával figyelhetjük a memória kihasználtságának növekedését is, ilyenkor tehát több memóriának kell használatban lennie.

11.14. A lapozóterület bõvítése

Nem számít, mennyire tervezünk jól elõre, mindig elõfordulhat, hogy a rendszerünk mégsem teljesíti a kitûzött elvárásokat. Amennyiben további lapozóterület hozzáadására lenne szükségünk, azt igen könnyen megtehetjük. Háromféleképpen növelhetjük a lapozásra szánt területet: hozzáadunk a rendszerhez egy újabb merevlemezes meghajtót, NFS-en keresztül lapozunk, vagy egy már meglevõ partíción hozunk létre lapozóállományt.

A lapozóterület titkosításával, valamint annak lehetõségeivel és okaival kapcsolatban lapozzuk fel a kézikönyv A lapozóterület titkosításaát.

11.14.1. Lapozás egy új merevlemezre

A lapozóterület bõvítésének legjobb módja természetesen remek indok egy új merevlemez beszerzésére is. Elvégre egy merevlemezt mindig fel tudunk ilyen célra használni. Ha ezt a megoldást választjuk, elõtte ajánlott (újra) elolvasni a kézikönyv Kezdeti beállításokában a lapozóterületek elrendezésére vonatkozó javaslatokat.

11.14.2. Lapozás NFS-en keresztül

NFS-en keresztül csak akkor lapozzunk, ha ezt helyi lemezek segítségével nem tudjuk megtenni. Az NFS alapú lapozás hatékonyságát erõsen behatárolja a rendelkezésre álló hálózati sávszélesség és további terheket ró az NFS szerverünkre is.

11.14.3. Lapozóállományok

Lapozóállománynak egy adott méretû állományt hozzunk létre. Ebben a példában erre egy /usr/swap0 nevû, 64 MB méretû állományt fogunk használni. Természetesen bármilyen más nevet is választhatunk.

Példa 6. Lapozóállomány létrehozása FreeBSD-ben

1. Gyõzõdjünk meg róla, hogy a rendszermagunk beállításai között megtalálható a memórialemez meghajtójának (md(4)) használata. Ez a GENERIC rendszermag alapból tartalmazza.

   ```
   device   md   # Memória "lemezek"
   ```

2. Hozzunk létre egy lapozóállományt (/usr/swap0):

   ```
   # dd if=/dev/zero of=/usr/swap0 bs=1024k count=64
   ```

3. Állítsuk be rá a megfelelõ engedélyeket (/usr/swap0):

   ```
   # chmod 0600 /usr/swap0
   ```

4. Adjuk meg a lapozóállományt az /etc/rc.conf állományban:

```
swapfile="/usr/swap0"   # Állítsuk be swapfile értékét, ha külsõ
lapozóállományra van szükségünk.
```

5. Indítsuk újra a számítógépünket, vagy a lapozóállomány azonnali használtba vételéhez írjuk be:

```
# mdconfig -a -t vnode -f /usr/swap0 -u 0 && swapon /dev/md0
```

11.15. Energia- és erõforrásgazdálkodás

Fontos a hardveres erõforrásaink hatékony kihasználása. Az ACPI megjelenése elõtt az operációs rendszerek csak nehézkesen és rugalmatlanul tudták kezelni a rendszer energiafelhasználási és hõszabályzási lehetõségeit. A hardvert a BIOS kezelte, ezért a felhasználó kevesebbet tudott látni és irányítani az energiagazdálkodási beállításokból. Az *Fejlett energiagazdálkodás (Advanced Power Management, APM)* ehhez nyújtott egy erõsen korlátozott felületet. Napjaink operációs rendszereiben az energia- és erõforráskezelés az egyik legfontosabb alkotóelem. Például, ha az operációs rendszerrel folyamatosan figyelni akarjuk a rendszer hõmérsékletének váratlan növekedését (és errõl figyelmeztetést kérni).

A FreeBSD kézikönyvének ezen szakaszában az ACPI-rõl adunk egy átfogó áttekintést, a végén pedig összefoglaljuk a témához tartozó irodalmat.

11.15.1. Mi az ACPI?

A speciális energia- és konfigurációs illesztõ felület (Advanced Configuration and Power Interface, avagy ACPI) gyártók egy csoportja által létrehozott szabvány, amely a hardveres erõforrások és az energiagazdálkodás egységes felületét rögzíti (innen a neve). Döntõ szerepet játszik a *Beállítások és az energiagazdálkodás operációs rendszerek általi vezérlésében*, vagyis segítségével az operációs rendszer még nagyobb mértékben és rugalmassággal tudja irányítani ezeket a lehetõségeket. A modern operációs rendszerek az ACPI felbukkanásával "kitolták" a jelenleg meglevõ Plug and Play felületek korlátait. Az ACPI az APM közvetlen leszármazottja.

11.15.2. A Fejlett energiagazdálkodás (APM) hiányosságai

A *Fejlett energiagazdálkodás (APM)* a rendszer által felhasznált energiát annak elfoglaltsága alapján vezérli. Az APM-et támogató BIOS-t a (rendszert) gyártó állítja elõ és az adott hardverplatformra jellemzõ. Az APM operációs rendszerben levõ meghajtója hozzáférést biztosít az *APM szoftveres felületéhez*, amivel lehetõség nyílik az energiaszintek kezelésére. Az APM-et 2000 elõtt és körül még mindig használták egyes rendszerek gyártásánál.

Az APM használata négy nagyobb gondot rejt magában. Elõször is, az energiagazdálkodást a (gyártófüggõ) BIOS végzi el, és az operációs rendszernek errõl semmilyen ismerete nincsen. Ennek egyik példája az, amikor a felhasználó az APM-et ismerõ BIOS-ban beállítja a merevlemezek

automatikus kikapcsolásának idejét, majd amikor ez letelik, a BIOS az operációs rendszer tudta nélkül egyszerûen leállítja a lemezt. Másodszor: az APM mûködését a BIOS-ban programozták le, és teljesen az operációs rendszer hatáskörén túl tevékenykedik. Ez azt jelenti, hogy a felhasználó csak úgy tudja korrigálni az APM-es BIOS-ok problémáit, ha frissíti az alaplapi ROM-ot. Ez viszont egy nagyon kockázatos folyamat, amelynek hibája révén a rendszerünk helyrehozhatatlan állapotba kerülhet. Harmadszor: az APM alapvetõen egy gyártófüggõ megoldás, ami azt vonja maga után, hogy sok az átfedés (ugyanazt valósítják meg több módon), és ha az egyik gyártó BIOS-ában hibát találnak, akkor a másikéban az nem feltétlenül javítható. Végül, de nem utolsósorban, az APM alapú BIOS-okban nincs elég hely az igazán kifinomult energiagazdálkodási sémák vagy bármi más kialakítására, amivel a felhasználók képesek lennének az igényeikhez alakítani a számítógépet.

A *Plug and Play BIOS (PNPBIOS)* sok szempontból megbízhatatlannak bizonyult. A PNPBIOS ráadásul egy 16 bites megoldás, ezért az operációs rendszereknek 16 bites emulációt kell használniuk a PNPBIOS eszközeinek "eléréséhez".

A FreeBSD APM meghajtójának dokumentációját az apm(4) man oldalon találjuk.

11.15.3. Az ACPI beállítása

Az acpi.ko meghajtó alapértelmezés szerint a loader(8) segítségével töltõdik be, és *ne* is fordítsuk bele a rendszermagba. Ezt azzal tudnánk magyarázni, hogy modulokkal könnyebb dolgozni, például ha a rendszermag újrafordítása nélkül egy másik acpi.ko modult akarunk használni. Ezzel a lényegében a tesztelés is egyszerûbbé válik. Másik magyarázat, hogy a rendszer ACPI támogatása nem minden esetben mûködik rendesen. Ha a rendszer indítása során valamilyen problémát tapasztalunk, akkor próbálkozzunk meg az ACPI kikapcsolásával. Ezt a meghajtót nem lehet és nem is szabad kidobni a memóriából, mivel a hardverrel a rendszerbuszon keresztül tartja a kapcsolatot. Az ACPI a `hint.acpi.0.disabled="1"` sor megadásával kapcsolható a /boot/loader.conf állományban vagy a loader(8) parancssorában.

Az ACPI és az APM nem használató egyszerre. Közülük a késõbb betöltött magától kilép, ha észreveszi, hogy a másikuk már mûködésbe lépett.

Az ACPI és az acpiconf(8) használatával a rendszerünk készenléti módba helyezhetõ az `-s` valamint az `1-5` paraméterek megadásával. Ezek közül is a legtöbb felhasználó számára csak az `1` vagy a `3` (állapot mentése a fizikai memóriába) érdekes. Az `5` opció egy szoftveres kikapcsolást eredményez, ehhez hasonlóan:

```
# halt -p
```

A további opciók a sysctl(8) man oldaláról érhetõek el. Ezen kívül még olvassuk el az acpi(4) és acpiconf(8) man oldalakat is.

11.16. A FreeBSD ACPI támogatásának használata és nyomonkövetése

Az ACPI az eszközök felderítésének, energiagazdálkodásának és a korábban a BIOS által kezelt

hardverek szabványosított hozzáférésének alapjaiban új módja. Az ACPI folyamatosan fejlõdik, de útját az egyes alaplapok *ACPI Machine Language* (AML) bytekód implementációjában megjelenõ hibák, a FreeBSD rendszermag alrendszereinek befejezetlensége és az Intel® ACPI-CA értelmezõjében levõ hibák lassítják.

Ez a leírás azzal a szándékkal készült, hogy segítsünk a felhasználóknak megtalálni az általuk tapasztalt problémák gyökerét és ezzel segíteni az ACPI fejlesztõket a nyomonkövetésében és kijavításában. A fejlesztõk köszönik, hogy ezt elolvassuk és segédkezünk a rendszerünkkel kapcsolatban felmerülõ problémák orvosolásában!

11.16.1. A nyomkövetési információk beküldése

Mielõtt beküldenénk bármilyen problémát is, gondoskodjunk róla, hogy a BIOS-unk, és ha lehetséges, akkor a beágyazott vezérlõk, legfrissebb verzióját használjuk.

Megkérnénk azokat, akik hibát akarnak bejelenteni, hogy a következõ információkat küldjék a freebsd-acpi@FreeBSD.org címre:

- A hibás mûködés leírása, beleértve a rendszer típusát és gyártmányát, illetve minden olyat, aminek köze lehet a hibához. Ha eddig még nem tapasztaltuk, igyekezzünk minél pontosabban leírni a hiba keletkezésének folyamatát.
- A `boot -v` paranccsal indított rendszer dmesg(8) kimenetét, beleértve a vizsgálni kívánt hiba által okozott összes hibaüzenetet.
- A `boot -v` paranccsal és az ACPI használata nélkül indított rendszer dmesg(8) kimenete abban az esetben, ha ez segít megoldani a problémát.
- A `sysctl hw.acpi` parancs kimenete. Ezzel egyébként kitûnõen kideríthetõ, milyen lehetõségeket is kínál fel a rendszerünk.
- Az általunk használt *ACPI forrásnyelvének* (ACPI Source Language, ASL) elérhetõsége az interneten. Mivel ezek akár igen nagyok is lehetnek, ezért a listára közvetlenül ne küldjünk ASL kódokat! Az ASL másolatát az alábbi parancs kiadásával hozhatjuk létre:

  ```
  # acpidump -dt > név-rendszer.asl
  ```

  (Adjuk meg a *név* helyett a bejelentkezéshez használt nevünket, a *rendszer* helyett pedig a gyártót/típust. Például: njl-FooCo6000.asl)

Habár a legtöbb fejlesztõ a FreeBSD-CURRENT levelezési listat figyeli, a problémáink leírását mindenképpen a FreeBSD ACPI levelezési lista listára küldjük, hogy biztosan észrevegyék. A fejlesztõk azt kérik, hogy legyünk türelmesek, hiszen emellett mindannyian teljes állásban is dolgoznak. Ha az általunk felfedezett hiba nem teljesen egyértelmû, akkor a fejlesztõk valószínûleg meg fognak kérni arra, hogy a send-pr(1) használatával hozzunk róla létre egy hivatalos hibajelentést. A hibajelentés készítésekor lehetõleg a fentebb megadott információkat ugyanúgy adjuk meg. Ez segít a probléma szemmel tartásában és elhárításában. Az FreeBSD ACPI levelezési lista lista kihagyása nélkül közvetlenül ne küldjünk hibajelentést, mivel a hibabejelentõ rendszert elsõsorban emlékeztetõnek használjuk, nem pedig a hibák tényleges bejelentésére. Gyakran

elõfordul, hogy valaki korábban már találkozott az adott problémával.

11.16.2. Háttér

Az ACPI minden olyan modern számítógépben megtalálható, mely megfelel az ia32 (x86), ia64 (Itanium) vagy amd64 (AMD) architektúrának. A teljes szabvány rengeteg lehetõséget biztosít, többek közt a processzor teljesítményének kezelését, az energiaszintek vezérlését, hõzónákat, különféle akkumulátor rendszereket, beágyazott vezérlõk és a buszok felsorolását. A legtöbb rendszer általában nem a teljes szabványt valósítja meg. Például egy asztali rendszer általában csak a buszok felsorolásával kapcsolatos részeket tartalmazza, miközben egy laptop felajánlhatja a hûtés és az akkumulátor kezelését is. A laptopokban gyakorta találunk készenléti üzemmódot a maguk elbonyolított formájában.

Egy ACPI-nak megfelelõ rendszert számos összetevõ alkot. A BIOS-ok és chipkészletek gyártói a memóriában egy elõre rögzített ponton elhelyeznek bizonyos táblázatokat (például FADT), amelyekkel megadják például az APIC összerendeléseit (ezt az SMP rendszerek használják), a konfigurációs regisztereket és az egyszerûbb konfigurációs értékeket. Itt ezenkívül még bytekódok egy táblázata (amit *Differenciált rendszerleírtó táblának*, Differentiated System Description Table, DSDT nevezünk) is megtalálható, ahol az eszközök és módszerek nevei szerepelnek faszerû elrendezésben.

Az ACPI-hoz tartozó meghajtónak képesnek kell lennie értelmezni ezeket a rögzített táblázatokat, implementálni egy bytekód-értelmezõt, módosítani az eszközmeghajtókat és a rendszermagot az ACPI alrendszerbõl érkezõ információk befogadásához. A Linuxszal és a NetBSD-vel közösen a FreeBSD kapott egy ilyen értelmezõt az Intel®tõl (ACPI-CA). Az ACPI-CA forráskódja a rendszer forrásai között, a src/sys/dev/acpica könyvtárban található. A src/sys/dev/acpica/0sd könyvtárban található források pedig lehetõvé teszik, hogy az ACPI-CA mûködhessen FreeBSD-n. Végezetül, az ACPI eszközöket megvalósító meghajtók a src/sys/dev/acpica könyvtárban találhatóak.

11.16.3. Gyakori problémák

Az ACPI megfelelõ mûködéséhez minden alkotórésznek helyesen kell mûködnie. A most következendõkben elõfordulásuk gyakorisága szerint felsorolunk néhány ismert problémát, valamint a hozzájuk tartozó javításokat vagy elkerülésük módszerét.

Gondok az egérrel

Egyes esetekben felfüggesztett állapotból visszatérve az egerünk nem hajlandó mûködni. Ezt úgy lehet elkerülni, ha /boot/loader.conf állományba beírjuk a `hint.psm.0.flags="0x3000"` sort. Ha ez nem segít, akkor a fentieknek megfelelõen küldjünk be egy hibajelentést.

Felfüggesztés/Folytatás

Az ACPI három (STR) állapotban képes a fizikai memória segítségével készenléti módba váltani, ezek az S1-S3, és egy állapotban használja a lemezt (STD), amelyet S4-nek hívnak. Az S5 neve a "szoftveres kikapcsolás", ami egy olyan állapotot takar, amikor a rendszerünk áram alatt van, de még nem üzemel. Az S4BIOS állapot a BIOS segítségével a lemezre menti a rendszert, az S4OS állapotot pedig teljes egészében az operációs rendszer valósítja meg.

A rendszerünk által ismert készenléti módokat a `sysctl hw.acpi` paranccsal ellenõrizhetjük. Íme mindez egy Thinkpad esetén:

```
hw.acpi.supported_sleep_state: S3 S4 S5
hw.acpi.s4bios: 0
```

Ez azt jelenti, hogy az `acpiconf -s` parancs kiadásával kipróbálhatjuk az S3, S4OS, és S5 állapotokat. Ha az `s4bios` értéke egy (1), akkor az S4BIOS támogatása helyett az S4OS állapotot kapjuk.

A felfüggesztés és folytatás kipróbálása során kezdjük az S1 állapottal, már amennyiben az támogatott a rendszerünkön. Ez az állapot többnyire használható, mivel nem igényel túlságosan sok támogatást a meghajtó részérõl. Eddig még senki sem implementálta az S2 állapotot, de ha ezt is tudja a rendszerünk, akkor az S1-hez hasonlót nyerünk vele. A következõ próba az S3 állapoté. Ez a legmélyebb STR állapot, és a hardver megfelelõ újraélesztéséhez rengeteg támogatás szükségeltetik a meghajtó részérõl. Ha gondjaink lennének a rendszerünk felébresztésével, nyugodtan írjunk egy levelet a FreeBSD ACPI levelezési lista listára, ám a probléma gyors megoldódásában ne reménykedjünk, hiszen ehhez még temérdek meghajtón és hardveren kell tesztelni és kell dolgozni.

Felfüggesztés és folytatás esetén gyakori probléma, hogy sok eszközmeghajtó nem menti el, nem állítja vissza vagy éppen nem hozza újra rendesen mûködésbe az adott eszközön található firmware-t, a regisztereket vagy memóriát. Az okok felderítéséhez elõször érdemes a következõket kipróbálni:

```
# sysctl debug.bootverbose=1
# sysctl debug.acpi.suspend_bounce=1
# acpiconf -s 3
```

Ezzel a módszerrel tesztelni tudjuk az összes meghajtó felfüggesztési és folytatási rutinjait anélkül, hogy ténylegesen S3 állapotba helyeznénk az eszközt. Bizonyos esetekben ezzel könnyen elcsíphetõ a hiba (például a firmware állapotának elvesztése, watchdog time out, megállás nélküli újrapróbálkozások). A rendszer ilyenkor nem vált S3 állapotra, vagyis az eszköz nem kerül energiatakarékos állapotba, és eltérõen a valós S3 állapottól továbbra is mûködik még abban az esetben is, amikor a szükséges felfüggesztési és folytatási rutinok teljesen hiányoznak.

Komolyabb esetben további segédeszközökre lesz szükségünk, vagyis soros portra és kábelre a soros vonali nyomkövetéshez, vagy Firewire portra és kábelre a dcons(4) használatához, valamint némi tapasztalatra a rendszermagon belüli hibakeresésben.

A problémát nagy mértékben segíti különválasztani, ha igyekszünk minél több meghajtót kivenni a rendszermagból. Ha így javul a helyzet, akkor már könnyen le lehet szûkíteni arra a meghajtóra a kört, aminek betöltésével esetleg gondok akadhatnak. Általában ilyenek a bináris meghajtók, mint például az nvidia.ko, az X11 megjelenítésért felelõs és az USB eszközök meghajtói, miközben az Ethernet eszközök remekül szoktak mûködni. Ha különösebb gond nélkül képesek vagyunk betölteni és eltávolítani ezeket a meghajtókat, akkor ezt a folyamatot önállósítani is tudjuk úgy, hogy az /etc/rc.suspend és /etc/rc.resume szkriptekbe beillesztjük az ehhez szükséges parancsokat. Ezekben egyébként találunk is egy megjegyzésbe rakott példát a meghajtók betöltésérõl és eltávolításáról. Ha az ébresztés után elszemetelõdik a képernyõ tartalma, akkor állítsuk át a

`hw.acpi.reset_video` változó értékét nullára (`0`). Sokat segíthet meg az is, ha a `hw.acpi.sleep_delay` értékét csökkentjük vagy növeljük.

Megpróbálhatjuk azt is, hogy elindítunk egy frissebb Linux disztribúciót ACPI támogatással és ugyanazon a hardveren kipróbáljuk az általa felkínált felfüggesztési és folytatási lehetõséget. Ha Linux alatt ez megbízhatóan mûködik, akkor nagy a valószínûsége, hogy ez FreeBSD alatt az egyik meghajtó hibájából fakadóan nem használható. Így fokozatosan le is tudjuk szûkíteni, hogy pontosan melyikkel lehet a gond, és ezzel a fejlesztõk munkáját segítjük. Megjegyeznénk, hogy az ACPI-t karbantartó fejlesztõk általában nem foglalkoznak más meghajtókkal (például hangkártya vagy ATA stb.), ezért az adott meghajtóval kapcsolatos hibáról javasolt értesíteni a FreeBSD-CURRENT levelezési lista listát és a meghajtóért felelõs fejlesztõt is. Ha van egy kis kedvünk és idõnk, mi magunk is belebiggyeszthetünk a meghajtóba néhány printf(3) függvényt annak kiderítésére, pontosan hol is fagy le a folytatási funkció.

Végül megpróbálkozhatunk az ACPI kikapcsolásával is, és áttérhetünk helyette az APM használatára. Ha az APM-mel mûködnek a készenléti állapotok, akkor érdemes inkább azzal dolgozni, különösen a régebbi (2000 elõtti) hardverek esetében. A gyártóknak eltartott egy ideig, amíg rendes ACPI támogatást voltak képesek adni, ezért a régebbi hardvereknél inkább a BIOS-nak akadnak gondjai az ACPI-val.

A rendszer lemerevedik (ideiglenesen vagy teljesen)

A legtöbb rendszer olyankor akad meg, amikor sok megszakítás elveszik, vagy amikor éppen sok megszakítás érkezik egyszerre. A chipkészleteknek számos baja származik abból, hogy a BIOS milyen módon állítja be a rendszer indítása elõtt a megszakításokat, mennyire helyes az APIC (MADT) táblázata és hogyan vezérli a *Rendszervezérlõ megszakítást* (System Control Interrupt, SCI).

A megszakítás-viharok a `vmstat -i` parancs kimenetében szereplõ elveszett megszakításokból azonosíthatók be, ahol keressünk rá az `acpi0` sorra. Ha ez a számláló másodpercenként kettõnél többel növekszik, akkor a megszakításaink viharba keveredtek. Ha a rendszer látszólag lefagyott, próbáljuk meg elõhívni a DDB-t (konzolban a `CTRL` + `ALT` + `ESC`) és gépeljük be, hogy `show interrupts`.

A megszakítási problémákkal kapcsolatban egyetlen reményünk az APIC támogatás kikapcsolása lehet a loader.conf állományban a `hint.apic.0.disabled="1"` sor hozzáadásával.

Végzetes hibák

Az ACPI-vel kapcsolatos végzetes hibák viszonylag ritkák, és javításuk a legfontosabb. Ilyenkor az elsõ teendõnk elkülöníteni a hiba reprodukálásának egyes lépéseit és (ha lehetséges) lekérni a hívási láncot. Kövessük az `options DDB` és a soros vonali konzol beállításához adott tanácsokat (lásd A DDB elérése a soros vonalról) vagy hozzunk létre egy dump(8) partíciót. A DDB-ben a hívási láncot a `tr` parancs segítségével kérhetjük le. Ha kézzel írjuk le láncot, akkor legalább az alsó öt (5) és a felsõ öt (5) sorát mindenképpen jegyezzük fel!

Ezután próbáljuk meg úgy szûkíteni a probléma lehetõségét, hogy az ACPI használata nélkül indítjuk a rendszert. Ha ezzel nincs semmi gond, akkor a `debug.acpi.disable` változó értékének megfelelõ beállításával egyenként meg tudjuk figyelni az ACPI alrendszer egyes részeit. Ehhez példákat az acpi(4) man oldalon találunk.

Felfüggesztés vagy leállítás után elindul a rendszer

Elõször is próbáljuk meg a `hw.acpi.disable_on_poweroff` változó értékét `0`-ra állítani a loader.conf(5) állományban. Ezzel távoltartjuk az ACPI alrendszert a rendszer leállítási folyamatától. Egyes rendszereknek valamilyen okból kifolyólag szükségük van itt az `1` (az alapértelmezett) értékre. Ez többnyire megoldja a problémát, amikor a rendszer váratlanul elindul a készenléti mód aktiválásákor vagy kikapcsoláskor.

Egyéb problémák

Ha más gondjaink lennének az ACPI-val (dokkoló állomásunk van, egyes eszközöket nem vesz észre stb.), akkor természetesen errõl is küldjünk egy leírást a levelezési listára. Azonban vegyük figyelembe, hogy egyes problémák a ACPI alrendszer eddig még nem implementált, befejezetlen részeihez kötõdnek, ezért azok megoldása még várat magára. Kérünk mindenkit, hogy legyen türelemmel és álljon készen a kiküldött javítások tesztelésére!

11.16.4. ASL, `acpidump` és IASL

A problémák leggyakoribb forrása, hogy a BIOS-gyártók rossz (vagy kifejezetten hibás!) bytekódokat adnak. Ez általában a következõhöz hasonló rendszerüzenetbõl derül ki:

```
ACPI-1287: *** Error: Method execution failed [\\_SB_.PCI0.LPC0.FIGD._STA] \\
(Node 0xc3f6d160), AE_NOT_FOUND
```

Az ilyen jellegû hibákat gyakran úgy lehet orvosolni, ha a BIOS-unkat frissítjük a legújabb verzióra. A legtöbb ilyen üzenet teljesen ártalmatlan, de ha vannak más problémáink is, például az akkumulátor állapota nem olvasható le, akkor elõször az AML környékén érdemes kutakodnunk. A bytekód, más néven AML, az ASL elnevezésû forrásnyelvbõl származik. Az AML egy DSDT néven ismert táblázatban található meg. Az ASL másolatát az acpidump(8) paranccsal készíthetjük el. Paraméterként egyaránt adjuk meg a `-t` (megmutatja a rögzített táblák tartalmát) és `-d` (visszafejti az AML kódokat az ASL nyelvére) kapcsolókat. A felírás pontos formátumát a A nyomkövetési információk beküldése címû szakaszban olvashatjuk.

Elsõként próbáljuk meg újrafordítani az így nyert ASL programot és keressünk benne hibákat. A figyelmeztetések általában nyugodtan figyelmen kívül hagyhatóak, azonban a hibák olyan implementációs hibákra utalnak, amelyek akadályozzák az ACPI helyes mûködését. Az ASL újrafordítását az alábbi paranccsal tudjuk elvégezni:

```
# iasl saját.asl
```

11.16.5. Az ASL kijavítása

Végeredményben az a célunk, hogy az ACPI megfelelõ mûködéséhez senkinek se kelljen hozzányúlnia semmihez. Azonban még mindig szükség van BIOS-gyártók által elkövetett gyakori hibák elkerülésének kifejlesztésére. A Microsoft® értelmezõje (acpi.sys és acpiec.sys) nem ellenõrzi szigorúan a szabvány szerinti megfelelést, ezért számos olyan BIOS-gyártó, akik csak Windows® alatt tesztelik az ACPI implementációjukat, soha nem fogják kijavítani a ASL kódjukban ejtett

hibáikat. Reménykedünk, hogy folyamatosan sikerül felderíteni és dokumentálni a Microsoft® értelmezõje által eltûrt szabványon kívüli viselkedést és leutánozni FreeBSD alatt is, hogy így ne kelljen a felhasználóknak kézzel a saját ASL forrásaikat javítgatni. Az ebbõl fakadó hibákat úgy tudjuk elkerülni és segíteni a fejlesztõknek azonosítani a hozzá társuló viselkedést, hogy magunk javítjuk az ASL-ben felfedezett hibákat. Ha ez beválik, akkor küldjük el a régi és új ASL közti diff(1) -et a fejlesztõknek, akik így majd az ACPI-CA-ban ki tudnak dolgozni egy megoldást a hibás viselkedésre, ezzel a javításunk szükségtelenné válik.

Most pedig következzenek a legismertebb hibaüzenetek, az okaik és javításuk:

Operációs rendszeri függõségek

Néhány AML úgy gondolja, hogy a világ csak a különbözõ Windows® verziókból áll. A FreeBSD-nek megadható, hogy másik operációs rendszernek adja ki magát, és ezzel talán meg is szüntethetõ pár hiba. Ezt a legegyszerûbb úgy tudjuk megtenni, ha a /boot/loader.conf állományhoz hozzáfûzzük a `hw.acpi.osname="Windows 2001"` sort, vagy itt egy olyan karakterláncot adunk meg, amit az ASL forrásban láttunk.

Hiányzó visszatérési érték

Bizonyos módszerek a szabvány szerint elvártaktól eltérõen nem adnak vissza explicit módon értéket. Mivel az ACPI-CA ezt nem kezeli le, ezért a FreeBSD részérõl tartalmaz egy olyan módosítást, amivel implicit módon is vissza lehet adni értéket. Ha biztosak akarunk lenni a visszaadni kívánt értékben, akkor helyezzünk el a megfelelõ helyekre explicit Return utasításokat. Az `iasl` a `-f` paraméterrel kényszeríthetõ az ilyen ASL források lefordítására.

Az alapértelmezett AML felülbírálása

Miután módosítottuk a saját.asl állományunkat, így tudjuk lefordítani:

```
# iasl saját.asl
```

Az `-f` kapcsoló megadásával kikényszeríthetjük az AML létrehozását még abban az esetben is, amikor hibákat tartalmaz. Ügyeljünk rá, hogy bizonyos hibákat (például a hiányzó visszatérési értékeket) a fordító magától kikerül.

Az `iasl` alapértelmezett kimenete a DSDT.aml állomány. A /boot/loader.conf átírásával így tudjuk ezzel helyettesíteni a BIOS-unk hibás változatát (ami még mindig megtalálható a flash memóriában):

```
acpi_dsdt_load="YES"
acpi_dsdt_name="/boot/DSDT.aml"
```

Ehhez ne felejtsük el a saját DSDT.aml állományunkat bemásolni a /boot könyvtárba.

11.16.6. Nyomkövetési információk kinyerése az ACPI-bõl

Az ACPI meghajtója nagyon rugalmas nyomkövetési lehetõségekkel rendelkezik. Ennek révén ugyanúgy megadhatjuk a nyomonkövetni kívánt alrendszert, mint ahogy annak mélységét is. A nyomkövetni kívánt alrendszereket "rétegekként" adjuk meg, valamint ezek ACPI-CA komponensekre (ACPI_ALL_COMPONENTS) és ACPI hardvertámogatásra (ACPI_ALL_DRIVERS) bomlanak le. A nyomkövetéskor keletkezõ kimenet részletességét a "szintként" adjuk meg, ami az ACPI_LV_ERROR-tól (csak a hibák) ACPI_LV_VERBOSE-ig (minden) terjedhet. A "szint" itt egy bitmaszk, ezért szóközzel elválasztva egyszerre több beállítás megadható. Ha túlságosan sok üzenet érkezik a konzol üzenetpufferébe, akkor szükségünk lehet a soros konzol keresztüli nyomkövetésre is. Az összes szint és réteg az acpi(4) man oldalon található meg.

A nyomkövetés alapértelmezés szerint nem engedélyezett. Az engedélyezéséhez hozzá kell adnunk az `options ACPI_DEBUG` sort a rendszermagunk beállításait tartalmazó állományhoz, amennyiben a rendszermagba fordítjuk az ACPI támogatást. Ha az /etc/make.conf állományba írjuk bele az `ACPI_DEBUG=1` sort, akkor azt globálisan engedélyezhetjük. Ha modulként használjuk, elegendõ csak a következõ módon újrafordítani az acpi.ko modult:

```
# cd /sys/modules/acpi/acpi
&& make clean &&
make ACPI_DEBUG=1
```

Telepítsük fel a acpi.ko modult a /boot/kernel könyvtárba és állítsuk be a számunkra megfelelõ szintet és réteget a loader.conf állományban. Az alábbi példában engedélyezzük az összes ACPI-CA komponens és az összes ACPI hardvermeghajtó (processzor, LID stb.) nyomkövetését. Csak a hibaüzeneteket írja ki részletesen.

```
debug.acpi.layer="ACPI_ALL_COMPONENTS ACPI_ALL_DRIVERS"
debug.acpi.level="ACPI_LV_ERROR"
```

Ha az általunk keresett információt egy adott esemény váltja ki (például egy felfüggesztés vagy egy ébresztés), akkor nem is fontos átírnunk hozzá a loader.conf állományt, hanem helyette a rendszer indítása után használjuk a `sysctl` parancsot a réteg és a szint megadására akkor, amikor a rendszert felkészítjük az eseményre. A `sysctl` változókat ugyanúgy nevezték el, mint a loader.conf állományban található beállításokat.

11.16.7. Hivatkozások

Az ACPI-rõl az alábbi helyeken találunk részletesebb információkat:

- A FreeBSD ACPI levelezési lista
- Az ACPI levelezési lista archívuma: http://lists.freebsd.org/pipermail/freebsd-acpi/
- A korábbi ACPI levelezési lista archívuma: http://home.jp.FreeBSD.org/mail-list/acpi-jp/
- Az ACPI specifikációja
- A FreeBSD következõ man oldalai: acpi(4), acpi_thermal(4), acpidump(8), iasl(8), acpidb(8)

- A DSDT nyomkövetése (angolul). (Példának a Compaqot hozza fel, de általánosságban véve hasznos.)

Chapter 12. A FreeBSD rendszerindítási folyamata

12.1. Áttekintés

A számítógép indulását és a rajta található operációs rendszer betöltõdését "rendszerindítási folyamatnak" nevezzük, vagy egyszerûen csak "bootolásnak". A FreeBSD rendszerindítási folyamata nagymértékû rugalmasságot kínál a rendszer indulását követõ események vezérlését illetõen, legyen az a számítógépre telepített különféle operációs rendszerek egyikének kiválasztása, vagy pedig ugyanazon operációs rendszer valamelyik változatának vagy rendszermagjának kiválasztása.

Ez a fejezet részleteiben bemutatja a rendszerindításhoz kapcsolódó konfigurációs opciókat, illetve a FreeBSD bootolásának testreszabhatóságát. Ebbe minden beleértendõ, ami a FreeBSD rendszermag beindulása és az eszközök keresése során történik, majd az init(8) elindításával zárul. Ha nem vagyunk teljesen biztosak benne, ez pontosan mikor is következik be, figyeljük, amikor a szöveg színe fehérrõl szürkére vált.

A fejezet elolvasása során megismerjük:

- milyen elemekbõl áll a FreeBSD rendszertöltõ alrendszere, és ezek miként kapcsolódnak egymáshoz;
- melyek azok a FreeBSD rendszerindításában résztvevõ elemeknek átadható opciók, amelyekkel vezérelhetõ ez a folyamat;
- a device.hints(5) alapjait.

Csak x86

Ez a fejezet kizárólag csak az Intel® x86 típusú architektúráján futó FreeBSD rendszerindítási folyamatát mutatja be.

12.2. A rendszerindítás problémája

Az operációs rendszer elindítása a számítógép bekapcsolása után egy felettébb érdekes problémát vet fel. Definíció szerint a számítógép ugyanis egy lépést sem tud megtenni az operációs rendszer elindulása nélkül. Például nem tud programokat futtatni a lemezrõl. Eszerint ha a számítógépünk nem képes programokat futtatni a lemezrõl az operációs rendszer segítsége nélkül, viszont az operációs rendszer programjai a lemezen vannak, mégis hogyan képes elindulni maga az operációs rendszer?

Maga a probléma a Münchhausen báró kalandjai c. könyvben leírtakhoz hasonló. A történet szerint ugyanis a fõszereplõ egy mocsárban ragadt derék lovával, azonban sikerült kihúznia magát belõle a saját hajánál fogva. Ez a motívum vált a számítógépek hõskorában a *rendszerbetöltés* alapjává, vagyis ahogyan betöltötték az operációs rendszereket. *(Ford.: ezt az angolban bootstrappingnek hívják, mivel a történet angol változata szerint a csizmáján (boot) emelkedett ki. Ebbõl alakult ki késõbb az elterjedt bootolás szó is.)*

Az x86-os konfigurációkon a BIOS (Basic Input/Output System, avagy "alapvetõ be- és kimeneti rendszer") felelõs az operációs rendszer betöltéséért. Ehhez a BIOS elõször megkeresi a merevlemez egy speciális helyén található Master Boot Record-ot (MBR). A BIOS elegendõ tudással rendelkezik az MBR beolvasásához és lefuttatásához, és feltételezi, hogy az MBR majd elvégzi az operációs rendszer betöltéséhez szükséges további feladatokat, helyenként a BIOS közremûködésével.

Az MBR-ben található programkódot hívják általában *boot manager*nek, kiváltképp abban az esetben, amikor az a felhasználóval is kommunikál. Ilyenkor a boot manager többnyire további kódot tartalmaz a lemez elsõ *sáv*ján vagy az egyik állományrendszerben. (A boot managereket néha *boot loader*nek is nevezzük, de a FreeBSD-s terminológia ezt a kifejezést a rendszerindítás egy késõbbi fokozatára használja.) Népszerûbb boot managerek: boot0 (avagy Boot Easy, a FreeBSD alapvetõ boot managere), GRUB, GAG és a LILO. (Ezek közül egyedül csak a boot0 fér el az MBR-ben.)

Amennyiben merevlemezeinken csupán egyetlen operációs rendszer foglal helyet, akkor egy szabványos MBR tökéletesen megfelelõ. Ez az MBR megkeresi az elsõ indítható (más néven aktív) slice-ot a lemezen, majd lefuttatja a benne található indítókódot az operációs rendszer többi részének felélesztéséhez. Az fdisk(8) által alapértelmezés szerint telepített MBR pontosan ilyen. Ennek alapja a /boot/mbr állomány.

Ha viszont több operációs rendszert is telepítettünk a lemezeinkre, akkor egy ettõl eltérõ boot managert érdemes használnunk, olyat, amely képes felsorolni a rendelkezésre álló operációs rendszereket, lehetõvé téve, hogy választani lehessen az indításuk között. Ezek közül kettõrõl esik szó a következõ alfejezetekben.

A FreeBSD rendszertöltõ alrendszerének fennmaradó része három fokozatra bontható. Az elsõ fokozatot az MBR indítja el, amely pontosan eleget tud ahhoz, hogy a számítógépet egy elõre megadott állapotba hozza és lefuttassa rajta a második fokozatot. A második fokozat ennél már egy kicsivel többre képes, majd ezt követi a harmadik fokozat. Ez a fokozat zárja le végül az operációs rendszer betöltésének feladatát. A munka tehát ezen három fokozat között oszlik meg, mivel a PC-szabványok komoly korlátozásokat tesznek az elsõ, illetve második fokozatban futtatható programok méretére. Ha így fûzzük össze a feladatokat, akkor a FreeBSD számára egy sokkal rugalmasabb betöltõt kapunk.

Ezután beindul a rendszermag (más néven kernel), és nekilát a számítógépben rendelkezésre álló hardvereszközök keresésének, majd elõkészíti õket a használatra. Ahogy a rendszermag beindításának folyamata véget ért, az átadja a vezérlést az init(8) nevû felhasználói programnak, amely megbizonyosodik a lemezek használhatóságáról. Az init(8) ezt követõen megkezdi az erõforrások felhasználói szintû konfigurálását: csatlakoztatja az állományrendszereket, beállítja a hálózati kártyá(ka)t, és elindítja mindazon programokat, amelyeknek egy FreeBSD rendszer indulásakor futnia kell.

12.3. A boot manager és az indulás fokozatai

12.3.1. A boot manager

Az MBR-ben található programkódot, avagy boot managert, sokszor csak a rendszerindítás *nulladik*

*fokozat*aként emlegetik. Ez az alfejezet a korábban említett két boot managert tárgyalja: a boot0-t és a LILO-t.

A boot0 boot manager: A FreeBSD telepítõje vagy a boot0cfg(8) által kialakított MBR alapértelmezett állapotban a /boot/boot0 állományon alapszik. (A boot0 program nagyon egyszerû, hiszen az -ben elhelyezhetõ kód csak 446 byte hosszúságú lehet, mert a végében még el kell férnie a slice-táblának és az `0x55AA` azonosítónak.) Ha telepítettük a boot0-t és a lemezeinken több operációs rendszer is megtalálható, akkor a rendszerindítás során egy hasonló képet kell látnunk:

Példa 7. A boot0 munkában

```
F1 DOS
F2 FreeBSD
F3 Linux
F4 ??
F5 Drive 1

Default: F2
```

Más operációs rendszerek, különösen a Windows®, telepítésük során felülírják a már meglevõ MBR-t a sajátjukkal. Ha ez történne, vagy egyszerûen csak szeretnénk a meglevõ MBR-t lecserélni a FreeBSD MBR-jével, adjuk ki a következõ parancsot:

```
# fdisk -B -b /boot/boot0 eszköznév
```

ahol az *eszköznév* annak az eszköznek a neve, ahonnan a rendszert indítani szeretnénk, tehát például ad0 az elsõ IDE-lemez esetén, vagy ad2 a második IDE-vezérlõn található elsõ IDE-lemez esetén, illetve da0 az elsõ SCSI-lemez esetén, és így tovább. Ha testre akarjuk szabni az MBR-t, használjuk a boot0cfg(8)-t.

A LILO boot manager: Ezen boot manager telepítéséhez és beállításához elsõként indítsuk el a Linuxot és vegyük hozzá az alábbi sort a rendszerünkben található /etc/lilo.conf konfigurációs állományhoz:

```
other=/dev/hdXY
table=/dev/hdX
loader=/boot/chain.b
label=FreeBSD
```

A fenti sablont kiegészítve, a linuxos konvenciók szerint adjuk meg a FreeBSD elsõdleges partícióját és meghajtóját úgy, hogy az *X*-et átírjuk a linuxos meghajtó betûjelére és az *Y*-t átírjuk a Linux® elsõdleges partíciójának számára. Ha SCSI-meghajtót használunk, a */dev/hd* részt is át kell írnunk az elõbbiek mellett */dev/sd*-re. A `loader=/boot/chain.b` sor elhagyható abban az esetben, ha mind a két operációs rendszer ugyanazon a meghajtón található. Ha befejeztük a módosítást, futtassuk le a `/sbin/lilo -v` parancsot a változtatásaink életbe léptetéséhez. Ezt ellenõrizhetjük is a képernyõn

megjelenõ üzenetek alapján.

12.3.2. Az elsõ fokozat (/boot/boot1) és a második fokozat (/boot/boot2)

Az elsõ és a második fokozat fogalmilag ugyanannak a programnak a része, a lemezen ugyanott helyezkedik el. A tárbeli megszorítások miatt ugyan el kellett választani õket egymástól, de a telepítésük mindig egy helyre történik. A telepítõ vagy a bsdlabel (lásd lentebb) használata során a /boot/boot nevû kombinált állományból másolódnak ki.

Az állományrendszereken kívül találhatóak, az aktív slice elsõ sávjában, annak elsõ szektorától kezdõdõen. Ez az a hely, ahol a boot0, illetve a többi boot manager is keresi a rendszerindítás folytatására alkalmas programot. A felhasznált szektorok száma könnyedén kideríthetõ a /boot/boot méretébõl.

Legfeljebb 512 byte-os méreténél fogva a boot1 állomány nagyon egyszerû felépítésû, és éppen csak annyit tud a slice-ra vonatkozó információkat tároló FreeBSD *bsdlabel*-rõl, hogy megtalálja a boot2-t és elindítsa.

A boot2 már egy kicsivel ügyesebb, és eléggé ismeri a FreeBSD állományrendszerét ahhoz, hogy megtaláljon rajta állományokat, valamint képes egy egyszerû felületet nyújtani a rendszermag vagy a betöltõ megválasztásához.

Mivel a betöltõ pedig már ennél is okosabb, és egy könnyen használható rendszerindítási konfigurációt tud a felhasználó számára nyújtani, ezért a boot2 általában ezt indítja el, de elõtte közvetlenül a rendszermag futtatását végzi el.

Példa 8. A boot2 mûködés közben

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

Ha le kellene váltani a korábban telepített boot1 és boot2 fokozatokat, használjuk a bsdlabel(8)-t:

```
# bsdlabel -B lemezslice
```

ahol a *lemezslice* annak a lemeznek és slice-nak a kombinációja, ahonnan indítjuk a rendszerünket, például az elsõ IDE-lemez elsõ slice-a esetén ez az ad0s1.

A veszélyesen dedikált mód (Dangerously Dedicated Mode)

Amikor a bsdlabel(8) meghívásakor csak a lemez nevét használjuk, például ad0-t, a parancs egy veszélyesen dedikált lemezt hoz létre, slice-ok nélkül! Szinte biztos, hogy nem ez az, amire szükségünk lenne, ezért mindig ellenõrizzük kiadása elõtt a bsdlabel(8) parancsot!

12.3.3. A harmadik fokozat (/boot/loader)

A betöltõ a három fokozatú rendszertöltés utolsó állomása. Az állományrendszerben /boot/loader néven találhatjuk meg.

A rendszertöltõt az egyszerû konfigurálhatóságot támogató, felhasználóbarát eszköznek tervezték, és könnyen megtanulható, beépített parancsokat használ, melyek mögött egy összetettebb parancsokat ismerõ, erõsebb értelmezõ áll.

A rendszertöltõ mûködése

Az inicializálás során a rendszertöltõ megpróbálja megkeresni a konzolt, és a lemezek közül igyekszik megtalálni azt, amelyikrõl elindult a rendszer. A keresések eredményének megfelelõen beállítja a változókat, majd elindul egy értelmezõ, ahol vagy szkriptbõl olvasva, vagy pedig interaktívan feldolgozásra kerülnek a parancsok.

A rendszertöltõ ezt követõen beolvassa a /boot/loader.rc állományt, az pedig alapértelmezés szerint feldolgozza a /boot/defaults/loader.conf állományt, ahol a változók értelmes kezdõértéket kapnak, valamint feldolgozza még a /boot/loader.conf állományt is, ahol a változók értékeit változtathatjuk meg. Miután ez lezajlott, a loader.rc a változók értékeinek megfelelõen cselekszik, betöltve az ily módon kiválasztott rendszermagot és a hozzá választott modulokat.

Végezetül, a rendszertöltõ beiktat egy, alapértelmezés szerint 10 másodperces várakozási szünetet, majd elindítja a rendszermagot, ha azt meg nem szakítjuk egy billentyû lenyomásával. Ha megszakítjuk ezt a várakozást, a rendszertöltõ egy parancssort ad, amelyen keresztül egyszerû parancsokat adhatunk ki neki: állíthatjuk a változók értékeit, modulokat távolíthatunk el a memóriából, modulokat töltethetünk be, elindíthatjuk a rendszert vagy újraindíthatjuk a számítógépet.

A rendszertöltõ beépített parancsai

Következzenek a leggyakrabban használt parancsok a rendszertöltõben. Az összes itt elérhetõ parancsot a loader(8) man oldalon találjuk meg.

autoboot *másodperc*

Megkezdi a rendszermag betöltését, ha nem szakítjuk meg a várakozást másodpercekben megadott idõtartam alatt. Ekkor egy visszaszámlálást láthatunk, ami az alapértelmezés szerint 10 másodperctõl indul.

boot [-opciók] [rendszermag]

Amennyiben léteznek, a megadott opciókkal azonnal megkezdi a megadott rendszermag betöltését. A *rendszermag* paraméter csak abban az esetben adható meg, ha elõtte kiadtunk egy *unload* parancsot, máskülönben a korábban betöltött rendszermaggal indul a rendszer.

boot-conf

Végigmegy a modulok ugyanazon automatikus konfigurációján, ahogy az a normális rendszerindítás során is történik. Ezen parancs használatának csak akkor van értelme, ha elõtte az `unload` parancsot használjuk, megváltoztatunk egy-két változót, általában a `kernel`-t.

help [témakör]

A /boot/loader.help állományban fellelhetõ súgóüzeneteket mutatja meg. Ha témakörnek `index`et adunk meg, akkor az elérhetõ témakörök listáját kapjuk meg.

include *állománynév* ...

Feldolgozza a megnevezett állományt: beolvassa, majd sorról sorra értelmezi. Hiba esetén azonnal megállítja a feldolgozást.

load [-t típus] *állománynév*

A név alapján betölti a rendszermagot, modult vagy az adott típusú állományt. Az állománynév után megadott további paraméterek az állománynak adódnak át.

ls [-l] [elérési útvonal]

Kilistázza a megadott elérési útvonalon található állományokat, vagy ennek hiányában a gyökér tartalmát. Ha hozzátesszük a `-l` kapcsolót, az állományok mérete is látható válik.

lsdev [-v]

Kilistázza az összes olyan eszközt, ahonnan modulokat tölthetünk be. Amennyiben a `-v` kapcsolót is megadjuk, további részleteket tudhatunk meg róluk.

lsmod [-v]

Kilistázza a betöltött modulokat. Ha többet szeretnénk megtudni róluk, adjuk meg a `-v` kapcsolót.

more *állománynév*

Megmutatja a megadott állomány tartalmát, minden `LINES` számú sor után szünetet tartva.

reboot

Azonnal újraindítja a számítógépet.

set *változó*

Beállítja a rendszertöltõ környezeti változójának értékét.

unload

Eltávolítja a memóriából az összes betöltött modult.

Rendszertöltõ példák

Íme néhány konkrét példa a rendszertöltõ használatára:

- Így indíthatjuk egyfelhasználós módban az általunk használt rendszermagot:

```
boot -s
```

- Távolítsuk el a betöltött rendszermagot és a moduljait, és töltsük be helyettük a korábbi (vagy egy másik) rendszermagot:

```
unload
```

```
load kernel.old
```

Itt használhatjuk a kernel.GENERIC nevet is, amely a telepítõlemezen található általános rendszermagra utal, vagy a kernel.old nevet, amely a korábban használt rendszermagot rejti (például amikor rendszermagot frissítettünk vagy készítettünk magunknak).

A következõképpen lehet betölteni a szokásos moduljainkat egy másik rendszermaggal:

```
unload
set kernel="kernel.old"
boot-conf
```

- Egy rendszermag-konfigurációs szkript (automatizált szkript, amely ugyanazokat a beállításokat végzi el, amelyeket mi magunk tennénk akkor, amikor a rendszermagot indítjuk) betöltése:

```
load -t userconfig_script /boot/kernel.conf
```

Rendszerbetöltõ képernyõk

A rendszertöltés során megjelenõ rendszerüzenetek megjelenítése helyett egy sokkal megnyerõbb, látványosabb rendszerindítást tudunk elérni betöltõ képernyõk használatával. Egy ilyen képet egészen a konzolos bejelentkezésig, vagy az X felett futó valamelyik bejelentkezõ képernyõ megjelenéséig láthatunk.

FreeBSD alatt alapvetõen két típusú környezet létezik. Ezek közül az egyik a hagyományos virtuális konzolos parancssoros felület. Ekkor a rendszertöltés befejezõdésekor egy szöveges parancssori bejelentkezõ promptot kapunk. A másik környezet az X11 által felkínált grafikus felület. Miután telepítettük az X11 szervert és valamelyik munkakörnyezetet, tehát például a GNOME, a KDE vagy az XFce környezetek valamelyikét, a `startx` paranccsal indíthatjuk el a grafikus felületet.

Némely felhasználók a megszokott szöveges bejelentkezés helyett is inkább valamelyik X11 alapú grafikus bejelentkezést szeretnék használni. A különbözõ bejelentkezõ képernyõk, mint amilyen az Xorg esetén az XDM, a GNOME esetén a gdm, vagy a KDE esetén a kdm (illetve a Portgyûjteménybõl származó egyéb megoldások) alapvetõen a konzolos bejelentkezés helyett nyújtanak egy grafikus bejelentkezõ felületet. Ilyenkor a sikeres bejelentkezést követõen a felhasználó közvetlenül egy grafikus környezetbe kerül.

A parancssoros felület esetén a rendszertöltõ képernyõ elrejti az összes rendszerüzenetet és a rendszer indításakor futtatott programok üzeneteit. Az X11 használata esetén azonban a felhasználók ezzel együtt már a többi, alapértelmezés szerint grafikus felülettel rendelkezõ rendszerhez (Microsoft® Windows® vagy más nem-UNIX operációs rendszer) hasonló élményt nyernek.

A rendszerbetöltõ képek támogatása

A FreeBSD csak BMP (.bmp) vagy ZSoft PCX formátumú, 256 színû rendszerbetöltõ képek megjelenítését támogatja. Emellett szabványos VGA kártyákon csak akkor fog mûködni, ha a kép 320x200 vagy annál kisebb felbontású.

Nagyobb méretû képek esetén, egészen az 1024x768-as felbontásig, a FreeBSD VESA támogatására lesz szükségünk. Ezt vagy a rendszer indításakor a VESA modul betöltésével engedélyezhetjük, vagy ha a rendszermag konfigurációs állományában megadjuk a `VESA` sort és készítünk egy saját rendszermagot (lásd A FreeBSD rendszermag testreszabása). A VESA támogatáson keresztül a felhasználók a teljes képernyõt betöltõ rendszerbetöltõ képeket is meg tudnak így jeleníteni.

A rendszerbetöltõ képernyõ a rendszer indítása közben bármikor tetszõlegesen kikapcsolható egy tetszõleges billentyû lenyomásával.

A megadott betöltõképernyõ alapértelmezés szerint a képernyõvédõ szerepét is betölti az X11 felületén kívül. Ha tehát egy ideig nem használjuk a számítógépünket, akkor a képernyõ átvált a betöltõképre és folyamatosan változtatni kezdi az intenzitását, a nagyon világosból a nagyon sötétbe, majd újrakezdi. Az alapértelmezett képernyõvédõ az /etc/rc.conf állományban a `saver=` sor megadásával állítható át. Ehhez a beállításhoz több különbözõ beépített képernyõvédõ tartozik, ezek teljes listáját a splash(4) man oldalon olvashatjuk. Ezek közül az alapértelmezett a "warp". Az /etc/rc.conf állományban megadható `saver=` csak a virtuális konzolokra vonatkozik, az X11 bejelentkezõ képernyõire semmilyen hatással sincs.

A rendszerbetöltõ néhány üzenete, valamint a rendszerindítási opciókat tartalmazó menü és a hozzá tartozó visszaszámlálás még a rendszerbetöltõ képernyõ használata során is meg fog jelenni.

A http://artwork.freebsdgr.org címen találhatunk néhány ilyen betöltõképernyõt. A sysutils/bsd-splash-changer port telepítésével pedig a rendszer egyes indításakor egy elõre megadott gyûjteménybõl tudunk véletlenszerûen választani egyet.

A rendszerbetöltõ képek használata

A betöltõképet tartalmazó (.bmp vagy .pcx kiterjesztésû) állományt a rendszerindító partícióra, például a /boot könyvtárba kell tennünk.

A normál (256 szín, legfeljebb 320x200-as felbontású) képek esetén a következõ sorokat adjuk hozzá a /boot/loader.conf állományhoz:

```
splash_bmp_load="YES"
bitmap_load="YES"
bitmap_name="/boot/betöltõkép.bmp"
```

Nagyobb felbontás esetén (legfeljebb 1024x768-as méretig) pedig a /boot/loader.conf állománynak a következõket kell tartalmaznia:

```
vesa_load="YES"
splash_bmp_load="YES"
bitmap_load="YES"
```

```
bitmap_name="/boot/betöltõkép.bmp"
```

Az iménti példában feltételeztük, hogy a /boot/betöltõkép.bmp állományt használjuk betöltõképként. Amikor azonban PCX állományokat akarunk használni, a következõ sorokat kell megadnunk, a felbontástól függõen a `vesa_load="YES"` sorral kiegészítve:

```
splash_pcx_load="YES"
bitmap_load="YES"
bitmap_name="/boot/betöltõkép.pcx"
```

Természetesen a kép neve sem csak "betöltõkép" lehet. Tetszõlegesen elnevezhetjük, egyedül csak arra kell ügyelünk, hogy BMP vagy PCX formátumú legyen: splash_640x400.bmp vagy például blue_wave.pcx.

További érdekes beállítások a loader.conf állományból:

`beastie_disable="YES"`

Ennek megadásakor nem jelenik meg a rendszerindítási lehetõségeket felkínáló menü, de a visszaszámlálás megmarad. Hiába tiltjuk le a menüt, ilyenkor továbbra is választanunk kell a lehetõségek közül.

`loader_logo="beastie"`

Ezzel a beállítással a menüben látható "FreeBSD" feliratot cserélhetjük le a korábbi kiadásokban szereplõ színes démonos emblémára.

12.4. Kapcsolat a rendszermaggal a rendszerindítás folyamán

Ahogy sikerült betölteni (a szokásos módon) a rendszertöltõvel vagy (a rendszertöltõ átugrásával) a boot2 segítségével, a rendszermag megvizsgálja az esetlegesen átvett rendszerindítási paramétereket, és azoknak megfelelõen viselkedik.

12.4.1. A rendszermag paraméterei

A rendszermag leginkább használt paraméterei:

`-a`

a rendszermag inicializálása során rákérdez a gyökér állományrendszerként csatlakoztatandó eszközre.

`-C`

a rendszer indítása CD-rõl.

`-c`

a UserConfig, a rendszerindítás során használt rendszermag-beállító, futtatása.

`-s`

a rendszer indítása egyfelhasználós módban.

`-v`

részletesebb információk megjelenítése a rendszermag indítása során.

Ezeken kívül még számos paraméter létezik, a teljes listát a boot(8) man oldalon találhatjuk meg.

12.5. Eszköz útmutatók (device.hints)

Ez a lehetõség csak a FreeBSD 5.0 vagy annál késõbbi verzióiban jelenik meg.

A rendszerindítás kezdeti szakaszában a loader(8) beolvassa a device.hints(5) állományt. Ebben az állományban tárolódnak a gyakran csak "eszköz útmutatóknak" nevezett változók, amelyek a rendszermag számára nyújtanak hasznos információkat az indulás során. Ezeket az "útmutatókat" az eszközmeghajtók hasznosítják az általuk ismert eszközök beállítása során.

Az eszközökre vonatkozó ilyen jellegû útmutatások a harmadik fázisban megjelenõ parancssorban is megadhatóak. A változókat a `set` (beállít) parancs segítségével tudjuk felvenni, míg az `unset` (eltávolít) paranccsal tudunk törölni, valamint a `show` (megmutat) paranccsal megjeleníteni az értéküket. Sõt, ezen a ponton a /boot/device.hints állománnyal már beállított változókat is felülbírálhatjuk. A rendszerindító parancssorában elvégzett módosítások viszont nem fognak megmaradni, és a következõ rendszerindítás alkalmával elvesznek.

Ahogy a rendszerünk használatra kész állapotba került, a kenv(1) parancs használható a változók értékeinek listázásához.

A /boot/device.hints állományban soronként egy-egy változót tudunk megadni, illetve a kettõskereszttel ("#") bevezetve megjegyzéseket illeszthetünk bele. A sorok szerkezete az alábbi:

```
útmutató.meghajtó.egység.kulcsszó="érték"
```

A harmadik fázisban pedig így adhatjuk meg:

```
set útmutató.meghajtó.egység.kulcsszó=érték
```

Itt a `meghajtó` az eszközmeghajtó neve, az `egység` az eszközmeghajtó által kezelt egyik egység sorszáma, a `kulcsszó` pedig az útmutatáshoz tartozó kulcsszó. Ez a következõk egyike lehet:

- `at`: az útmutatás az eszköz által használt buszra vonatkozik.
- `port`: az útmutatás az eszköz által használt I/O-címre vonatkozik.
- `irq`: az útmutatás az eszköz által használt megszakítás sorszámára vonatkozik.
- `drq`: az útmutatás az eszköz által használt DMA-csatorna sorszámára vonatkozik.

- `maddr`: az útmutatás az eszköz által használt fizikai memóriaterület kezdõcímére vonatkozik.
- `flags`: az eszközhöz tartozó bitek beállítása.
- `disabled`: ha az értéke 1, akkor az adott eszköz használatát letiltjuk.

Az eszközmeghajtók elfogadhatnak (vagy várhatnak) olyan útmutatásokat is, amelyek itt nem szerepelnek, ezért mindegyik esetében érdemes áttekinteni a hozzájuk tartozó man oldalt. Bõvebb információért lásd a device.hints(5), kenv(1), loader.conf(5) és loader(8) man oldalakat.

12.6. Init: A folyamatirányítás elindítása

Miután a rendszermag sikeresen elindult, átadja a vezérlést az init(8) felhasználói folyamatnak, amely vagy az /sbin/init, vagy pedig a rendszerindítóban megadott `init_path` változó által mutatott program.

12.6.1. Az automatikus újraindulási folyamat

Az automatikus újraindulási folyamat gondoskodik róla, hogy az indulást követõen rendelkezésre álló állományrendszerek ne legyenek sérültek. Amennyiben mégis sérültek és a fsck(8) nem tudja megjavítani õket, az init(8) a rendszert egyfelhasználós módba állítja, ahol a rendszergazdának kell közvetlenül megoldania a fennálló problémákat.

12.6.2. Egyfelhasználós mód

Ezt a módot az automatikus újraindítási folyamat során érhetjük el, vagy akkor, ha a rendszert a `-s` kapcsolóval indítjuk, esetleg a rendszerindítóban beállítjuk a `boot_single` változót.

Ezt a módot többfelhasználós módban, a shutdown(8) hívásával is aktiválhatjuk, ha nem adjuk meg az újraindítást (`-r`) vagy leállítást (`-h`) kérõ opciók egyikét sem.

Ha az /etc/ttys állományban a `console` értékét `insecure` (nem biztonságos)ra állítjuk, a rendszer az egyfelhasználós módba lépés elõtt kérni fogja a `root` felhasználó jelszavát.

Példa 9. Nem biztonságos konzol megadása az /etc/ttys-ben

```
# name  getty                           type    status          comments
#
# If console is marked "insecure", then init will ask for the root password
# when going to single-user mode.
console none                            unknown off insecure
```

Az `insecure` (nem biztonságos) konzol az, ahol nem tekintjük megbízhatónak a rendszerkonzol fizikai biztonságát, és biztosak akarunk lenni benne, hogy csak az képes használni a rendszert egyfelhasználós módban, aki ismeri a `root` felhasználó jelszavát. Ez tehát nem arra utal, hogy magát a konzolt akarjuk nem biztonságos módban mûködtetni. Szóval, ha biztonságot akarunk, az `insecure`-t válasszuk, ne pedig a `secure`-t.

12.6.3. Többfelhasználós mód

Ha az init(8) mindent rendben talál, vagy ha a felhasználó kilépett az egyfelhasználós módból, a rendszer többfelhasználós módba lép át, ahol megkezdi az erõforrások konfigurálását.

Az erõforrások konfigurációja (rc)

Az erõforrásokat konfiguráló alrendszer beolvassa a folyamathoz kapcsolódó változók alapértelmezett értékeit az /etc/defaults/rc.conf állományból, majd módosítja õket a rendszer egyéni beállításai szerint, amit a /etc/rc.conf állományból olvas ki. Ezután elvégzi az /etc/fstab alapján az állományrendszerek csatlakoztatását, elindítja a hálózati szolgáltatásokat, egyéb rendszerdaemonokat, és végezetül lefuttatja a telepített csomagok indítószkriptjeit.

Az erõforrásokat konfiguráló alrendszerrõl magáról az rc(8) man oldalon, valamint az érintett szkriptek tanulmányozásával tudhatunk meg többet.

12.7. A leállítási folyamat

A shutdown(8) paranccsal vezérelt leállítás során az init(8) megpróbálja lefuttatni az /etc/rc.shutdown szkriptet, majd ezt követõen `TERM` (befejeztetés) jelzést küld az aktuálisan futó folyamatoknak, kis idõ múlva pedig `KILL` (leállítás) jelzést azoknak, amelyek még nem álltak le addig a pillanatig.

Azokon az architektúrákon és rendszereken, ahol elérhetõ a fejlett energiagazdálkodás támogatása, a FreeBSD-t a `shutdown -p now` paranccsal állíthatjuk le, amit közvetlenül a számítógép automatikus kikapcsolása követ. A FreeBSD-s rendszer újraindításához egyszerûen csak adjuk ki a `shutdown -r now` parancsot. Fontos tudni, hogy alapértelmezés szerint a shutdown(8) használatához `root` felhasználónak, vagy legalább az `operator` csoport tagjának kell lennünk. Ezekre a feladatokra egyébként a halt(8) és reboot(8) parancsok is használhatóak. Alkalmazásukról bõvebben a hozzájuk, valamint a shutdown(8)-hoz tartozó man oldalakon találhatunk bõvebben információkat.

Az energiagazdálkodás használatához a rendszermagnak beépítve vagy a megfelelõ modul betöltésével bizosítania kell az acpi(4) támogatást.

Chapter 13. Felhasználók és hozzáférések alapvető kezelése

13.1. Áttekintés

A FreeBSD lehetõvé teszi, hogy egyazon idõben egyszerre több felhasználó is dolgozhasson a számítógépen. Közülük nyilvánvalóan csak egy képes elõtte ülni , de rajta kívül még sok más felhasználó is be tud jelentkezni a munkájához hálózaton keresztül. A rendszer használatához minden egyes felhasználónak hozzáféréssel kell rendelkeznie.

A fejezet elolvasása során megismerjük:

- a FreeBSD rendszerben megtalálható különféle felhasználói hozzáférések közti különbségeket;
- hogyan készítsünk új felhasználói hozzáféréseket;
- hogyan töröljünk felhasználói hozzáféréseket;
- hogyan változtassuk meg a hozzáférés adatait, mint például a felhasználók teljes nevét vagy a választott parancsértelmezõjét;
- hogyan korlátozzuk az egyes hozzáféréseket vagy hozzáférések egy csoportját az olyan erõforrások, mint például a memória vagy a processzoridõ védelmében;
- hogyan használjuk csoportokat a hozzáférések karbantartásának megkönnyítésére.

A fejezet elolvasásához ajánlott:

- a UNIX® és a FreeBSD alapjainak ismerete (A UNIX alapjai).

13.2. Bevezetés

A rendszert bármilyen fajta módon csak hozzáféréseken keresztül tudjuk elérni, minden programot felhasználók futtatnak, ezért a felhasználók és hozzáférések kezelése a FreeBSD rendszerek szerves része.

A FreeBSD rendszerben minden hozzáférés rendelkezik bizonyos információkkal az azonosításhoz.

Felhasználó neve

A felhasználónevet a `login:` felirat megjelenésekor kell megadni. A felhasználók neveinek egyedinek kell lenni a számítógépen, tehát két felhasználó nem használhatja ugyanazt a nevet. A passwd(5) man oldalon megtalálhatjuk azokat a szabályokat, amelyek az érvényes felhasználónevek létrehozására vonatkoznak. Általánosságban elmondható, hogy a felhasználóneveknek kisbetûseknek kell lenniük és legfeljebb nyolc karakterbõl állhatnak.

Jelszó

Minden hozzáféréshez tartozik egy jelszó is. Ez a jelszó lehet akár üres is, ebben az esetben nincs szükség jelszóra a hozzáféréshez. Ez viszont többnyire nagyon rossz ötlet: minden hozzáférést erõsen ajánlott jelszóval védeni.

Felhasználó azonosítója (User ID, UID)

Az UID egy szám, amely hagyományosan 0-tól 65535-ig terjed , és a felhasználó rendszeren belüli egyedi azonosítására használatos. A FreeBSD az UID-ot a felhasználók azonosítására használja - bármelyik parancs, amely lehetõvé teszi felhasználónevek megadását, át fogja alakítani UID-dé, mielõtt ténylegesen dolgozni kezdene vele. Ez tehát azt jelenti, hogy több hozzáférésünk is lehet több különbözõ felhasználónévvel, de ugyanazzal az UID-del. Legalább is a FreeBSD ezeket egyetlen felhasználónak tekinti, de nem is valószínû, hogy ilyenre valaha szükségünk is lenne.

Csoportazonosító (Group ID, GID)

A csoportazonosító (Group ID, GID) egy szám, amely általában 0-tól 65535-ig terjed , és azt az elsõdleges csoportot azonosítja be egyedileg, amelyikhez a felhasználó tartozik. A csoportok segítségével az erõforrások hozzáférésének vezérlését tudjuk megoldani a felhasználók GID-jével az UID-dek helyett. Ezzel jelentõs mértékben csökkenthetõ egyes konfigurációs állományok mérete. Egy felhasználó egyszerre több csoport tagja is lehet.

Bejelentkezési osztály

A bejelentkezési osztályok a csoportszervezés kibõvítését célozzák meg, további rugalmasságot nyújtanak, amikor a rendszert az egyes felhasználók igényeihez szabjuk.

Jelszóváltási idõ

Alapértelmezés szerint a FreeBSD nem kényszeríti rá a felhasználókat, hogy rendszeresen megváltoztassák a jelszavukat. Ezt felhasználónként kikényszeríthetjük, és így az egyes, vagy akár az összes felhasználót kötelezhetjük az adott idõközönként jelszóváltásra.

A hozzáférés lejárati ideje

A FreeBSD-ben alapértelmezés szerint nem évülnek el a hozzáférések. Ha azonban olyan hozzáféréseket kell létrehoznunk, melyeknek korlátoznunk kell az élettartamukat, mint például egy iskolában a diákok számára, akkor ilyenkor meg tudjuk adni a lejáratuk idejét. Ezen dátum után a hozzáféréssel már nem lehet bejelentkezni a rendszerbe, viszont a hozzá tartozó könyvtárban tárolt állományok továbbra is megmaradnak.

Felhasználó teljes neve

Míg a felhasználónév tökéletesen azonosítja a FreeBSD számára a hozzáférést, nem feltétlenül tükrözi a felhasználó valódi nevét. Ezt az információt is meg lehet adni a hozzáféréshez.

Felhasználói könyvtár

A felhasználói könyvtár a rendszerben található azon könyvtár teljes elérési útvonala, ahová a felhasználó a bejelentkezést követõen kerül. Elterjedt megszokás, hogy az összes felhasználó könyvtárát a /home/felhasználónév vagy a /usr/home/felhasználónév könyvtárba teszik. A felhasználók ezekben a könyvtárakban tárolják a személyes állományaikat, és tetszõleges könyvtárakat hozhatnak létre benne.

Felhasználói parancsértelmezõ

A parancsértelmezõ biztosítja azt az alapértelmezett környezetet, amelyben a felhasználó kapcsolatba tud lépni a rendszerrel. Többféle parancsértelmezõ is akad, és a tapasztaltabb felhasználók ragaszkodnak is némelyikükhöz, ami gyakran látható is a hozzáférésük beállításaiban.

Három fõ típusa van a hozzáféréseknek: az adminisztrátori, a rendszer és a felhasználói hozzáférések. Az adminisztátori hozzáférés, amelyre gyakran `root`ként hivatkoznak, használatos a rendszer karbantartására, és semmilyen korlátozás nem érvényes rá. A rendszerhozzáférések szolgáltatásokat futtatnak. Végezetül a felhasználói hozzáféréseket használják a valódi emberek, akik bejelentkeznek, leveleket olvasnak és így tovább.

13.3. Az adminisztrátori hozzáférés

Az adminisztátori hozzáférés, amelyet általában csak `root`nak nevezünk, a rendszeradminisztrációs feladatok elvégzéséhez van igazítva, és nem ajánlott az olyan hétköznapi tevékenységek elvégzéséhez, mint például a levelek olvasása és írása, a rendszer bejárása vagy a programozás.

Ezért az adminisztrátor, eltérõen az átlagos felhasználói hozzáférésektõl, képes mindenféle határok nélkül tevékenykedni, és az adminisztrátori hozzáférés helytelen használata látványos katasztrófákat idézhet elõ. A felhasználói hozzáférések képtelenek merõ véletlenségbõl tönkretenni a rendszert, ezért általánosságban véve az a legjobb, ha egyszerû felhasználói hozzáféréseket használunk, amint módunk van rá, hacsak nincs szükségünk kifejezetten különleges jogosultságokra.

Minden esetben érdemes alaposan megfontolni az adminisztrátorként kiadott parancsokat, mivel egyetlen hiányzó szóköz vagy más egyéb karakter helyrehozhatatlan károkat okozhat a rendszerben.

Ezért, ha még nem tettük volna meg korábban, legyen az elsõ dolgunk a fejezet elolvasása után, hogy létrehozunk egy kiemelt jogosultságokkal nem rendelkezõ felhasználót saját magunk számára a hétköznapi feladatok lebonyolítására. Ez ugyanúgy vonatkozik a többfelhasználós és az egyfelhasználós módban futó rendszerekre is. A fejezet egy késõbbi részében leírjuk, hogyan lehet további hozzáféréseket létrehozni, és hogyan kell váltani egy mezei felhasználó és az adminisztrátor hozzáférése között.

13.4. Rendszerhozzáférések

A rendszer általi hozzáférések azok, amelyek olyan szolgáltatások futtatásáért felelõsek, mint például a DNS, a levelezés, a webszerverek és így tovább. Ennek oka a biztonság: ha minden szolgáltatást adminisztrátorként futtatnánk, bármit meg tudnának tenni a rendszerben.

Ilyen rendszerfelhasználók a `daemon`, `operator`, `bind` (a névfeloldáshoz), `news`, és a `www`.

A `nobody` ("senki") egy általános jogosultságok nélküli rendszerfelhasználó. Mindazonáltal nem szabad elfelejtenünk, hogy minél több szolgáltatást bízunk a `nobody`-ra, annál több állomány és program kerül vele kapcsolatba, ennélfogva annál erõsebbé válik a rendszer számára ez a felhasználó.

13.5. Felhasználói hozzáférések

A felhasználói hozzáférések a valós felhasználók elsõdleges eszközei a rendszer felé, és ezek a hozzáférések szigetelik el a felhasználókat és a környezeteket, megakadályozva, hogy a felhasználók kárt okozzanak akár a rendszerben, akár egymásnak, valamint lehetõvé teszik a

felhasználók számára a környezeteik testreszabását anélkül, hogy a többiekét módosítani kellene.

Minden olyan személynek, aki hozzá akar férni a rendszerünkhöz, rendelkeznie kell felhasználói azonosítóval. Ezáltal meg tudjuk állapítani, ki mivel foglalkozik éppen a rendszerben, és meg tudjuk akadályozni, hogy a felhasználók elérjék egymás beállításait, olvassák egymás leveleit és így tovább.

Minden felhasználó alakítani tudja a saját környezetét, és ezzel mintegy berendezkedik a rendszerünkben, különféle parancsértelmezõk, szövegszerkesztõk, billentyû-hozzárendelések és nyelvek használatával.

13.6. A hozzáférések módosítása

Egy UNIX®-os környezetben több különbözõ parancs közül választhatunk a felhasználói hozzáférések módosításakor. A legáltalánosabb parancsokat az alábbiakban foglaljuk össze, amit ezután a használatukat részletesebben bemutató példák követnek.

Parancs	Leírás
adduser(8)	az új felhasználók felvételére ajánlott parancssoros alkalmazás
rmuser(8)	a felhasználók eltávolítására ajánlott parancssoros alkalmazás
chpass(1)	rugalmas eszköz a felhasználói adatbázis információinak megváltoztatására
passwd(1)	egy egyszerû parancssoros segédprogram a felhasználói jelszavak megváltoztatásához
pw(8)	egy erõteljes és rugalmas segédeszköz a felhasználói hozzáférések teljeskörû módosításához

13.6.1. adduser

Az adduser(8) a felhasználók hozzáadására használható egyszerû program. Bejegyzéseket hoz létre a rendszer passwd és group állományaiban. Ezen kívül még létrehozza az új felhasználó könyvtárát is, odamásolja az alapértelmezett konfigurációs állományokat a /usr/shared/skel könyvtárból (ezek a felhasználóknál ponttal kezdõdõen jelennek meg, de az említett könyvtárban "dot" elõtaggal szerepelnek), és opcionálisan küld egy üdvözlõlevelet az újdonsült felhasználónak.

Példa 10. Felhasználó hozzáadása a FreeBSD-ben

```
# adduser
Username: jantyik
Full name: Jantyik Zsolt
Uid (Leave empty for default):
Login group [jantyik]:
Login group is jantyik. Invite jantyik into other groups? []: wheel
```

```
Login class [default]:
Shell (sh csh tcsh zsh nologin) [sh]: zsh
Home directory [/home/jantyik]:
Use password-based authentication? [yes]:
Use an empty password? (yes/no) [no]:
Use a random password? (yes/no) [no]:
Enter password:
Enter password again:
Lock out the account after creation? [no]:
Username   : jantyik
Password   : ****
Full Name  : Jantyik Zsolt
Uid        : 1001
Class      :
Groups     : jantyik wheel
Home       : /home/jantyik
Shell      : /usr/local/bin/zsh
Locked     : no
OK? (yes/no): yes
adduser: INFO: Successfully added (jantyik) to the user database.
Add another user? (yes/no): no
Goodbye!
#
```

A jelszó a beírás során egyáltalán nem jelenik meg, még csillagokat sem láthatunk a karakterek helyén. Ezért vigyázzunk, nehogy elgépeljük véletlenül a jelszót!

13.6.2. `rmuser`

Az rmuser(8) használható a felhasználók teljes eltávolítására a rendszerbõl. Az rmuser(8) az alábbi lépéseket hajtja végre:

1. Eltávolítja a felhasználó crontab(1) bejegyzéseit (amennyiben léteznek).
2. Eltávolítja az at(1) felhasználóhoz tartozó munkáit.
3. Leállítja a felhasználó által birtokolt összes futó programot.
4. Eltávolítja a felhasználót a rendszer helyi jelszó állományából.
5. Eltávolítja a felhasználó könyvtárát (amennyiben az a felhasználó birtokában van).
6. Eltávolítja a felhasználóhoz tartozó beérkezõ leveleket tartalmazó állományt a /var/mail könyvtárból.
7. Eltávolítja a felhasználó tulajdonában levõ összes állományt az olyan ideiglenes tárhelyekrõl, mint például a /tmp könyvtár.
8. Végezetül eltávolítja a felhasználó nevét az összes olyan csoportból, amelyhez az /etc/group szerint tartozik.

Ha menet közben egy csoport üressé válik, és a csoport neve megegyezik a felhasználó nevével, a csoport is eltávolításra kerül. Ez kiegészíti az adduser(8) eszközzel létrehozott felhasználónkénti egyedi csoportokat.

A rmuser(8) nem használható adminisztrátori hozzáférések törlésére, mivel az szinte majdnem mindig a teljes összeomlást vonja maga után.

Alapértelmezés szerint interaktív módban mûködik, melynek során megpróbál megbizonyosodni róla, hogy tényleg a megfelelõ dolgot cselekedjük.

Példa 11. `rmuser` Hozzáférések interaktív eltávolítása

```
# rmuser jantyik
Matching password entry:
jantyik:*:1001:1001::0:0:Jantyik Zsolt:/home/jantyik:/usr/local/bin/zsh
Is this the entry you wish to remove? y
Remove user's home directory (/home/jantyik)? y
Updating password file, updating databases, done.
Updating group file: trusted (removing group jantyik -- personal group is empty)
done.
Removing user's incoming mail file /var/mail/jantyik: done.
Removing files belonging to jantyik from /tmp: done.
Removing files belonging to jantyik from /var/tmp: done.
Removing files belonging to jantyik from /var/tmp/vi.recover: done.
#
```

13.6.3. chpass

A chpass(1) segítségével meg tudjuk változtatni a felhasználói adatbázisban található információkat, mint például a jelszavakat, parancsértelmezõket és a személyes adatokat.

Csak a rendszeradminisztrátoroknak, mint például magának az adminisztrátornak, szabad megváltoztatnia a felhasználók adatait a chpass(1) programmal.

Amikor az opcionálisan megadható felhasználói névtõl eltekintve nem adunk át neki paramétereket, a chpass(1) egy szövegszerkesztõben megnyitja az érintett felhasználó adatait. Miután kiléptünk belõle, a felhasználói adatbázist a megváltoztatott adatoknak megfelelõen frissíti.

Ha nem adminisztrátorként hívjuk meg, akkor a rendszer kérni fogja a jelszavunkat, miután kiléptünk a szövegszerkesztõbõl.

Példa 12. A `chpass` interaktív használata adminisztrátorként

```
# A jantyik nevű felhasználó adatainak módosítása.
Login: jantyik
Password: *
```

```
Uid [#]: 1001
Gid [# or name]: 1001
Change [month day year]:
Expire [month day year]:
Class:
Home directory: /home/jantyik
Shell: /usr/local/bin/zsh
Full Name: Jantyik Zsolt
Office Location:
Office Phone:
Home Phone:
Other information:
```

Egy átlagos felhasználó a bemutatott adatoknak csak igen kis részét képes módosítani, és azokat is csak saját maga számára.

Példa 13. A `chpass` interaktív használata normál felhasználóként

```
# A jantyik nevű felhasználó adatainak megváltoztatása.
Shell: /usr/local/bin/zsh
Full Name: Jantyik Zsolt
Office Location:
Office Phone:
Home Phone:
Other information:
```

A chfn(1) és chsh(1) parancsok csupán linkek a chpass(1) parancsra, akárcsak a ypchpass(1), ypchfn(1) és az ypchsh(1). A NIS támogatása teljesen magától mûködik, ezért az `yp` elõtag használata nem kötelezõ. Ha ez nem érthetõ, nem kell megijedni, a NIS-t majd a Hálózati szerverekben bemutatjuk.

13.6.4. passwd

Felhasználóként a saját jelszavunkat, adminisztrátorként pedig bármelyik felhasználó jelszavát a passwd(1) segítségével változtathatjuk meg a megszokott módon.

A véletlen balesetek és az illetéktelen változtatások ellen védelmet nyújt, hogy az eredeti jelszót is meg kell adnunk az új jelszó beállításához.

Példa 14. A jelszavunk megváltoztatása

```
% passwd
Changing local password for jantyik.
Old password:
New password:
```

```
Retype new password:
passwd: updating the database...
passwd: done
```

Példa 15. Egy másik felhasználó jelszavának megváltoztatása adminisztrátorként

```
# passwd jantyik
Changing local password for jantyik.
New password:
Retype new password:
passwd: updating the database...
passwd: done
```

Ahogy a chpass(1), az yppasswd(1) is csak egy link a passwd(1) parancsra, így a NIS mind a két megadási módban mûködik.

13.6.5. pw

A pw(8) egy olyan parancssori segédprogram, amellyel felhasználókat és csoportokat tudunk létrehozni, törölni, módosítani és megjeleníteni. Ez tulajdonképpen a rendszer felhasználókat és csoportokat tároló állományainak egyfajta kezelõfelülete. A pw(8) kiválóan paraméterezhetõ, aminek köszönhetõen remekül kiaknázható tudása a különféle parancsértelmezõk szkriptjeiben, habár a kezdõ felhasználók nehézkesebbnek érezhetik a kezelését a korábban mutatott parancsokhoz képest.

13.7. A felhasználók korlátozása

Ha már vannak felhasználóink, gyakran szóba kerülhet esetükben a rendszer használatának korlátozása. A FreeBSD rengeteg módon engedi korlátozni a rendszergazdának az egyénenként használható erõforrások mennyiségét a rendszerben. Ezek a korlátok két részre oszthatóak: a lemezkvótákra és egyéb erõforráskorlátokra.

A lemezkvóták a felhasználók lemezhasználatát korlátozzák, és lehetõvé teszik, hogy állandó újraszámolás nélkül, gyorsan ellenõrizni tudjuk ennek mértékét. A kvótákat a Az állományrendszerek kvótáiban részletezzük.

A többi erõforrás korlátozása magában foglalja a processzoridõ, memória és minden olyan erõforrás behatárolását, amihez a felhasználó csak hozzá tud férni. Ezeket bejelentkezési osztályokon keresztül határozzuk meg, ezekrõl esik itt most szó.

A bejelentkezési osztályokat az /etc/login.conf állományban adhatjuk meg. Ennek pontos ismertetése nem tárgya ennek a szakasznak, de ezt megtalálhatjuk a login.conf(5) man oldalon. Elegendõ csak annyit mondanunk, hogy minden felhasználóhoz tartozik egy bejelentkezési osztály (alapértelmezés szerint a `default` nevû), és minden egyes bejelentkezési osztályhoz tulajdonságok egy halmaza társul. Ezek a bejelentkezési tulajdonságok `név=érték` párosokból állnak, ahol *név* egy

jól ismert azonosító, illetve az *érték* egy tetszõleges sztring, melyet a nevétõl függõ módon dolgozunk fel. A bejelentkezési osztályok és tulajdonságok beállítása eléggé magától értetõdõ, és a login.conf(5) man oldal is jól leírja.

A rendszer általában nem magát az /etc/login.conf állományban található beállításokat olvassa be, hanem az /etc/login.conf.db állományt, amiben gyorsabban lehet keresni. Az /etc/login.conf állományból az /etc/login.conf.db állományt az alábbi paranccsal tudjuk legyártani:

```
# cap_mkdb /etc/login.conf
```

Az erõforrások korlátozása két irányban is eltér a sima hétköznapi bejelentkezési tulajdonságoktól. Elõször is minden korláthoz létezik egy gyenge (aktuális) és egy erõs korlát. A gyenge korlátok a felhasználók vagy az alkalmazások részérõl még finomíthatóak, de az erõs korláton túl már nem. Ez utóbbit mindig tudja csökkenteni a felhasználó, de sose tudja növelni. Másodsorban a legtöbb erõforráskorlát az adott felhasználó által futtatott programokra egyenként vonatkozik, nem pedig az összesre együttesen. Megjegyezzük azonban, hogy ezeket az eltéréseket a korlátok különleges kezelése indokolja, nem pedig a bejelentkezési tulajdonságok rendszerének megvalósítása (tehát a korlátok *valójában nem* ezen tulajdonságok speciális esetei.)

Így aztán, minden további magyarázkodás nélkül, felsoroljunk alant a leggyakrabban alkalmazott erõforráskorlátokat (a többi, más egyéb bejelentkezési tulajdonságokkal együtt, megtalálható a login.conf(5) man oldalon).

`coredumpsize`

A program által létrehozott memóriakivonat maximális méretét határolja be ez a korlát, értelemszerûen a többi lemezterületre vonatkozó korlátnak (például a `filesize` vagy a lemezkvóták) alárendelt módon. Mindazonáltal ezt gyakran használjuk egyfajta enyhébb lemezfoglalási korlátként. Mivel nem maguk a felhasználók hozzák létre ezeket az állományokat és sokszor nem is törlik le ezeket, ez a beállítás azonban megmentheti ezeket a nagyobb programok (mint például az emacs) összeomlása során keletkezõ memóriakivonatok felesleges helyfoglalásától.

`cputime`

Az a maximális processzoridõ, amit a felhasználó által futtatott programok egyenként fogyaszthatnak. A vétkezõ programok futását a rendszermag leállítja.

Ez a korlát a processzor*idõ*re vonatkozik, nem pedig a processzor kihasználtságának százalékára, ahogy a top(1) és a ps(1) szokta megjeleníteni. Ez utóbbi alapján korlátozni ugyanis, még ezen leírás készítésének pillanataiban nem lehetséges, és meglehetõsen hasztalan is lenne: egy fordítóprogram - ami minden bizonnyal egy szabályosan futó program - könnyen fel tudja emészteni majdnem az egész processzort egy idõre.

`filesize`

A felhasználó által birtokolható állományok maximális mérete. Eltérõen a lemezkvótáktól, ez a korlát az egyes állományokra vonatkozik, nem pedig a felhasználó összes állományára

együttesen.

`maxproc`

A felhasználó által egyidõben, az elõtérben és a háttérben futtatható programok maximális száma. Érthetõ okokból ez az érték nem lehet nagyobb, mint a rendszerben a sysctl(8) által definiált `kern.maxproc` (a rendszermag által maximálisan futtatható programok számának) értéke. Érdemes még továbbá megjegyezni, hogy ez a beállítás gátolhatja a felhasználó munkáját: gyakran hasznos lehet egyszerre több példányban is bejelentkezni a rendszerbe vagy csövekkel összekapcsolt programokat futtatni. Bizonyos feladatok, mint például egy nagyobb program lefordítása, több program futására is szétterjedhetnek (például a make(1), cc(1) és egyéb köztes feldolgozókra).

`memorylocked`

Ezzel korlátozhatjuk az egyes futó programok által zárolható memóriaterület méretét a központi memóriában (lásd mlock(2)). Egyes rendszerkritikus programok, mint például az amd(8), zárolják magukat a központi memóriában, és ezért soha nem lapozódnak ki onnan. Ennek köszönhetõen nem érinti ezeket a rendszer lapozásból eredõ esetleges lelassulása.

`memoryuse`

Ez az a maximális memóriamennyiség, amelyet egy futó program egyszerre használhat. Ebbe együttesen beleértendõ a központi memóriában és a lapozóállományban elfoglalt hely. Ez ugyan nem minden szempontból korlátozza egy program memóriahasználatát, de indulásnak megfelelõ.

`openfiles`

A felhasználó egyes futtatott programjai által egy idõben megnyitható állományok maximális száma. FreeBSD-ben az állományok közé a foglalatok és az IPC-csatornák is beszámítanak. Ezért vigyázzunk, nehogy véletlenül túlságosan alacsonyra állítsuk ezt az értéket. Ezt rendszerszinten a `kern.maxfiles` sysctl(8) érték határozza meg.

`sbsize`

A korlátozás a felhasználó által egyszerre maximálisan elérhetõ hálózati memória és így a rendszermag puffereire vonatkozik. Eredetileg a régebbi, sok csatlakozást felemésztõ DoS (Denial of Service) támadások ellen nyújtana védelmet, de általánosságban alkalmazható a hálózati kommunikáció korlátozására is.

`stacksize`

Ez a felhasználó által mûködtetett egyes programok vermeinek maximális mérete. Önmagában nem elegendõ a programok által használt memóriamennyiség korlátozására, így emiatt inkább a többi korláttal együtt érdemes alkalmazni.

Van néhány tényezõ, amelyekre érdemes odafigyelni az erõforrások korlátainak beállítása során. Most következik pár tipp, javaslat és egyéb megjegyzés a témához.

- A rendszerindítás során az /etc/rc által indított programok a `daemon` bejelentkezési osztályba tartoznak.
- Habár a rendszerrel érkezõ /etc/login.conf állományban remekül be van állítva a legtöbb korlát, de nekünk, mint rendszergazdáknak, kell ismernünk a saját rendszerünk korlátait. Ezen

korlátok túlzott tágításával a rendszerünk könnyen leterhelhetővé válik, míg a túlzott szűkítésével akadályozhatjuk a hatékony használatát.

- Az X Window System (X11) felhasználóinak a többi felhasználónál valószínűleg jóval több erőforráshoz kell tudniuk hozzáférni. Az X11 már önmagában sok erőforrást eszik, de egyben bátorítja is a felhasználókat több program párhuzamos futtatására.
- Ne felejtsük el, hogy sok korlát az egyes különállóan futó programokra vonatkozik, nem pedig a felhasználó összes futtatott programjára. Például ha beállítjuk 50-re az `openfiles` értékét, a felhasználó által elindított programok mindegyike legfeljebb 50 állományt tud majd megnyitni. Emiatt a felhasználó által egyszerre ténylegesen megnyitható állományok száma az `openfiles` és a `maxproc` aktuális értékeinek szorzatából adódik. Ugyanez igaz a memóriahasználatra is.

Az erőforrások korlátozásáról, a bejelentkezési osztályokról és tulajdonságaikról a hozzájuk tartozó man oldalakon olvashatunk: cap.mkdb(1), getrlimit(2) és login.conf(5).

13.8. Csoportok

Egy csoport nem több felhasználók összességénél. A csoportokat a nevük és az azonosítójuk (Group ID, GID) azonosítja be. A FreeBSD-ben (és a legtöbb UNIX®-szerű rendszerben) a rendszermag két tényező alapján dönt arról, mit szabad tennie egy futó programnak: ezek közül az egyik a tulajdonosának azonosítója (UID), a másik azon csoportok listája, melyeknek tagja a tulajdonos. Eltérően a UID-től, egy futó programhoz csoportok listája tartozik. Amikor egy felhasználó vagy egy futó program "csoportazonosítójára" hivatkoznak, általában csak a lista első elemére gondolnak.

A csoportok nevei és azonosítói közti megfeleltetéseket az /etc/group állományban találjuk. Ez lényegében egy szimpla szöveges állomány, négy kettősponttal elválasztott mezőt tartalmaz. Ezek közül az első a csoport neve, a második a titkosított jelszó, a harmadik a csoport azonosítója, a negyedik pedig a tagok vesszővel tagolt felsorolása. Akár kézzel is nyugodtan szerkeszthető (feltételezve persze, hogy nem vétünk benne szintaktikai hibát!). A szintaxis teljes leírását a group(5) man oldalon találhatjuk meg.

Ha nem akarjuk magunk szerkeszteni az /etc/group állományt, használhatjuk a pw(8) parancsot is csoportok létrehozására és törlésére. Például hozzuk létre a `pg_csoport` nevű csoportot és vizsgáljuk meg, valóban létrejött-e:

Példa 16. A csoportok tagjainak beállítása a pw(8) használatával

```
# pw groupadd pg_csoport
# pw groupshow pg_csoport
pg_csoport:*:1100:
```

A fent szereplő `1100`-as érték a `pg_csoport` csoportazonosítója. Ebben a pillanatban a `pg_csoport`nak még egyetlen tagja sincs, ami miatt lényegében haszontalan. Így hát hívjuk meg a `pg_csoport`ba a korábban létrehozott `jantyik` nevű felhasználót.

Példa 17. A csoport tagjainak beállítása a pw(8) használatával

```
# pw groupmod pg_csoport -M jantyik
# pw groupshow pg_csoport
pg_csoport:*:1100:jantyik
```

Az `-M` kapcsoló paramétere a csoportba sorolandó felhasználók neveinek vesszõkkel tagolt listája. A korábbi szakaszok alapján már tudjuk, hogy a jelszavakat tároló állomány egyben azokat a csoportokat is tartalmazza, ahova az egyes felhasználók tartoznak. Az utóbbiakat (a felhasználókat) automatikusan beleteszi a rendszer a csoportlistába, de az érintett felhasználó nem fog megjelenni tagként a pw(8) parancs `groupshow` utasításával, azonban az id(1) és a hozzá hasonló eszközökkel már látható lesz. Más szavakkal élve, a pw(8) csak az /etc/group állományt módosítja, és soha nem próbál meg további adatokat kiolvasni az /etc/passwd állományból.

Példa 18. Egy új tag felvétele a csoportba a pw(8) használatával

```
# pw groupmod pg_csoport -m kisati
# pw groupshow pg_csoport
pg_csoport:*:1100:jantyik,kisati
```

Az `-m` kapcsoló paramétere azon felhasználók vesszõvel tagolt listája, akiket fel akarunk venni a csoportba. Tehát eltérõen az elõzõ példától, ezeket a felhasználókat felvesszük a csoportba, nem pedig átírjuk velük a csoport jelenlegi tagjainak listáját.

Példa 19. Az id(1) használata a csoporttagság megállapítására

```
% id jantyik
uid=1001(jantyik) gid=1001(jantyik) groups=1001(jantyik), 1100(pg_csoport)
```

Ahogy láthatjuk is, a `jantyik` nevû felhasználó tagja a `jantyik` nevû csoportnak és a `pg_csoport`nak is.

A pw(8) mûködésérõl a saját man oldalán, az /etc/group formátumáról pedig a group(5) man oldalon találhatunk több információt.

Chapter 14. Biztonság

14.1. Áttekintés

Ez a fejezet egy alapvetõ bevezetés a rendszerek biztonsági fogalmaiba, ad néhány általános jótanácsot és a FreeBSD-vel kapcsolatban feldolgoz néhány komolyabb témát. Az itt megfogalmazott témák nagy része egyaránt ráhúzható rendszerünk és általánosságban véve az internet biztonságára is. A internet már nem az "békés" hely, ahol mindenki a kedves szomszéd szerepét játssza. A rendszerünk bebiztosítása elkerülhetetlen az adataink, szellemi tulajdonunk, idõnk és még sok minden más megvédésére az internetes banditák és hasonlók ellen.

A FreeBSD segédprogramok és mechanizmusok sorát kínálja fel a rendszerünk és hálózatunk sértetlenségének és biztonságának fenntartására.

A fejezet elolvasása során megismerjük:

- az alapvetõ rendszerbiztonsági fogalmakat, különös tekintettel a FreeBSD-re;
- milyen olyan különbözõ titkosítási mechanizmusok érthetõek el a FreeBSD-ben, mint például a DES és az MD5;
- hogyan állítsunk be egyszeri jelszavas azonosítást;
- hogyan burkoljunk az inetd segítségével TCP kapcsolatokat;
- hogyan állítsuk be a KerberosIV-t a FreeBSD 5.0-nál korábbi változatain;
- hogyan állítsuk be a Kerberos5-t a FreeBSD-n;
- hogyan állítsuk be az IPsec-et és hozzunk létre VPN-t FreeBSD/Windows® gépek között;
- hogyan állítsuk be és használjuk az OpenSSH-t, a FreeBSD SSH implementációját;
- mik azok az ACL-ek az állományrendszerben és miként kell ezeket használni;
- hogyan kell használni a Portaudit segédprogramot a Portgyûjteményböl telepített külsõ szoftvercsomagok biztonságosságának ellenõrzésére;
- hogyan hasznosítsuk a FreeBSD biztonsági tanácsait tartalmazó leírásokat
- mit jelent a futó programok nyilvántartása és hogyan engedélyezzük azt FreeBSD-n.

A fejezet elolvasásához ajánlott:

- az alapvetõ FreeBSD és internetes fogalmak ismerete.

A könyvben további biztonsági témákról is szó esik, például a Kötelező hozzáférés-vezérlés (MAC)ben a Kötelezõ hozzáférés-vezérlésrõl (MAC) és a Tűzfalakben pedig az internetes tûzfalakról.

14.2. Bevezetés

A biztonság egy olyan funkció, ami a rendszergazdától indul és nála is végzõdik. Míg az összes többfelhasználós BSD UNIX® rendszer önmagában is valamennyire biztonságos, a felhasználók "fegyelmezéséhez" szükség további biztonsági mechanizmusok kiépítésére és karbantartására, ami

minden bizonnyal egy rendszergazda egyik legfontosabb kötelessége. A számítógépek csak annyira biztonságosak, mint amennyire beállítjuk, és a biztonsági megfontolások állandó versenyben vannak az emberi kényelemmel. A UNIX® rendszerek általánosságban véve órási mennyiségû program párhuzamos futtatására képesek, melyek többsége kiszolgálóként fut - ez azt jelenti, hogy hozzájuk kívülrõl érkezõ egyedek csatlakozhatnak és társaloghatnak velük. Ahogy a tegnap kicsi és nagy számítógépei napjaink asztali gépeivé váltak és ahogy a számítógépek egyre többen csatlakoznak hálózatra és az internetre, a biztonság fontossága is egyre jobban növekszik.

A rendszerek biztonsága a támadások különbözõ formáival is foglalkozik, többek közt olyan támadásokkal, amelyek a rendszer összeomlását vagy használhatatlanságát célozzák meg, de nem próbálják meg veszélybe sodorni a `root` felhasználó hozzáférését ("feltörni a gépet"). A biztonsággal kapcsolatos problémák több kategóriára oszthatóak:

1. A szolgáltatások mûködésképtelenné tételére irányuló (DoS, Denial of Service) támadások.
2. A felhasználói fiókok veszélyeztetése.
3. Rendszergazdai jogok megszerzése a közeli szervereken keresztül.
4. Rendszergazdai jogok megszerzése a felhasználói fiókokon keresztül.
5. Kiskapuk létrehozása a rendszerben.

A szolgáltatások mûködésképtelenné tételére irányuló támadások olyan tevékenységre utalnak, amelyek képesek megfosztani egy számítógépet az erõforrásaitól. A DoS támadások többnyire nyers erõvel kivitelezett technikák, melyek vagy a rendszer összeomlasztását vagy pedig a használhatatlanná tételét veszik célba úgy, hogy túlterhelik az általa felkínált szolgáltatásokat vagy a hálózati alrendszert. Egyes DoS támadások a hálózati alrendszerben rejtõzõ hibákat igyekeznek kihasználni, amivel akár egyetlen csomaggal is képesek romba dönteni egy számítógépet. Ez utóbbit csak úgy lehet orvosolni, ha a hibát kijavítjuk a rendszermagban. A szerverekre mért csapásokat gyakran ki lehet védeni a paramétereik ügyes beállításával, melyek segítségével korlátozni tudjuk az ezeket ért terhelést egy kellemetlenebb helyezetben. A nyers erõt alkalmazó hálózati támadásokkal a legnehezebb szembenézni. Például az álcázott támadadások, melyeket szinte lehetetlen megállítani, remek eszközök arra, hogy elvágják gépünket az internettõl. Ezzel viszont nem csak azt iktatják ki, hanem az internet-csatlakozásunkat is eldugítják.

A DoS támadásoknál még gyakrabban elõfordul, hogy feltörik a felhasználók fiókjait. A rendszergazdák többsége még mindig futtat telnetd, rlogin, rshd és ftpd szervereket a gépen. Ezek a szerverek alapértelmezés szerint nem titkosított kapcsolaton keresztül mûködnek. Ebbõl következik, hogy ha nincs annyira sok felhasználónk és közülük néhányan távoli helyekrõl jelentkeznek be (ami az egyik leggyakoribb és legkényelmesebb módja ennek), akkor elõfordulhat, hogy valami megneszeli a jelszavaikat. A körültekintõ rendszergazdák mindig ellenõrzik a bejelentkezéseket tartalmazó naplókat és igyekeznek kiszûrni a gyanús címeket még abban az esetben is, amikor a bejelentkezés sikeres volt.

Mindig arra kell gondolni, hogy ha a támadónak sikerült megszerezni az egyik felhasználó hozzáférését, akkor akár képes lehet a `root` felhasználó fiókjának feltörésére is. Azonban a valóságban egy jól õrzött és karbantarott rendszer esetén a felhasználói hozzáférések megszerzése nem feltétlenül adja a támadó kezére a `root` hozzáférését. Ebben fontos különbséget tenni, hiszen a `root` felhasználó jogai nélkül a támadó nem képes elrejteni a nyomait és legjobb esetben sem tud többet tenni, mint tönkretenni az adott felhasználó állományait vagy összeomlasztani a rendszert.

A felhasználói fiókok feltörése nagyon gyakran megtörténik, mivel a felhasználók messze nem annyira elõvigyázatosak, mint egy rendszergazda.

A rendszergazdáknak mindig észben kell tartani, hogy egy számítógépen több módon is meg lehet szerezni a `root` felhasználó hozzáférését. A támadó megtudhatja a `root` jelszavát, hibát fedezhet fel az egyik rendszergazdai jogosultsággal futó szerverben és képes feltörni a `root` hozzáférést egy hálózati kapcsolaton keresztül, vagy a támadó olyan programban talál hibát, aminek segítségével el tudja érni a `root` fiókját egy felhasználói hozzáférésen keresztül. Miután a támadó megtalálta a rendszergazdai jogok megszerzésének módját, nem feltétlenül kell kiskapukat elhelyeznie a rendszerben. Az eddig talált és javított, rendszergazdai jogok megszerzését lehetõvé tevõ biztonsági rések egy része esetében viszont a támadónak akkora mennyiségû munkát jelentene eltüntetni maga után a nyomokat, hogy megéri neki egy kiskaput telepíteni. Ennek segítségével a támadó ismét könnyedén hozzájuthat a `root` felhasználó hozzáféréséhez a rendszerben, de ezen keresztül egy okos rendszergazda képes is a behatolót leleplezni. A kiskapuk lerakásának megakadályozása valójában káros a biztonság szempontjából nézve, mert ezzel nem szüntetjük meg azokat a lyukakat, amin keresztül a támadó elõször bejutott.

A támadások elleni védelmet mindig több vonalban kell megvalósítani, melyeket így oszthatunk fel:

1. A rendszergazda és a személyzet hozzáférésének védelme.
2. A rendszergazdai jogokkal futó szerverek és a suid/sgid engedélyekkel rendelkezõ programok védelme.
3. A felhasználói hozzáférések védelme.
4. A jelszavakat tároló állomány védelme.
5. A rendszermag belsejének, a nyers eszközök és az állományrendszerek védelme.
6. A rendszert ért szabálytalan módosítások gyors észlelése.
7. Állandó paranoia.

A fejezet most következõ szakaszában az imént felsorolt elemeket fejtjük ki részletesebben.

14.3. A FreeBSD védelme

Parancs kontra protokoll

A dokumentumban a félkövéren fogjuk szedni az alkalmazásokat, és `egyenszélességû` betûkkel pedig az adott parancsokra hivatkozunk. A protokollokat nem különböztetjük meg. Ez a tipográfiai elkülönítés hasznos például az ssh egyes vonatkozásainak esetén, mivel ez egyben egy protokoll és egy parancs is.

A most következõ szakaszok a FreeBSD védelmének azon módszereit ismertetik, amelyekrõl a fejezet elõzõ szakaszában már írtunk.

14.3.1. A rendszergazda és a személyzet hozzáférésének védelme

Elõször is: ne törjük magunkat a személyzeti fiókok biztonságossá tételével, ha még a rendszergazda hozzáférését sem tettük eléggé biztonságossá. A legtöbb rendszerben a `root` hozzáféréshez tartozik egy jelszó. Elsõként fel kell tennünk, hogy ez a jelszó *mindig* megszerezhetõ.

Ez természetesen nem arra utal, hogy el kellene távolítanunk. A jelszó szinte mindig szükséges a számítógép konzolon keresztüli eléréséhez. Valójában arra szeretnénk rávilágítani, hogy a konzolon kívül sehol máshol ne lehessen használni ezt a jelszót, még a su(1) paranccsal sem. Például gondoskodjunk róla, hogy az /etc/ttys állományban megadott pszeudó terminálokat "insecure" (nem biztonságos) típusúnak állítottuk be, és így a `telnet` vagy az `rlogin` parancsokon keresztül nem lehet rendszergazdaként bejelentkezni. Ha más szolgáltatáson keresztül jelentkezünk be, például az sshd segítségével, akkor ebben az esetben is gondoskodjunk róla, hogy letiltottuk a közvetlen rendszergazdai bejelentkezés lehetõségét. Ezt úgy tudjuk megtenni, ha megnyitjuk az /etc/ssh/sshd_config állományt és a `PermitRootLogin` paramétert átállítjuk a `no` értékre. Vegyünk számba minden lehetséges hozzáférési módot - az FTP és a hozzá hasonló módok gyakran átszivárognak a repedéseken. A rendszergazdának csak a rendszerkonzolon keresztül szabad tudnia bejelentkeznie.

Természetesen egy rendszergazdának valahogy el kell érnie a `root` hozzáférést, ezért ezzel felnyitunk néhány biztonsági rést. De gondoskodjunk róla, hogy ezek a rések további jelszavakat igényelnek a mûködésükhöz. A `root` hozzáférés eléréséhez érdemes felvenni tetszõleges személyzeti (staff) hozzáféréseket a `wheel` csoportba (az /etc/group állományban). Ha a személyzet tagjait a `wheel` csoportba rakjuk, akkor innen a `su` paranccsal fel tudjuk venni a `root` felhasználó jogait. A személyzet tagjait létrehozásukkor közvetlenül sose vegyük fel a `wheel` csoportba! A személyzet tagjai elõször kerüljenek egy `staff` csoportba, és majd csak ezután az /etc/group állományon keresztül a `wheel` csoportba. A személyzetnek csak azon tagjait tegyük ténylegesen a `wheel` csoportba, akiknek valóban szükségük van a `root` felhasználó hozzáférésére. Ha például a Kerberost használjuk hitelesítésre, akkor megcsinálhatjuk azt is, hogy a Kerberos .k5login állományában engedélyezzük a ksu(1) parancson keresztül a `root` hozzáférés elérését a `wheel` csoport alkalmazása nélkül. Ez a megoldás talán még jobb is, mivel a `wheel` használata esetén a behatolónak még mindig lehetõsége van hozzájutni a `root` hozzáféréséhez olyankor, amikor a kezében van a jelszavakat tároló állomány és meg tudja szerezni a személyzet valamelyik tagjának hozzáférését. A `wheel` csoport által felkínált megoldás ugyan jobb, mint a semmi, de kétségtelenül nem a legbiztonságosabb.

A hozzáférések teljes körû letiltásához a pw(8) parancsot érdemes használni:

```
# pw lock személyzet
```

Ezzel meg tudjuk akadályozni, hogy a felhasználó akármilyen módon, beleértve az ssh(1) használatát is, hozzá tudjon férni a rendszerünkhöz.

A hozzáférések blokkolásának másik ilyen módszere a titkosított jelszó átírása egyetlen “*” karakterre. Mivel ez a karakter egyetlen titkosított jelszóra sem illeszkedik, ezért a felhasználó nem lesz képes bejelentkezni. Ahogy például a személyzet alábbi tagja sem:

```
izemize:R9DT/Fa1/LV9U:1000:1000::0:0:Ize-Mize:/home/izemize:/usr/local/bin/tcsh
```

Erre cseréljük ki:

```
izemize:*:1000:1000::0:0:Ize-Mize:/home/izemize:/usr/local/bin/tcsh
```

Ezzel megakadályozzuk, hogy az `izemize` nevû felhasználó a hagyományos módszerekkel be tudjon jelentkezni. Ez a megoldás azonban a Kerberost alkalmazó rendszerek esetén nem mûködik, illetve olyan helyezetekben sem, amikor a felhasználó az ssh(1) paranccsal már létrehozott magának kulcsokat.

Az ilyen védelmi mechanizmusok esetében mindig egy szigorúbb biztonsági szintû géprõl jelentkezünk be egy kevésbé biztonságosabb gépre. Például, ha a szerverünk mindenféle szolgáltatásokat futtat, akkor a munkaállomásunknak egyetlen egyet sem lenne szabad. A munkaállomásunk biztonságossá tételéhez a lehetõ legkevesebb szolgáltatást szabad csak futtatnunk, de ha lehet, egyet sem, és mindig jelszóval védett képernyõvédõt használjuk. Természetesen ha a támadó képes fizikailag hozzáférni a munkaállomásunkhoz, akkor szinte bármilyen mélységû védelmet képes áttörni. Ezt mindenképpen számításba kell vennünk, azonban ne felejtsük el, hogy a legtöbb betörési kísérlet távolról, hálózaton keresztülrõl érkezik olyan emberektõl, akik fizikailag nem férnek hozzá a munkaállomásunkhoz vagy a szervereinkhez.

A Kerberos és a hozzá hasonló rendszerek használatával egyszerre tudjuk a személyzet tagjainak jelszavát letiltani vagy megváltoztatni, ami egyből érvényessé válik minden olyan gépen, ahová az adott felhasználónak bármilyen hozzáférése is volt. Nem szabad lebecsülnünk ezt a gyors jelszóváltási lehetõséget abban az esetben, ha a személyzet valamelyik tagjának hozzáférését megszerezték. Hagyományos jelszavak használatával a jelszavak megváltoztatása N gépen igazi káosz. A Kerberosban jelszóváltási megszorításokat is felállíthatunk: nem csak a Kerberos által adott jegyek járnak le idõvel, hanem a Kerberos rendszer meg is követelheti a felhasználóktól, hogy egy adott idõ (például egy hónap) után változtasson jelszót.

14.3.2. A rendszergazdai jogokkal futó szerverek és SUID/SGID engedélyekkel rendelkezõ programok védelme

A bölcs rendszergazda mindig csak akkor futtat szervereket, amikor szüksége van rá, se többet, se kevesebbet. Az egyéb fejlesztõktõl származó szerverekkel bánjunk különösen óvatosan, mivel gyakran hajlamosak hibákat tartalmazni. Például az imapd vagy a popper használata olyan, mintha az egész világnak ingyenjegyet osztogatnánk a rendszerünk `root` hozzáféréséhez. Soha ne futtassunk olyan szervert, amelyet nem vizsgáltunk át kellõ alapossággal. Sok szervert nem is feltétlenül kell `root` felhasználóként futtatni. Például az ntalk, comsat és finger démonok egy speciális *járókában* (sandbox) futnak. Ezek a járókák sem teljesen tökéletesek, hacsak erre külön figyelmet nem fordítunk. Ilyenkor a többvonalas védelem eszménye még mindig él: ha valakinek sikerült betörnie a járókába, akkor onnan ki is tud törni. Minél több védelmi vonalat húzunk a támadó elé, annál jobban csökken a sikerének valószínûsége. A történelem során lényegében minden `root` jogokkal futó szerverben, beleértve az alapvetõ rendszerszintû szervereket is, találtak már biztonsági jellegû hibát. Ha a gépünkre csak az sshd szolgáltatáson keresztül tudnak belépni, és soha nem használja senki a telnetd, rshd vagy rlogind szolgáltatásokat, akkor kapcsoljuk is ki ezeket!

A FreeBSD most már alapértelmezés szerint járókában futtatja az ntalkd, comsat és finger szolgáltatásokat. Másik ilyen program, amely szintén esélyes lehet erre, az a named(8). Az /etc/defaults/rc.conf megjegyzésben tartalmazza a named járókában futtatásához szükséges paramétereket. Attól függõen, hogy egy új rendszert telepítünk vagy frissítjük a már meglévõ rendszerünket, a járókákhoz tartozó speciális felhasználói hozzáférések nem feltétlenül jönnek létre. Amikor csak lehetséges, az elõrelátó rendszergazda kikísérletez és létrehoz ilyen járókákat.

Vannak más olyan szerverek, amelyek tipikusan nem járókákban futnak. Ilyen többek közt a sendmail, popper, imapd, ftpd és még sokan mások. Léteznek rájuk alternatívák, de a telepítésük valószínûleg több munkát igényel, mint amennyit megérné számunkra veszõdni velük (és itt megint lesújt a kényelmi tényezõ). Ezeket a szervereket többnyire `root` felhasználóként kell futtatnunk és a rajtuk keresztül érkezõ betörési kísérleteket más módokra támaszkodva kell észlelnünk.

A `root` felhasználó keltette biztonsági rések másik nagy csoportja azok a végrehajtható állományok a rendszerben, amelyek a suid és sgid engedélyekkel rendelkeznek, futtatásuk rendszergazdai jogokkal történik. Az ilyen binárisok többsége, mint például az rlogin, a /bin és /sbin, /usr/bin vagy /usr/sbin könyvtárakban található meg. Habár semmi sem biztonságos 100%-ig, a rendszerben alapértelmezetten suid és sgid engedéllyel rendelkezõ binárisok ebbõl a szempontból meglehetõsen megbízhatónak tekinhetõek. Alkalmanként azonban találnak a `root` felhasználót veszélyeztetõ lyukakat az ilyen binárisokban is. Például 1998-ban az `Xlib`-ben volt egy olyan rendszergazdai szintû hiba, amellyel az xterm (ez általában suid engedéllyel rendelkezik) sebezhetõvé vált. Mivel jobb félni, mint megijedni, ezért az elõretekintõ rendszergazda mindig igyekszik úgy csökkenteni az ilyen engedélyekkel rendelkezõ binárisok körét, hogy csak a személyzet tagjai legyenek képesek ezeket futtatni. Ezt egy olyan speciális csoport létrehozásával oldhatjuk meg, amelyhez csak a személyzet tagjai férhetnek hozzá. Az olyan suid binárisoktól pedig, amelyeket senki sem használ, igyekszik teljesen megszabadulni (`chmod 000`). A monitorral nem rendelkezõ szervereknek általában nincs szükségük az xterm mûködtetésére. Az sgid engedéllyel rendelkezõ binárisok is legalább ugyanennyire veszélyesek. Ha a behatoló képes feltörni egy `kmem` csoporthoz tartozó sgid binárist, akkor képes lesz olvasni a /dev/kmem állomány tartalmát, ezáltal hozzájut a titkosított jelszavakhoz és így megszerezheti magának akármelyik hozzáférést. Sõt, a `kmem` csoportot megszerzõ behatolók figyelni tudják a pszeudó terminálokon keresztül érkezõ billentyûleütéseket, még abban az esetben is, amikor a felhasználók egyébként biztonságos módszereket használnak. A `tty` csoportot bezsebelõ támadók szinte bármelyik felhasználó terminájára képesek írni. Ha a felhasználó valamilyen terminál programot vagy terminál emulátort használ a billentyûzet szimulációjával, akkor a behatoló tud olyan adatokat generálni, amivel a felhasználó nevében adhat ki parancsokat.

14.3.3. A felhasználói hozzáférések védelme

A felhasználók hozzáféréseit szinte a legnehezebb megvédeni. Míg a személyzet tagjaival szemben lehetünk kíméletlenül szigorúak és "ki is csillagozhatjuk" a jelszavukat, addig a felhasználók hozzáféréseivel általánosságban véve ezt nem tehetjük meg. Ha a kezünkben van a megfelelõ mértékû irányítás, akkor még gyõzhetünk és kényelmesen biztonságba helyezethetjük a felhasználók hozzáférését. Ha nincs, akkor nem tehetünk mást, mint állandóan õrködünk a hozzáférések felett. Az ssh és Kerberos használata a felhasználók esetén sokkalta problematikusabb, mivel ilyenkor jóval több adminisztrációra és mûszaki segítségnyújtásra van szükség, de még mindig jobb megoldás a titkosított jelszavakhoz képest.

14.3.4. A jelszavakat tároló állomány védelme

Az a legbiztosabb, ha minél több jelszót kicsillagozunk és a hozzáférések hitelesítésére ssh-t vagy Kerberost használunk. Igaz, a titkosított jelszavakat tároló állományt (/etc/spwd.db) csak a `root` képes olvasni, de a támadó meg tudja szerezni ezt a jogot még olyankor is, ha `root` felhasználóként nem feltétlenül tud írni.

A rendszerünkben futó biztonsági szkripteknek a jelszavakat tároló állomány változását folyamatosan tudnia kell figyelnie és jelentie (lásd lentebb a Az állományok sértetlenségének ellenõrzése címû fejezetet).

14.3.5. A rendszermag belsejének, a nyers eszközök és az állományrendszerek védelme

Ha a támadó megszerzi a `root` hozzáférését, akkor szinte bármit képes megtenni, de vannak bizonyos elõnyei. Például a mostanság fejlesztett legtöbb rendszermag tartalmaz valamilyen beépített csomaglehallgatót, amit FreeBSD alatt a bpf eszköz valósít meg. A támadók szinte mindig megpróbálnak valamilyen csomaglehallgatót használni a feltört gépen. A legtöbb rendszeren azonban nem kell feltétlenül megadnunk ezt az örömet, ezért nem is kell beépítenünk a rendszermagba a bpf eszközt.

De ha még ki is iktatjuk a bpf eszközt, még aggódhatunk a /dev/mem és /dev/kmem miatt. Egyébként ami azt illeti, a behatoló még így is képes írni a nyers eszközökre. Sõt, a rendszermagba képesek vagyunk modulokat is betölteni a kldload(8) használatával. A vállalkozó kedvû támadó a rendszermag moduljaként képes telepíteni és használni a saját bpf eszközét vagy bármilyen más, a csomagok lehallgatására alkalmas eszközt. Az ilyen problémák elkerülése érdekében a rendszermagot a legmagasabb védelmi szinten kell üzemeltetni, tehát legalább egyes szinten.

A rendszermag védelmi szintjét több különbözõ módon lehet állítani. A védelmi szintet úgy lehet a legegyszerûbben növelni, ha a `sysctl` paranccsal beállítjuk a `kern.securelevel` nevû, rendszerszintû változó értékét:

```
# sysctl kern.securelevel=1
```

A FreeBSD rendszermag alapértelmezés szerint a `-1` védelmi szinten indul. Ez egészen addig `-1` marad, amíg a rendszergazda vagy valamelyik init(8) során hívott rendszerindító szkript ezt meg nem változtatja. A rendszer indítása során úgy tudjuk beállítani a megfelelõ védelmi szintet, ha az /etc/rc.conf állományban megadjuk a `kern_securelevel_enable` változót a `YES` értékkel, illetve `kern_securelevel` értékeként a kívánt védelmi szintet.

A FreeBSD alapértelmezett védelmi szintje közvetlenül a rendszerindító szkriptek lefutása után `-1`. Ezt "nem biztonságos módnak" nevezik, mivel az állományok írásáért felelõs állományjelzõk nem feltétlenül mûködnek, mindegyik eszköz írható, olvasható és a többi.

Miután a védelmi szintet `1` vagy annál magasabb értékre állítottuk, akkor a rendszer figyelembe veszi a csak hozzáfûzést (append-only) és módosíthatatlanságot (immutable) megszorító állományjelzõket, nem engedélyezi a tiltásukat és az eszközök közvetlenül nem érhetõek el. A különbözõ védelmi szintek részletesebb bemutatását a security(7) man oldalon olvashatjuk (vagy a FreeBSD 7.0 elõtti változataiban a init(8) man oldalon).

Az `1` és az afeletti védelmi szinteken többek közt az X11 nem feltétlenül lesz futtatható (mivel a /dev/io eszköz elérése blokkolt), illetve a rendszer frissítése is akadályokba fog ütközni (a `installworld` futtatása során ideiglenesen ki kell kapcsolni az append-only és immutable állományjelzõket). Az X11 esetében ezt valahogy még ki lehet kerülni úgy, hogy ha az xdm(1) démont még a

> rendszerindítás elején aktiváljuk (amikor a védelmi szint még kellõen alacsony). Az összes védelmi szint és megszorítás esetén azonban nem mindig adható ilyen jellegû javaslat, ezért ilyenkor mindig érdemes elõre tervezni egy keveset. Emellett fontos alaposan megismerni a különbözõ védelmi megszorításokat, mivel jelentõs mértékben visszafoghatják a rendszer használhatóságát. Ez segít az adott helyzetben az egyszerûbb megoldást választani és ezáltal elkerülni a kellemetlen meglepetéseket.

Ha a rendszermag védelmi szintjét az 1 érték vagy afelé emeljük, akkor hasznos lehet a fontosabb (lényegében minden olyan programnak, amely a védelmi szint helyes beállítódása elõtt lefut) programoknak, könyvtáraknak és szkripteknek beállítani az `schg` állományjelzõt. Ilyenkor azonban vegyük figyelembe, hogy a rendszer frissítése is nehezebbé válik a magasabb védelmi szinteken. Egy mûködõképesebb megoldás lehet, ha rendszerünket egy magasabb védelmi szinten használjuk, de nem állítjuk be mindegyik rendszerszintû állományra az `schg` állományjelzõt. Másik lehetõség még a / és /usr partíciók írásvédett csatlakoztatása. Ne felejtsük el azonban, hogy ha túlságosan szigorúak vagyunk magunkhoz, akkor azzal egyúttal a behatolás észlelését is meg tudjuk nehezíteni!

14.3.6. Az állományok sértetlenségének ellenõrzése: binárisok, konfigurációs állományok stb.

Ha arról van szó, csak a legfontosabb rendszerszintû konfigurációs- és vezérlõállományokat tudjuk megvédeni, még mielõtt a korábban emlegetett kényelmi tényezõ kimutatná a foga fehérjét. Például, ha a `chflags` paranccsal beállítjuk az `schg` állományjelzõt a / és /usr állományrendszereken található legtöbb állományra, akkor az minden bizonnyal csökkenti a hatékonyságunkat, hiszen az állományok védelmének növekedésével csökken az észlelés lehetõsége. A védelmi vonalaink közül ugyanis az utolsó talán az egyik legfontosabb - a detektálás. A felépített biztonsági rendszerünk legnagyobb része szinte teljesen hasztalan (vagy ami még rosszabb, a biztonság hamis érzetét kelti), ha nem vagyunk képesek észrevenni a betörési kísérleteket. A védelmi rendszer egyik részére nem a támadó megállításához, hanem a lelassításához van szükség, hogy így majd munka közben érhessük tetten.

A betörés tényét legjobban a megváltozott, hiányzó vagy éppen váratlanul felbukkanó állományok utáni kutatással tudjuk felismerni. A módosított állományokat általában egy másik (gyakran központosított) korlátozott hozzáférésû rendszerbõl ellenõrizhetjük a legjobban. Fontos, hogy ha egy korlátozott hozzáférésû, kiemelten védett rendszeren írjuk a védelemért felelõs szkripteket, akkor azok szinte teljesen láthatlanok lesznek a támadó számára. A legjobb kihasználás érdekében a korlátozott hozzáférésû gépnek jelentõs mértékû rálátással kell rendelkeznie az összes többi gépre, amit írásvédett NFS exportok vagy ssh kulcspárok felhasználásával érhetünk el. A hálózati forgalmat leszámítva az NFS látszik a legkevésbé - segítségével lényegében észrevétlenül tudjuk figyelni az egyes gépek állományrendszereit. Ha a megfigyelésre használt szerver a kliensekhez switchen keresztül csatlakozik, akkor az NFS gyakran jobb választásnak bizonyul. Ha a szerver hubon vagy több hálózati elemen keresztül éri el a megfigyelni kívánt klienseket, akkor az NFS nem eléggé biztonságos (és hatékony), ezért ilyen esetekben az ssh választása lehet a kedvezõ még az ssh által hagyott nyomokkal együtt is.

Miután a korlátozott hozzáférésû gépünk legalább látja a hozzá tartozó kliensek rendszereit, el kell készítenünk a tényleges monitorozást végzõ szkripteket. Ha NFS csatlakozást tételezünk fel, akkor

az olyan egyszerû rendszereszközökkel, mint például a find(1) és md5(1) képesek vagyunk összerakni ezeket. A szemmel tartott kliensek állományait naponta legalább egyszer érdemes ellenõrizni md5-tel, valamint még ennél gyakrabban is tesztelni az /etc és /usr/local/etc könyvtárakban található konfigurációs és vezérlõállományokat. Ha valamilyen eltérést tapasztal az ellenõrzést végzõ szerverünk és a rajta levõ md5 információk is helyesek, akkor értesítenie kell a rendszergazdát. Egy jó védelmi szkript képes megkeresni az oda nem illõ suid binárisokat, valamint az új vagy törölt állományokat a / és a /usr partíciókon.

A védelmi szkriptek megírása valamivel nehezebb feladat, ha ssh-t használunk az NFS helyett. A futtatásukhoz a szkripteket és az általuk használt eszközöket (például find) az `scp` paranccsal lényegében át kell másolni a kliensekre, amivel így láthatóvá válnak. Ne feledjük továbbá, hogy az ssh kliens már eleve feltört lehet. Szó, ami szó, ha nem megbízható összeköttetésekrõl beszélünk, akkor az ssh használata elkerülhetetlen, de nem feltétlenül egyszerû.

Egy jó védelmi szkript észreveszi a felhasználók és a személyzet tagjainak hozzáférését vezérlõ állományokban, mint például az .rhosts, .shosts, .ssh/authorized_keys és társaiban keletkezett változásokat is, amelyek esetleg elkerülhetik egy `MD5` alapú ellenõrzés figyelmét.

Ha netalán órási mennyiségû tárterületettel rendelkeznénk, akkor eltarthat egy ideig, amíg végigsöprünk az összes partíció összes állományán. Ebben az esetben érdemes olyan beállításokat megadni az állományrendszerek csatlakoztatásánál, amivel le tudjuk tiltani a suid engedéllyel rendelkezõ binárisok futtatását. Ezzel kapcsolatban a mount(8) parancs `nosuid` opcióját nézzük meg. Hetente legalább egyszer azért mégis érdemes átnézni az ilyen partíciókat is, mivel ez a réteg a betörési kísérletek felderítésével foglalkozik, függetlenül a sikerességüktõl.

A futó programok nyilvántartása (lásd accton(8)) egy olyan viszonylag kevés költséggel járó lehetõség az operációs rendszerben, ami segítségünkre lehet a betörés utáni események kiértékelésében. Különösen hasznos olyankor, amikor megpróbáljuk modellezni, miképp is sikerült a támadónak bejutnia a rendszerünkbe, természetesen feltételezve, hogy az ehhez felhasznált feljegyzések a betörés után is érintetlenek maradtak.

Végül a védelmet ellátó szkripteknek javasolt feldolgozni a naplóállományokat is, valamint a naplókat magukat is a lehetõ legbiztonságosabb formában generálni - ilyenkor nagyon hasznos lehet, ha egy távoli gépre naplózunk. A behatoló megpróbálja majd eltüntetni a nyomait, a naplóállományok viszont nagyon fontosak a rendszergazda számára a betörési kísérletek idejének és módjának megállapításában. A naplókat úgy tudjuk tartósan rögzíteni, ha a rendszerkonzol üzeneteit soros porton keresztül gyûjtjük össze a konzolok felügyeletéért felelõs biztonságos gépen.

14.3.7. Állandó paranoia

Egy kis paranoia sosem árt. Elmondható, hogy a rendszergazda tetszõleges számú biztonsági intézkedéssel élhet egészen addig, amíg az nincs hatással a kényelmére, és a kényelmet *befolyásoló* biztonsági intézkedéseket pedig megfelelõ mérlegelés mellett tegye meg. Ami még ennél is fontosabb, hogy mindig változtassunk valamit a biztonsági hálónkon - mivel ha egy az egyben követjük a dokumentumban leírtakat, akkor ezzel együtt kiadjuk a bejutás receptjét annak a leendõ támadónknak, aki szintén elolvasta ugyanezt.

14.3.8. A szolgáltatások mûködésképtelenné tételét célzó támadások

Ez a szakasz a szolgáltatások mûködésképtelenségét elérni kívánó, más néven "Denial of Service" típusú támadásokkal foglalkozik. Noha nem tudunk túlságosan sokat tenni a manapság felbukkanó álcázott, a hálózatunk totális leterhelését célbavevõ támadások ellen, akadnak olyan általános érvényû eszközök, amelyekkel elejét vehetjük a szervereink szétbomzásának:

1. A létjövõ szerverpéldányok korlátozása.
2. Az ugródeszkaszerû támadások (támadás ICMP-válasszal, pingszórás stb.) korlátozása.
3. A rendszermag útválasztási gyorsítótárának túlterhelése.

A DoS támadások egyik jellemzõ sémája szerint egy sokszorozódni képes szervert támadnak meg, amelynek igyekeznek minél több példányát legyártatni, míg végül az ezt futtató rendszer ki nem fogy a memóriából, állományleíróból satöbbibõl és megállásra nem kényszerül. Az inetd (lásd inetd(8)) számos lehetõséget kínál fel ennek megakadályozására. Ezzel kapcsolatban szeretnénk megjegyezni, hogy bár ezzel el tudjuk kerülni a gépünk leállását, semmilyen garanciát nem ad arra, hogy a szolgáltatás a támadás során is zavartalanul üzemel tovább. Alaposan olvassuk el az inetd man oldalát és legyünk különös tekintettel a `-c`, `-C` és `-R` kapcsolóira. Vigyázzunk, hogy az inetd `-C` kapcsolóját képesek kijátszani az álcázott IP-vel érkezõ támadások, ezért inkább az elõbbi kapcsolók valamilyen kombinációja az ajánlott. Egyes szerverprogramoknál be lehet állítani a példányainak maximális számát.

A Sendmail rendelkezik egy `-OMaxDaemonChildren` beállítással, ami a terhelésben levõ késleltetése miatt néha mintha jobban beválna, mint a Sendmail terheléskorlátozó paraméterei. A Sendmail indításakor tehát a `MaxDaemonChildren` paramétert javasolt megadni egy olyan értékkel, amely elegendõ a Sendmail számára betervezett terhelés kiszolgálására, de még kevés ahhoz, hogy a Sendmail fûbe harapjon tõle. Továbbá bölcs dolog a Sendmailt várakozási sorral (`-ODeliveryMode=queued`) és démonként (`sendmail -bd`), külön feldolgozási menetekkel (`sendmail -q15m`) futtatni. Ha továbbra is valós idejû kézbesítést akarunk, akkor a feldolgozást kisebb idõközökkel is lefuttathatjuk (például `-q1m`), de arra *mindig ügyeljünk*, hogy a `MaxDaemonChildren` beállítása ne okozzon kaszkádosítási hibákat a Sendmail mûködésében.

A Syslogd közvetlenül is támadható, ezért határozottan javasoljuk a `-s` használatát, amikor csak lehet, minden más esetben pedig a `-a` beállítást.

Fordítsunk kellõ figyelmet a TCP kapcsolatok burkolását végzõ TCP Wrapper"reverse-ident" lehetõségére, ami szintén közvetlenül támadható. Ebbõl az okból kifolyólag valószínûleg nem is akarjuk a TCP Wrapper által felkínált reverse-ident-et használni.

Jól járunk el abban az esetben, ha a belsõ szolgáltatásainkat az útválasztóink mentén tûzfal segítségével védjük meg a külsõ hozzáféréstõl. Ezzel lényegében a helyi hálózatunkat kívülrõl fenyegetõ támadások ellen védekezünk, de ez nem nyújt elegendõ védelmet a belsõ szolgáltatásaink esetén a `root` hozzáférés megszerzésére irányuló kísérletek ellen. Mindig egy exkluzív, tehát zárt tûzfalat állítsunk be, vagyis "tûzfalazzunk mindent *kivéve* az A, B, C, D és M-Z portokat". Ezen a módon ki tudjuk szûrni az összes alacsonyabb portot, kivéve bizonyos eseteket, mint például a named (ha az adott zónában ez az elsõdleges gép), ntalkd, sendmail vagy más interneten keresztül elérhetõ szolgáltatásokat. Ha másképpen állítjuk a tûzfalat - inkluzív, nyílt avagy megengedõ módon, akkor jó eséllyel elfelejtünk "lezárni" egy csomó szolgáltatást, vagy úgy adunk hozzá egy új

belsõ szolgáltatást, hogy közben elfelejtjük frissíteni a tûzfalat. Ennél még azon is jobb, ha a tûzfalon nyitunk egy magasabb portszámú tartományt, és ott valósítjuk meg ezt a megengedõ jellegû mûködést, az alacsonyabb portok veszélybe sodrása nélkül. Vegyük azt is számításba, hogy a FreeBSD-ben a kiosztott portokat dinamikusan állíthatjuk a `net.inet.ip.portrange` sysctl változókon keresztül (`sysctl -a | fgrep portrange`), ami nagyságrendekkel megkönnyíti a tûzfal beállítását. Ennek megfelelõen például meg tudjuk adni, hogy a 4000-tõl 5000-ig terjedõ porttartomány a 49152-tõl 65535-ig húzódó tartományba kerüljön át, majd a 4000 alatti összes portot blokkoljuk (természetesen az internetrõl szándékosan hozzáférhetõ portok kivételével).

A DoS támadások másik elterjedt fajtája az ún. "ugródeszka támadás" - ilyenkor a szervert úgy próbálják túlterhelni, hogy folyamatosan válaszokat kérnek tõle a helyi hálózatról vagy egy másik számítógéprõl. Az ilyen természetû támadások közül is a legnépszerûbb az *ICMP pingszórásos támadás*. A támadó olyan ping csomagokat küld szét a helyi hálózaton, amelyek forrásának azt a gépet jelöli meg, amelyiket meg akarja támadni. Ha a hálózatokat elválasztó útválasztók nem fogják meg a pingszórást, akkor a helyi hálózatról összes gépe nekilát válaszolgatni a meghamisított forrás címére, amivel így teljesen leterhelik az áldozatot. Ez különösen akkor hatásos, amikor a támadó ugyanezt a trükköt eljátssza egyszerre több tucat különbözõ hálózatban is. Az üzenetszórással járó támadások akár százhúsz megabitnyi forgalmat is képesek generálni másodpercenként. A második legelterjedtebb ugródeszkás támadás az ICMP hiba-visszajelzési rendszere ellen irányul. Ilyenkor a támadó ICMP hibaüzeneteket kiváltó csomagok készítésével képes eltömíteni egy szerver bejövõ hálózati kapcsolatát és az ICMP válaszokkal pedig a szerver maga dugítja el a kimenõ hálózati kapcsolatát. Ez a fajtájú támadás képes kinyomni az összes memóriát a szerverbõl és ezzel összeomlasztani, különösen olyankor, amikor a szerver nem tudja elég gyorsan elnyelni az általa generált ICMP válaszokat. A `net.inet.icmp.icmplim` sysctl változóval tudunk gátat szabni a támadások ezen fajtájának. Az ugródeszkás támadások utolsó nagyobb osztálya az inetd olyan szolgáltatásait szemeli ki, mint például az udp echo. A támadó ilyenkor egyszerûen küld a helyi hálózatunkon található A és B szerverünknek egy olyan UDP csomagot, ahol forrásként az A szerver echo portját adja meg, célnak pedig a B szerver echo portját. Ezután a két szerver elkezdi egymás között passzolgatni ezt az egyetlen csomagot. A támadó még több ilyen csomag befecskendezésével pillanatok alatt képes leterhelni a két szervert és helyi hálózatot. Hasonló problémák vannak a belsõ chargen portjával is. Egy hozzáértõ rendszergazda ezért kikapcsolja az összes ilyen inetd-alapú belsõ tesztelõ szolgáltatást.

Az álcázott csomagok felhasználhatóak a rendszermag útválasztó gyorsítótárának túlterhelésére is. Ezzel kapcsolatban nézzük meg a `net.inet.ip.rtexpire`, `rtminexpire` és `rtmaxcache` sysctl változókat. A véletlenszerû IP-címekkel megcímzett álcázott csomagok hatására a rendszermag létrehoz mindegyikõjükhöz egy ideiglenesen pufferelt utat az útválasztó táblázatában, amelyet a `netstat -rna | fgrep W3` paranccsal tudunk lekérdezni. Az ilyen útvonalak nagyjából 1600 másodperc múlva elévülnek. Ha a rendszermag észleli, hogy a gyorsítótárazott útválasztási táblázat mérete túlságosan megnövekedett, akkor automatikusan csökkenti az `rtexpire` értékét, de soha nem megy a `rtminexpire` alá. Ebbõl két probléma adódik:

1. A rendszermag nem reagál elég gyorsan amikor egy alig terhelt szervert hirtelen megtámadnak.
2. Az `rtminexpire` nem elég kicsi ahhoz, hogy a rendszermag túléljen egy tartósabb rohamot.

Ha a szervereink az internethez T3 (kb. 45 Mbit/s) vagy gyorsabb összeköttetésen keresztül csatlakoznak, akkor határozottan javasolt kézileg behangolni a sysctl(8) segítségével az `rtexpire` és az `rtminexpire` értékeket. Soha ne állítsuk egyiket sem nullára (hacsak nem akarjuk összeomlasztani

a gépünket). Ha például mind a kettõt 2 másodpercre állítjuk, akkor az többnyire elegendõ az útválasztási táblázat megvédéséhez.

14.3.9. Hozzáférés Kerberosszal és SSH-val

Van néhány dolog, amit a Kerberos és az ssh esetén ajánlatos tisztázni, mielõtt használjuk ezeket. A Kerberos 5 egy kifogástalan hitelesítési protokoll. A telnet és rlogin Kerberos által módosított változatában vannak olyan hibák, amelyek alkalmatlanná teszik ezeket a bináris adatfolyamok helyes kezelésére. Sõt, alapértelmezés szerint a Kerberos nem titkosítja a kapcsolatot, csak ha megadjuk neki a `-x` kapcsolót. Az ssh alapértelmezés szerint mindent titkosít.

Az ssh minden szempontból nagyon jól teljesít kivéve, hogy alapértelmezés szerint átküldi a kulcsokat is. Ez azt jelenti, hogy ha van egy olyan biztonságos munkaállomásunk, ahol a rendszer többi részéhez tartozó kulcsainkat tartjuk és egy nem biztonságos gépre akarunk vele ssh-n keresztül belépni, akkor a kulcsaink használatóvá válnak. A tényleges kulcsokat ugyan nem látja senki, de a bejelentkezés során az ssh megnyit egy közvetítéshez használt portot, amit a nem biztonságos gépen a támadó egy feltört `root` hozzáférés birtokában ki tud használni úgy, hogy a kulcsaink segítségével hozzá tudjon férni egy másik olyan géphez, amelyet a kulcsok nyitnak.

Ha lehetséges, akkor a személyzet bejelentkeztetéséhez az ssh-t és Kerberost együttesen használjuk. Az ssh lefordítható Kerberos támogatással. Ezzel csökkentjük a potenciálisan kiszivárgó ssh kulcsok esélyét, miközben jelszavainkat a Kerberosszal védjük. Az ssh kulcsokat csak biztonságos gépekrõl és csak automatizált feladatok esetén használjuk (amire a Kerberos lényegében nem alkalmas). Emellett javasoljuk azt is, hogy az ssh beállításai között tiltsuk le a kulcsok átküldését (key forwarding) vagy használjuk az `from=IP/DOMAIN` opciót, amivel az ssh csak a megadott gépekrõl engedi az authorized_keys állomány és a így benne levõ kulcsok használatát.

14.4. DES, Blowfish, MD5 és a Crypt

Minden UNIX® rendszer használójához tartozik egy jelszó is a hozzáféréséhez. Teljesen nyilvánvalónak tûnik, hogy ezt a jelszót csak az adott felhasználó és az adott operációs rendszer ismeri. A jelszavakat a titokban tartásukhoz ún. "csapóajtó függvényekkel" titkosítják, amelyeket könnyû titkosítani, ám nehéz visszafejteni. Tehát amit egy perccel ezelõtt még nyilvalónak tituláltunk, az mostanra már nem is teljesen igaz: *valójában* az operációs rendszer sem ismeri a jelszót. Az operációs rendszer csak a jelszó *titkosított* változatát ismeri. A jelszó "titkosítatlan" formáját csak nyers erõ igényebevételével tudjuk megkeresni az összes lehetséges jelszó szénakazlában.

Sajnos, annak idején, amikor a jelszavak titkosítása bekerült a UNIX®-ba, egyedül a DES, vagy más néven a Data Encryption Standard (Adattitkosítási szabvány) jött szóba. Ez alapvetõen nem jelentett problémát az Egyesült Államok állampolgárai számára, de mivel a DES forráskódját nem lehetett kivinni az Egyesült Államokból, a FreeBSD-nek találnia kellett valami olyasmit, ami mind megfelel az Egyesült Államok törvényeinek, mind pedig kompatibilis marad az összes többi DES-t használó UNIX® variánssal.

Ezt úgy oldották meg, hogy felosztották a titkosítással foglalkozó függvénykönyvtárakat, így az Egyesült Államokban élõ felhasználók tudtak DES könyvtárakat telepíteni és használni, miközben a többi nemzet felhasználói olyan más titkosítási módszert tudtak választani, amit kinn is lehetett

alkalmazni. Ennek tulajdonítható, hogy a FreeBSD alapértelmezés szerint az MD5 segítségével titkosít. Az MD5-öt a DES-nél sokkalta biztonságosabbnak tartják, ezért a DES telepítésének lehetõségét leginkább csak kompatibilitási okokból ajánlották fel.

14.4.1. A titkosítási mechanizmus azonosítása

Jelenleg a könyvtár ismeri a DES, MD5 és Blowfish függvényeit. A FreeBSD a jelszavak titkosításához alapból az MD5-öt használja.

Nagyon könnyen meg tudjuk mondani, hogy a FreeBSD éppen melyik titkosítási módszert alkalmazza. Ennek egyik lehetõsége, ha az /etc/master.passwd állományt vizsgáljuk meg. Az MD5 függvényével titkosított jelszavak hosszabbak, mint a DES függvényével titkosítottak és a `$1$` karakterekkel kezdõdnek. A `$2a$` karakterekkel kezdõdõ jelszavakat Blowfish-sel titkosították. A DES kódolású jelszavaknak nincs semmilyen különleges ismertetõjelük, de általánosságban elmondható róluk, hogy rövidebbek az MD5 jelszavaknál és olyan 64 karakteres ábécével kódolják ezeket, amelyek nem tartalmazzák a `$` karaktert, így tehát a viszonylag rövid, nem dollárjellel kezdõdõ karakterláncok minden bizonnyal DES kódolású jelszavak.

Az új jelszavak kódolásához használt formátumot az /etc/login.conf állományban tárolt `passwd_format` bejelentkezési tulajdonság adja meg, amelynek értékei `des`, `md5` vagy `blf` lehetnek. A login.conf(5) man oldalon tájékozódhatunk bõvebben a bejelentkezési tulajdonságokról.

14.5. Egyszeri jelszavak

A FreeBSD alapértelmezés szerint támogatja az OPIE-t (One-time Passwords In Everything, azaz "Egyszeri jelszavak mindenben"), ami alapból az MD5 függvényét használja.

A jelszavak három fajtáját fogjuk a továbbiakban tárgyalni. Az elsõ a megszokott UNIX® stílusú avagy Kerberos jelszó. Ezt a továbbiakban "UNIX® jelszónak" nevezzük. A második fajtában az OPIE opiekey(1) nevû segédprogramja által generált és a bejelentkezésnél a opiepasswd(1) által elfogadott jelszavak tartoznak. Ezeket "egyszeri jelszavaknak" fogjuk nevezni. A jelszavak utolsó típusa az a titkos jelszó, amit az `opiekey` programnak (és néha a `opiepasswd` programnak) adunk meg, ami ebbõl egyszer használatos jelszavakat állít elõ. Ezt innentõl "titkos jelszónak" vagy csak egyszerûen "jelszónak" hívjuk.

A titkos jelszónak semmi köze sincs a UNIX® jelszavunkhoz. Természetesen megegyezhetnek, de ezt nem ajánljuk. Az OPIE által használt titkos jelszavaknak nem kell a régi UNIX® jelszavakhoz hasonlóan legfeljebb 8 karakteresnek lenniük , bármekkorát használhatunk. A hat vagy hét szóból álló jelszavak ilyenkor igen gyakoriak. Az OPIE jobbára a UNIX® jelszórendszerétõl teljesen függetlenül mûködik.

A jelszavak mellett két másik fajta adat fontos az OPIE számára. Közülük az egyiket "magnak" vagy "kulcsnak" nevezik, ami két betûbõl és öt számjegybõl áll. A másik az "iterációk száma", ami egy 1 és 100 közötti számot takar. Az OPIE úgy hozza létre az egyszeri jelszavakat, hogy egymás után fûzi a magot és a titkos jelszót, majd az iterációk megadott számának megfelelõ mennyiségben kiszámolja rá az MD5 függvény értékét és az eredményt hat rövid angol szóba önti. Ez a hat angol szó lesz a mi egyszeri jelszavunk. A hitelesítéssel foglalkozó rendszer (elsõsorban a PAM) figyelemmel kíséri a legutoljára használt egyszeri jelszavunkat, és csak akkor engedi a felhasználót hitelesíteni, ha az általa megadott jelszó kódolt változata megegyezik az elõzõleg megadott jelszaváéval. A csapóajtó

függvények használata miatt lehetetlen legenerálni a következõ egyszeri jelszót, ha a sikerült megszereznünk az egyiket. Az iterációk száma minden egyes sikeres bejelentkezés után csökken eggyel, amivel a felhasználót és a bejelentkeztetõ programot szinkronban tartja. Amikor így az iterációk száma eléri az egyet, az OPIE-t újra kell inicializálni.

Az említésre kerülõ rendszerek mindegyikéhez tartozik néhány program. Az `opiekey` bekéri az iterációk számát, a magot és a titkos jelszót, majd elõállít egy egyszer használatos jelszót vagy azok folytonos listáját. Az `opiepasswd` az OPIE inicializálásért, a jelszavak, az iterációk számának és a mag megváltoztatásáért felelõs. Egyaránt elfogad titkos jelmondatot, iterációs számot vagy magot és egy egyszeri jelszót. Az `opieinfo` megvizsgálja a felhasználókra vonatkozó adatbázist (/etc/opiekeys) és kiírja az adott felhasználó által használt iterációs számot és magot.

Négyféle különbözõ mûveletrõl fogunk most itt beszélni. Az elsõben egy biztonságos kapcsolaton keresztül elsõként inicializáljuk az egyszeri jelszavakat, vagy megváltoztatjuk a jelszót vagy a magot az `opiepasswd` segítségével. A második mûveletben ugyanarra adjuk ki az `opiepasswd` parancsot egy nem biztonságos kapcsolaton keresztül az `opiekey` paranccsal együtt egy biztonságos kapcsolaton keresztül. A harmadikban az `opiekey` használatával nem biztonságos kapcsolaton keresztül jelentkezünk be. A negyedikben az `opiekey` paranccsal létrehozunk egy adott mennyiségû kulcsot, amelyeket aztán leírhatunk vagy kinyomtathatunk, hogy magunkkal tudjuk vinni olyan helyre, ahonnan nem tudnk biztonságos módon csatlakozni.

14.5.1. Inicializálás biztonságos kapcsolattal

Az OPIE elsõ inicializálásához adjuk ki az `opiepasswd` parancsot:

```
% opiepasswd -c
[grimreaper] ~ $ opiepasswd -f -c
Adding unfurl:
Only use this method from the console; NEVER from remote. If you are using
telnet, xterm, or a dial-in, type ^C now or exit with no password.
Then run opiepasswd without the -c parameter.
Using MD5 to compute responses.
Enter new secret pass phrase:
Again new secret pass phrase:
ID unfurl OTP key is 499 to4268
MOS MALL GOAT ARM AVID COED
```

A figyelmeztetés fordítása:

```
Ezt a módszert csak konzolról alkalmazzuk, SOHA ne távoli kapcsolaton
keresztül!  Ha telnetet, xtermet vagy betárcsázós kapcsolatot használunk, akkor
azonnal nyomjunk ^C-t vagy ne adjunk meg jelszót.
```

Az `Enter new secret pass phrase:` vagy `Enter secret password:` kérdések után adjunk meg egy jelmondatot, illetve jelszót. Ne felejtsük el, hogy ez nem bejelentkezéshez használt jelszó lesz, hanem ebbõl jönnek majd létre az egyszeri kulcsaink. Az "ID" sor adja meg az aktuális példányunk paramétereit: a bejelentkezéshez használt nevünket, az iterációk számát és a magot. Amikor a

bejelentkezések során a rendszer emlékszik a paraméterekre és megjeleníti ezeket, nem kell megjegyeznünk. Az utolsó sor adja meg a paramétereinknek és a titkos jelszavunknak megfelelõ egyszeri jelszót. Ha most azonnal akarnánk bejelentkezni, akkor ezt az egyszeri jelszót kellene hozzá használnunk.

14.5.2. Inicializálás nem biztonságos kapcsolattal

Ha egy nem biztonságos kapcsolaton keresztül akarjuk inicializálni vagy megváltoztatni a jelszavunkat, akkor szükségünk lesz valahol egy megbízható kapcsolatra, ahol le tudjuk futtatni az `opiekey` parancsot. Ez lehet egy számunkra biztonsági szempontból elfogadható gép parancssora. Emellett ki kell találnunk egy iterációs számot (erre a 100 egy jó választás) és adnunk egy magot vagy használni egy véletlenszerûen generáltat. Az inicializálás színtere felé vezetõ nem biztonságos kapcsolaton keresztül adjuk ki az `opiepasswd` parancsot:

```
% opiepasswd

Updating unfurl:
You need the response from an OTP generator.
Old secret pass phrase:
        otp-md5 498 to4268 ext
        Response: GAME GAG WELT OUT DOWN CHAT
New secret pass phrase:
        otp-md5 499 to4269
        Response: LINE PAP MILK NELL BUOY TROY

ID mark OTP key is 499 gr4269
LINE PAP MILK NELL BUOY TROY
```

Az alapértelmezett mag elfogadásához nyomjuk le a `Return` billentyût. Mielõtt megadnánk a hozzáférés jelszavát, menjünk át a biztonságos kapcsolatra és adjuk meg neki ugyanezeket a paramétereket:

```
% opiekey 498 to4268
Using the MD5 algorithm to compute response.
Reminder: Don't use opiekey from telnet or dial-in sessions.
Enter secret pass phrase:
GAME GAG WELT OUT DOWN CHAT
```

Most váltsunk vissza a nem biztonságos kapcsolatra és másoljuk be az így generált egyszeri jelszót a megfelelõ programba.

14.5.3. Egyetlen egyszeri jelszó létrehozása

Miután sikeresen inicializáltuk az OPIE-t és bejelentkezünk, a következõket láthatjuk:

```
% telnet example.com
Trying 10.0.0.1...
```

```
Connected to example.com
Escape character is '^]'.

FreeBSD/i386 (example.com) (ttypa)

login: felhasználói_név
otp-md5 498 gr4269 ext
Password:
```

Mellékesen megjegyezzük, hogy az OPIE paranccsorának van egy (itt nem látható) hasznos képessége: ha `Return` billentyût nyomunk a jelszó bekérésekor, akkor a program megmutatja a begépelt betûket, így láthatjuk pontosan mit is írunk be. Ez nagyon kényelmes lehet olyankor, amikor valahonnan, például egy lapról olvassuk a jelszót.

A bejelentkezéshez ekkor le kell valahogy generálnunk az egyszeri jelszavunkat. Ezt egy megbízható rendszeresen tudjuk megtenni az `opiekey` lefuttatásával. (Ennek vannak DOS-os, Windows®-os és Mac OS®-es változatai is.) Paraméterként az iterációs számot és a magot kell megadnunk. Ezt akár közvetlenül át is másolhatjuk annak a gépnek a bejelentkezési képernyõjérõl, ahova be akarunk jelentkezni.

A megbízható rendszeren tehát:

```
% opiekey 498 to4268
Using the MD5 algorithm to compute response.
Reminder: Don't use opiekey from telnet or dial-in sessions.
Enter secret pass phrase:
GAME GAG WELT OUT DOWN CHAT
```

Most már megvan a bejelentkezéshez szükséges egyszeri jelszavunk.

14.5.4. Több egyszeri jelszó létrehozása

Néha olyan helyekre kell mennünk, ahol se egy megbízható gép, sem pedig biztonságos kapcsolat nem található. Ilyen esetekben megadhatjuk az `opiekey` parancsnak, hogy elõre gyártson le több egyszer használatos jelszót, amit késõbb aztán ki tudunk nyomtatni. Például:

```
% opiekey -n 5 30 zz99999
Using the MD5 algorithm to compute response.
Reminder: Don't use opiekey from telnet or dial-in sessions.
Enter secret pass phrase: <secret password>
26: JOAN BORE FOSS DES NAY QUIT
27: LATE BIAS SLAY FOLK MUCH TRIG
28: SALT TIN ANTI LOON NEAL USE
29: RIO ODIN GO BYE FURY TIC
30: GREW JIVE SAN GIRD BOIL PHI
```

Az `-n 5` öt kulcsot kér egymás után, a `30` pedig megadja az utolsó iterációs számot. Vegyük észre,

hogy a kulcsokat a felhasználás sorrendjével *ellentétes* sorrendben írja ki a program. Ha igazán paranoiások vagyunk, akkor írjuk le kézzel a jelszavakat. Ha viszont annyira nem, akkor egyszerûen küldjük át ezeket az `lpr` parancsnak. Megfigyelhetjük, hogy minden sorban látható az iterációs szám és a hozzá tartozó egyszeri jelszó. Hasznos lehet a felhasználás szerinti felírni a jelszavakat.

14.5.5. A UNIX® jelszavak használatának leszûkítése

Az OPIE képes a bejelentkezéshez használt IP-címek alapján leszûkíteni a UNIX® jelszavak használatát. Ehhez az /etc/opieaccess használható, amely alapból megtalálható a rendszerünkön. Az opieaccess(5) man oldalán találhatjuk meg a rá vonatkozó információkat és az összes vele kapcsolatos biztonsági megfontolást.

Íme egy példa az opieaccess állományra:

```
permit 192.168.0.0 255.255.0.0
```

Ezzel a sorral megengedjük a UNIX® jelszavak használatát minden olyan felhasználó számára, akinek az IP-je illeszkedik a megadott címre és maszkra (ez viszont álcázással kijátszható).

Ha az opieaccess állományból egyetlen szabály sem illeszkedik, akkor alapértelmezés szerint nem engedélyezettek a nem OPIE típusú jelszavak.

14.6. A TCP kapcsolatok burkolása

Aki ismeri az inetd(8) programot, az már biztosan hallott a TCP kapcsolatok burkolásáról, eredeti nevén a a TCP wrapperekrõl. Azonban csak kevesek képesek felfogni ezek valódi hasznát. Úgy néz ki, mindenki csak tûzfalakon keresztül akarja megoldani a hálózati kapcsolatot kezelését. Habár a tûzfalakat sok mindenre fel lehet ugyan használni, egyetlen tûzfal nem képes például szövegesen válaszolni a kapcsolatok kezdeményezõinek. Ellenben bármelyik TCP-wrapper szoftver képes erre, sõt még többre is. A következõ néhány szakaszban szemügyre vesszük a TCP wrapperek számos lehetõségét, és ahol lehetséges, ott konfigurációs állományokkal is illusztráljuk ezek használatát.

A TCP burkoló szoftverek kiterjesztik az inetd képességeit minden alatta dolgozó szerverdémon támogatására. Ezzel a módszerrel meg lehet oldani a naplózást, üzenetek küldését a kapcsolatokhoz, a démonok elérhetõségének korlátozását stb. Noha ezen lehetõségek közül néhány tûzfallal is megvalósítható, ezzel nem csupán egy további védelmi réteget húzunk fel a rendszerünk köré, hanem túllépjük mindazt, amit egy tûzfallal irányítani lehet.

A TCP burkolók használatával hozzáadott funkcionalitás azonban nem helyettesít egy jó tûzfalat. A TCP kapcsolatok burkolását tûzfallal vagy más egyéb biztonsági megoldással együtt tudjuk csak eredményesen használni, viszont a rendszerünk biztonságában egy újabb remek védelmi vonalat képvisel.

Mivel lényegében ez az inetd beállításának kibõvítése, ezért a szakasz elolvasásához feltételezzük az inetd beállításával kapcsolatos tudnivalók ismeretét.

Bár az inetd(8) által indított programok nem egészen tekinthetõen "démonoknak",

> hagyományosan démonnak hívják ezeket. Ezért rájuk ebben a szakaszban is ezt a kifejezést használjuk.

14.6.1. Kezdeti beállítások

FreeBSD alatt a TCP burkolók használatának egyetlen feltétele csupán annyi, hogy az inetd parancsot a `-Ww` paraméterrel indítsuk az rc.conf állományból. Az egyébként az alapbeállítás. Természetesen nem árt, ha helyesen állítjuk be az /etc/hosts.allow állományt is, ellenkezõ esetben a syslogd(8) egyébként dobálni fogja errõl az üzeneteket.

> Eltérõen a TCP burkolók egyéb implementációitól, a hosts.deny állományt itt már nem használjuk. Minden beállítást az /etc/host.allow állományba kell raknunk.

A legegyszerûbb konfiguráció esetén a démonok kapcsolódását egyszerûen engedélyezhetjük vagy letilthatjuk az /etc/hosts.allow állományban szereplõ beállításokkal. A FreeBSD alapértelmezett beállításai szerint minden inetd által indított démonhoz lehet kapcsolódni. Ennek megváltoztatásával az alapkonfiguráció áttekintése után foglalkozunk.

Az alapkonfiguráció általában `démon : cím : cselekvés` alakú. Itt a `démon` egy olyan démonra utal, amelyet az `inetd` indított el. A `cím` egy érvényes hálózati név, IP-cím vagy szögletes zárójelek ([ ]) között megadott IPv6 formátumú cím. A cselekvést tartalmazó mezõ (`action`) lehet `allow` vagy `deny` annak megfelelõen, hogy engedélyezzük vagy tiltjuk a megadott címrõl a csatlakozást. Nem szabad elfelejtenünk, hogy az így megadott beállítások közül mindig az elsõként illeszkedõ érvényesül, ami arra utal, hogy a konfigurációs állományban szereplõ szabályok egymás után növekvõ sorrendben értékelõdnek ki. Ha valamelyikük illeszkedik, akkor a keresés megáll.

Rengeteg egyéb opció is megadható még, de ezekrõl csak a késõbbi szakaszokban fogunk szólni. Egy egyszerû konfigurációs állomány már ennyi információból is könnyedén összeállítható. Például, ha engedélyezni szeretnénk a POP3 kapcsolatokat a mail/qpopper démonon keresztül, akkor a következõ sorral kell kiegészítenünk a hosts.allow állományt:

```
# Ez a sor kell a POP3 kapcsolatokhoz:
qpopper : ALL : allow
```

Miután hozzáadtuk ezt a sort, az inetd szervert újra kell indítanunk. Ezt vagy a kill(1) paranccsal, vagy pedig az /etc/rc.d/inetd szkript restart paraméterével tehetjük meg.

14.6.2. Komolyabb beállítások

A TCP kapcsolatok burkolásánál is meg lehet adni további opciókat. Segítségükkel még jobban irányítani tudjuk a kapcsolatok kezelésének módját. Néhány esetben az is hasznos lehet, ha küldünk valamilyen választ az egyes gépeknek vagy démonoknak. Máskor szükségünk lehet a csatlakozások naplózására vagy e-mailen keresztüli jelzésére a rendszergazda felé. Teljesen más helyezetekben csak a helyi hálózatunkról engedjük meg a csatlakozást. Ez mind lehetséges a `helyettesítõ jelekként` ismert beállítási opciók, kiterjesztõ karakterek és külsõ parancsok végrehajtásának használatával. A következõ két szakasz az ilyen és ehhez hasonló szituációk megoldására íródott.

Külsõ parancsok

Tegyük fel, hogy olyan helyezetben vagyunk, amikor a kapcsolatot tiltani akarjuk, de közben azért szeretnénk errõl értesíteni a kapcsolatot kezdeményezõ felet is. Hogyan tudjuk ezt megcsinálni? Ezt a `twist` nevû opcióval tehetjük meg. Amikor megpróbál valaki csatlakozni, akkor a `twist` hívódik meg és végrehajt egy megadott parancsot vagy szkriptet. Erre találunk is egy példát a hosts.allow állományban:

```
# The rest of the daemons are protected.
ALL : ALL \
        : severity auth.info \
        : twist /bin/echo "You are not welcome to use %d from %h."
```

Ez a példa a következõ üzenetet jeleníti meg: "You are not allowd to use `a démon neve` from `hálózati név`." (Jelentése: "A `démon neve` démont nem érheti el a `hálózati név` helyrõl!") Ez minden olyan démon esetén megjelenik, amirõl nem nyilatkoztunk korábban az állományban. Ezzel nagyon könnyen vissza tudunk küldeni egy választ a kapcsolat kezdményezõje felé, miután a kapcsolatot eldobtuk. Vegyük észre, hogy a visszaküldendõ üzenetet `"` karakterek közé *kell* tennünk, ez alól semmi sem kivétel.

DoS támadást lehet elõidézni azzal, ha egy támadó vagy támadók egy csoportja csatlakozási kérelmekkel kezdi el bombázni a démonainkat.

Ilyen esetekben használhatjuk a `spawn` opciót is. A `spawn` a `twist` opcióhoz hasonlóan implicit módon tiltja a kapcsolódást és arra használható, hogy lefuttassunk vele egy parancsot vagy szkriptet. A `spawn` azonban a `twist` opciótól eltérõen nem küld vissza semmilyen választ a kapcsolatot létrehozni kívánó egyénnek. Ehhez példaként vegyük a következõ sort a konfigurációs állományban:

```
# We do not allow connections from example.com:
ALL : .example.com \
    : spawn (/bin/echo %a from %h attempted to access %d >> \
      /var/log/connections.log) \
    : deny
```

Ezzel a `*.example.com` címtartományból érkezõ összes kapcsolódási kísérlet sikertelen lesz, miközben ezzel egyidõben a /var/log/connections.log állományba rögzítjük a csatlakozni akaró egyén hálózati nevét, IP-címét és a démont.

A korábban már kifejtett helyettesítõ karakterek túl, mint például az `%a`, még léteznek továbbiak is. Róluk a hosts_access(5) man oldalon találhatjuk meg a teljes listát.

Helyettesítõ jelek

Az eddigi példákban folyamatosan csak az `ALL` opciót adtuk meg. Azonban rajta kívûl léteznek mások is, amivel a megoldás funkcionalitását még egy kicsivel tovább növelhetjük. Például az `ALL` használható egy démon, egy tartomány vagy egy IP-cím illesztésére. A másik ilyen helyettesítõ jel a `PARANOID`, amelyet olyan gépek IP-címének illesztésekor alkalmazhatunk, ami feltételezhetõen

hamis. Más szóval a PARANOID olyan cselekvések megadását teszi lehetõvé, amelyek akkor hajtódnak végre, amikor a kapcsolatot létrehozó gép IP-címe eltér a hálózati nevétõl. A most következõ példa valószínûleg segít fényt deríteni ennek lényegére:

```
# Block possibly spoofed requests to sendmail:
sendmail : PARANOID : deny
```

A példában minden olyan kapcsolatkérést elutasítunk, ami a sendmail felé a hálózati névtõl eltérõ IP-címrõl irányul.

Ha rossz DNS beállításokat használunk, a PARANOID megadásával súlyosan mozgásképtelenné tehetjük a kliensünket vagy szerverünket. Ezért legyünk óvatosak vele!

A helyettesítõ jelekrõl és hozzájuk tartozó további lehetõségekrõl a hosts_access(5) man oldalon tájékozódhatunk.

A hosts.allow állományból ki kell venni az elsõ sort ahhoz, hogy bármilyen egyéb konfigurációs beállítás mûködõképes legyen. Ezt említettük a szakasz elején is.

14.7. KerberosIV

A Kerberos egy olyan járulékos rendszer/protokoll, amellyel a felhasználók egy biztonságos szerver szolgáltatásain keresztül tudják hitelesíteni magukat. Ilyen szolgáltatás többek közt a távoli bejelentkezés, távoli másolás, a rendszeren belüli biztonságos másolás és minden olyan egyéb veszélyes feladat, amit számottevõen megbízhatóbbá és irányíthatóbbá tettek.

A következõ utasítások a FreeBSD-hez mellékelt Kerberos beállításához adnak útmutatást. A teljes leíráshoz azonban érdemes fellapoznunk a menet közben hivatkozott man oldalakat is.

14.7.1. A KerberosIV telepítése

A Kerberos a FreeBSD egyik választható komponense. Legkönnyebben úgy tudjuk feltelepíteni, ha a FreeBSD telepítése során a sysinstall programban kiválasztjuk a krb4 vagy krb5 terjesztések valamelyikét. Ezzel felrakhatjuk a Kerberos "eBones" (KerberosIV) vagy "Heimdal" (Kerberos5) elnevezésû változatait. A FreeBSD azért tartalmazza ezeket az implementációkat, mert nem az Amerikai Egyesült Államokban vagy Kanadában fejlesztették, így az Egyesült Államok titkosításokkal kapcsolatos kiviteli korlátozások korában minden olyan rendszer adminisztrátora el tudta érni, aki nem ezekben az országokban lakott.

A Kerberos MIT által fejlesztett implementációját egyébként a Portgyûjteményõl a security/krb5 porton keresztül érhetjük el.

14.7.2. A kezdeti adatbázis létrehozása

Ezt a lépést csak a Kerberos szerveren kell elvégezni. Elõször is gyõzõdjünk meg róla, hogy semmilyen korábbi Kerberos adatbázis nem található a gépen. Váltsunk az /etc/kerberosIV

könyvtárra és ellenõrizzük a következõ állományok meglétét:

```
# cd /etc/kerberosIV
# ls
README      krb.conf        krb.realms
```

Ha rajtuk kívül további állományok is feltûnnének (mint például a principal.* vagy master_key), akkor a `kdb_destroy` paranccsal pusztítsuk el a régi Kerberos adatbázist, vagy ha nem fut már a Kerberos, akkor egyszerûen csak töröljük le ezeket.

Ezután lássunk neki a krb.conf és krb.realms állományok átírásán keresztül a Kerberos egyes övezeteinek (realm) létrehozásához. Itt most az `EXAMPLE.COM` lesz a létrehozandó övezet, a hozzá tartozó szerver pedig a `grunt.example.com`. Így szerkesszük át vagy készítsünk el a neki megfelelõ krb.conf állományt:

```
# cat krb.conf
EXAMPLE.COM
EXAMPLE.COM grunt.example.com admin server
CS.BERKELEY.EDU okeeffe.berkeley.edu
ATHENA.MIT.EDU kerberos.mit.edu
ATHENA.MIT.EDU kerberos-1.mit.edu
ATHENA.MIT.EDU kerberos-2.mit.edu
ATHENA.MIT.EDU kerberos-3.mit.edu
LCS.MIT.EDU kerberos.lcs.mit.edu
TELECOM.MIT.EDU bitsy.mit.edu
ARC.NASA.GOV trident.arc.nasa.gov
```

A többi övezetnek valójában nem feltétlenül kell itt lennie. Ezek csupán azért szerepelnek itt, hogy bemutassák miként lehet egyetlen géphez hozzárendelni egyszerre több övezetet is. Az egyszerûség kedvéért nyugodtan elhagyhatóak.

Az elsõ sor nevezi meg a rendszer által mûködtetett övezeteket. Az utána következõ sorokban övezeteket és hálózati neveket láthatunk. Itt az elsõ elem egy övezetet nevez meg, a második elem pedig az övezet "kulcselosztó központját" (key distribution center). A hálózati nevet követõ `admin server` kulcsszavak arra utalnak, hogy az adott gép adminisztratív szerepet ellátó adatbázist is tartalmaz. Ezeket a fogalmakat részleteiben a Kerberos man oldalain ismerhetjük meg.

Ezután hozzá kell adnunk a `grunt.example.com` nevû gépet az `EXAMPLE.COM` övezethez, valamint az `.example.com` tartományban levõ összes géphez létre kell hoznunk egy bejegyzést az `EXAMPLE.COM` övezetben. A krb.realms állományt ehhez a következõképpen kellene módosítanunk:

```
# cat krb.realms
grunt.example.com EXAMPLE.COM
.example.com EXAMPLE.COM
.berkeley.edu CS.BERKELEY.EDU
.MIT.EDU ATHENA.MIT.EDU
.mit.edu ATHENA.MIT.EDU
```

Ismét hozzátesszük, hogy a többi övezetnek nem kötelezõ itt szerepelnie. Ezek csupán azt demonstrálják, hogy miként kell egy gépet egyszerre több övezethez is beállítani. Az átláthatóság kedvéért minden további nélkül eltávolíthatjuk ezeket.

Itt az elsõ sor az *adott* rendszert elhelyezi egy nevesített övezetbe. A többi sor azt mutatja meg, hogyan kell alapértelmezett módon a meghatározott altartományokba tartozó gépeket egy nevesített övezethez hozzárendelni.

Most már készen állunk az adatbázis létrehozására. Ehhez egyedül a Kerberos szerverét (avagy Kulcselosztó központját) kell elindítanunk. Adjuk ki a `kdb_init` parancsot:

```
# kdb_init
Realm name [default  ATHENA.MIT.EDU ]: EXAMPLE.COM
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.

Enter Kerberos master key:
```

Az üzenet fordítása:

```
Most az adatbázis mesterkulcsát kell megadni.  Fontos, hogy
NE FELEJTSÜK EL ezt a jelszót.
```

Most el kell mentenünk a kulcsot, így a helyi gépen futó szerverek fel tudják szedni. Ehhez a `kstash` parancsra van szükségünk:

```
# kstash

Enter Kerberos master key:

Current Kerberos master key version is 1.

Master key entered. BEWARE!
```

Az üzenet fordítása:

```
A Kerberos mesterkulcsának jelenlegi változata: 1.

VIGYÁZAT, megadták a mesterkulcsot!
```

Ez elmenti a titkosított mesterkulcsot az /etc/kerberosIV/master_key állományba.

14.7.3. Az egész beüzemelése

Mindegyik Kerberosszal õrzött rendszerrel kapcsolatban két ún. szereplõt (principal) kell még hozzátennünk az adatbázishoz. A nevük `kpasswd` és `rcmd`. Minden rendszerhez létre kell hoznunk

ezeket a szereplõket, példányonként (instance) az egyes rendszerek neveivel.

A kpasswd és rcmd démonok teszik lehetõvé a többi rendszer számára, hogy megváltoztathassák a Kerberos jelszavukat, valamint hogy futtathassák az rcp(1), rlogin(1) és rsh(1) parancsokat.

Vegyük fel ezeket a bejegyzéseket is:

```
# kdb_edit
Opening database...

Enter Kerberos master key:

Current Kerberos master key version is 1.

Master key entered.  BEWARE!
Previous or default values are in [brackets] ,
enter return to leave the same, or new value.

Principal name: passwd
Instance: grunt

<Not found>, Create [y] ? y

Principal: passwd, Instance: grunt, kdc_key_ver: 1
New Password:                    <---- írjuk be, hogy "RANDOM"
Verifying password

New Password: <---- írjuk be, hogy "RANDOM"

Random password [y] ? y

Principal's new key version = 1
Expiration date (enter yyyy-mm-dd) [ 2000-01-01 ] ?
Max ticket lifetime (*5 minutes) [ 255 ] ?
Attributes [ 0 ] ?
Edit O.K.
Principal name: rcmd
Instance: grunt

<Not found>, Create [y] ?

Principal: rcmd, Instance: grunt, kdc_key_ver: 1
New Password:       <---- írjuk be, hogy "RANDOM"
Verifying password

New Password:           <---- írjuk be, hogy "RANDOM"

Random password [y] ?

Principal's new key version = 1
Expiration date (enter yyyy-mm-dd) [ 2000-01-01 ] ?
```

```
Max ticket lifetime (*5 minutes) [ 255 ] ?
Attributes [ 0 ] ?
Edit O.K.
Principal name:         <---- ha nem adunk meg semmit, akkor kilép
```

14.7.4. A szerver állomány létrehozása

Most pedig kivonatolni kell azokat a példányokat, amelyek szolgáltatást definiálnak a gépen. Erre az `ext_srvtab` parancsot használjuk. Ennek eredményeképpen keletkezik egy állományt, amelyet *biztonságos eszközökkel* át kell másolni vagy át kell mozgatni az egyes Kerberos kliensek /etc könyvtárába. Ennek az állománynak egyaránt jelent kell lennie a szerveren és a kliensen is, nélküle a Kerberos mûködésképtelen.

```
# ext_srvtab grunt
Enter Kerberos master key:

Current Kerberos master key version is 1.

Master key entered. BEWARE!
Generating 'grunt-new-srvtab'....
```

Ez a parancs most létrehozott egy ideiglenes állományt, amit át kell nevezni az srvtab névre, hogy megtalálhassák a szerverek. Az eredeti rendszeren a mv(1) paranccsal tudjuk a helyére rakni:

```
# mv grunt-new-srvtab srvtab
```

Ha egy kliensnek szánjuk az állományt és a hálozatunkat nem tekinthetjük biztonságosnak, akkor a kliens-new-srvtab állományt másoljuk egy mozgatható adathordozóra és megbízható módon jutassuk el. Ne felejtsük el az állományt srvtab néven átrakni a kliens /etc könyvtárába és az engedélyeit 600-ra állítani:

```
# mv grumble-new-srvtab srvtab
# chmod 600 srvtab
```

14.7.5. Az adatbázis feltöltése

Ezt követõen rögzítenünk kell néhány felhasználót is adatbázisban. Elõször is hozzunk létre egy bejegyzést a `janos` nevû felhasználónak. Ezt a `kdb_edit` parancs kiadásával tesszük meg:

```
# kdb_edit
Opening database...

Enter Kerberos master key:

Current Kerberos master key version is 1.
```

```
Master key entered.  BEWARE!
Previous or default values are in [brackets] ,
enter return to leave the same, or new value.

Principal name: janos
Instance:

<Not found>, Create [y] ? y

Principal: janos, Instance: , kdc_key_ver: 1
New Password:                <---- adjunk meg egy biztonságos jelszót
Verifying password

New Password:                <---- itt ismét adjuk meg a jelszót
Principal's new key version = 1
Expiration date (enter yyyy-mm-dd) [ 2000-01-01 ] ?
Max ticket lifetime (*5 minutes) [ 255 ] ?
Attributes [ 0 ] ?
Edit O.K.
Principal name:        <---- ha nem írunk be semmit, akkor kilép
```

14.7.6. Próbáljuk ki

Elsõként a Kerberos démonait kell beindítanunk. Ezzel kapcsolatban megjegyeznénk, hogy ha ehhez megfelelõen átírtuk az /etc/rc.conf állományunkat, akkor ez az újraindítással együtt magától lezajlik. Ezt csak a Kerberos szerveren kell megcsinálni. A Kerberos kliensei maguktól összeszedik a mûködésükhöz szükséges adatokat az /etc/kerberosIV könyvtárból.

```
# kerberos &
Kerberos server starting
Sleep forever on error
Log file is /var/log/kerberos.log
Current Kerberos master key version is 1.

Master key entered. BEWARE!

Current Kerberos master key version is 1
Local realm: EXAMPLE.COM
# kadmind -n &
KADM Server KADM0.0A initializing
Please do not use 'kill -9' to kill this job, use a
regular kill instead

Current Kerberos master key version is 1.

Master key entered.  BEWARE!
```

A fenti figyelmeztetés fordítása:

```
A program leállítására ne a 'kill -9' parancsot, hanem a
normális kill parancsot használjuk
```

Ezután a `kinit` parancs használatával próbáljunk meg az elõbb létrehozott `janos` azonosítónak kérni egy jegyet:

```
% kinit janos
MIT Project Athena (grunt.example.com)
Kerberos Initialization for "janos"
Password:
```

A `klist` paranccsal most próbáljuk meg kilistázni a tokeneket és így ellenõrizni, hogy valóban rendelkezünk velük:

```
% klist
Ticket file:    /tmp/tkt245
Principal:      janos@EXAMPLE.COM

  Issued           Expires          Principal
Apr 30 11:23:22  Apr 30 19:23:22  krbtgt.EXAMPLE.COM@EXAMPLE.COM
```

Ezután a passwd(1) használatával próbáljuk meg megváltoztatni a jelszavunkat. Ezzel tudjuk ellenõrizni, hogy a kpasswd démon hozzáfér a Kerberos adatbázisához:

```
% passwd
realm EXAMPLE.COM
Old password for janos:
New Password for janos:
Verifying password
New Password for janos:
Password changed.
```

14.7.7. Adminisztrátori jogosultságok felvétele

A Kerberos lehetõvé teszi, hogy *mindegyik* olyan felhasználónak, akinek rendszergazdai jogokra lenne szüksége, a su(1) eléréséhez *külön* meg tudjunk adni egy jelszót. Most már tudunk mondani egy olyan azonosítót is, amely jogosult a su(1) használatával `root` jogokat szerezni. Ezt úgy tudjuk megoldani, ha az adott szereplõhöz társítunk egy `root` példányt. A `kdb_edit` használatával készíteni tudunk egy `janos.root` bejegyzést a Kerberos adatbázisában:

```
# kdb_edit
Opening database...

Enter Kerberos master key:
```

```
Current Kerberos master key version is 1.

Master key entered.  BEWARE!
Previous or default values are in [brackets] ,
enter return to leave the same, or new value.

Principal name: janos
Instance: root

<Not found>, Create [y] ? y

Principal: janos, Instance: root, kdc_key_ver: 1
New Password:                    <---- ide csak egy BIZTONSÁGOS jelszót adjuk meg!
Verifying password

New Password:            <---- adjuk meg ismét a jelszót

Principal's new key version = 1
Expiration date (enter yyyy-mm-dd) [ 2000-01-01 ] ?
Max ticket lifetime (*5 minutes) [ 255 ] ? 12 <--- ne állítsuk nagyon hosszúra!
Attributes [ 0 ] ?
Edit O.K.
Principal name:              <---- ha nem adunk meg semmit, akkor kilép
```

Ezt követõen úgy tudunk megbizonyosodni a mûködésérõl, hogy megpróbálunk neki tokeneket szerezni:

```
# kinit janos.root
MIT Project Athena (grunt.example.com)
Kerberos Initialization for "janos.root"
Password:
```

Most rakjuk bele a felhasználót a `root`.klogin állományába:

```
# cat /root/.klogin
janos.root@EXAMPLE.COM
```

Ezután próbáljunk meg kiadni a su(1) parancsát:

```
% su
Password:
```

Nézzük meg milyen tokenjeink is vannak:

```
# klist
Ticket file:    /tmp/tkt_root_245
```

```
Principal:      janos.root@EXAMPLE.COM

  Issued           Expires          Principal
May  2 20:43:12  May  3 04:43:12  krbtgt.EXAMPLE.COM@EXAMPLE.COM
```

14.7.8. Más parancsok használata

Az iménti példában létrehoztunk egy janos nevû szereplõt, amihez a root egy példányát rendeltük. Ez egy olyan felhasználón alapján történt, akinek a neve megegyezik a hozzá tartozó szereplõvel, ami a Kerberosban alapértelmezés. Amennyiben a szükséges megjegyzések megtalálhatóak a root könyvtárában levõ .klogin állományban, akkor a felhasználó.root formátumú szereplõ.példány azonosító megengedi a felhasználó számára, hogy végrehajtsa a su(1) parancsot.

```
# cat /root/.klogin
janos.root@EXAMPLE.COM
```

Ehhez hasonlóan, ha a felhasználó saját könyvtárában megtalálható egy ilyen állomány:

```
% cat ~/.klogin
janos@EXAMPLE.COM
jozsef@EXAMPLE.COM
```

Ezzel a konfigurációval bárki, aki janos felhasználóként vagy jozsef felhasználóként (a kinit parancson keresztül) hitelesítette magát EXAMPLE.COM övezetbõl, ezen a rendszeren (grunt) bejelentkezhet a janos nevû felhasználóként vagy hozzáférhet az állományaihoz az rlogin(1), rsh(1) vagy rcp(1) használatával.

Például janos most egy másik Kerberost használó rendszerre jelentkezik be:

```
% kinit
MIT Project Athena (grunt.example.com)
Password:
% rlogin grunt
Last login: Mon May  1 21:14:47 from grumble
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
        The Regents of the University of California.  All rights reserved.

FreeBSD BUILT-19950429 (GR386) #0: Sat Apr 29 17:50:09 SAT 1995
```

Vagy jozsef jelentkezik be ugyanazon a gépen janos hozzáférésével (a janos nevû felhasználónak a fentebb bemutatt .klogin állomány található a könyvtárában és a Kerberos üzemeltetéséért felelõs személy létrehozott egy *jozsef* nevû szereplõt egy null példánnyal):

```
% kinit
% rlogin grunt -l janos
```

```
MIT Project Athena (grunt.example.com)
Password:
Last login: Mon May  1 21:16:55 from grumble
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
        The Regents of the University of California.  All rights reserved.
FreeBSD BUILT-19950429 (GR386) #0: Sat Apr 29 17:50:09 SAT 1995
```

14.8. Kerberos5

A FreeBSD 5.1 után következõ mindegyik FreeBSD kiadás már csak a Kerberos5 támogatást tartalmaz. Ezért bennük csak a Kerberos5 található meg, és a beállítása sok szempontból hasonlít a KerberosIV beállításához. A most következõ információk csak és kizárólag a FreeBSD 5.0 kiadás után következõkben található Kerberos5 változatra vonatkoznak. A KerberosIV szolgáltatásait a felhasználók csomagként, a security/krb4 porton keresztül érhetik el.

A Kerberos egy hálózati kiegészítõ rendszer/protokoll, amivel a felhasználók egy biztonságos szerveren keresztül képesek magukat azonosítani. A távoli bejelentkezések, távoli másolások, a rendszer belüli védett másolások valamint egyéb nagyon kockázatos feladatok, szolgáltatások biztonsága és felügyelete így jelentõs mértékben javítható.

A Kerberos úgy írható le, mint az személyazonosságok ellenõrzésére feljogosított rendszer. Vagy tekinthetjük egy megbízható külsõ megfigyelõ által végzett hitelesítési rendszernek is. A Kerberos csak egyetlen funkciót kínál fel - ez a felhasználók biztonságos hitelesítése a hálózaton. Viszont nem nyújt semmilyen felhatalmazási (mit csinálhatnak a felhasználók) vagy vizsgálati (mit csináltak végül a felhasználók) lehetõséget. Miután egy kliens és a szerver a Kerberos használatával azonosították egymást, az egymás közt folyó kommunikációjuk titkosításával képesek megõrzi az átáramló adatok sértetlenségét és lehallgathatatlanságát.

Ennek tükrében a Kerberos használata csak más olyan biztonsági módszerekkel együttesen javasolt, amelyek felhatalmazást és vizsgálati szolgáltatásokkal is rendelkeznek.

A most következõ utasítások arra igyekeznek útmutatást adni, hogy miként használjuk a FreeBSD-vel együtt terjesztett Kerberos verziót. Azonban a teljes leírást csak a témához tartozó man oldalak átolvasásával együtt kapjuk meg.

A Kerberos telepítésének bemutatásához az alábbi névtereket fogjuk használni:

- A DNS tartomány ("zóna") az `example.org` lesz.
- A Kerberos övezet az EXAMPLE.ORG lesz.

Kérjük, hogy még abban az esetben is valódi tartományneveket adjuk meg, amikor a Kerberos használatát csak a belsõ hálózaton tervezzük. Ezzel elkerülhetjük az egyes Kerberos övezetek együttmûködése során felmerülõ DNS problémákat.

14.8.1. A Kerberos története

A Kerberost az MIT hozta létre a hálózati biztonsággal kapcsolatos problémák egyik megoldásaként. A Kerberos erõs titkosítást használ, ezért a kliensek képesek egy nem biztonságos hálózaton is

azonosítani magukat a szerver felé (és fordítva).

A Kerberos egyaránt utal egy hálózati protokoll nevére és azokra programokra, amelyek implementálják (például Kerberos telnet). Az 5 a protokoll jelenlegi verziója, amit az RFC 1510 ír le.

A protokollnak számos szabad változata létezik, rengeteg típusú operációs rendszerre. A Massachusettsi Mûszaki Intézet (Massachusetts Institute of Technology, MIT), ahol a Kerberost eredetileg kifejlesztették, napjainkban is folytatja a saját Kerberos csomagjának fejlesztését. Többnyire az Egyesült Államokban használják titkosításra, mivel régebben az amerikai kiviteli korlátozások voltak rá érvényesek. Az MITKerberos változata portként érhetõ el (security/krb5). A Heimdal Kerberos egy másik 5 verziójú implementáció, amit a kiviteli korlátozások elkerülése érdekében határozottan az Egyesült Államokon kívül fejlesztettek ki (ezért gyakran megtalálhatjuk a különbözõ nem kereskedelmi UNIX® variánsokban). A Heimdal Kerberos terjesztés portként elérhetõ (security/heimdal) és kisebb méretben a FreeBSD alaptelepítésének is része.

Mivel ezzel az írással a legtöbb felhasználót kívánjuk segíteni, ezért a következõ utasítások a FreeBSD telepítésében mellékelt Heimdal terjesztés használatát feltételezik.

14.8.2. A Heimdal kulcselosztójának telepítése

A kulcselosztó központ (Key Distribution Center, avagy KDC) az a centralizált hitelesítési szolgáltatás, amit a Kerberos nyújt - lényegében az a számítógép, amely Kerberos-jegyeket bocsájt ki. A KDC"megbízhatónak" tekinthetõ a Kerberos által kialakított övezetben levõ többi számítógép számára, ezért védelme kiemelten fontos.

Itt jegyeznénk meg, hogy habár a Kerberos szerver futtatása nagyon kevés számítógépes erõforrást igényel, ennek ellenére biztonsági szempontból egy külön számítógépet javasoljunk a kulcselosztó szerepének betöltéséhez.

Mielõtt nekifognánk a KDC konfigurálásának, ellenõrizzük, hogy az /etc/rc.conf tartalmazza a KDC mûködéséhez szükséges beállításokat (az elérési utakat természetesen a saját rendszerünk szerint állítsuk be):

```
kerberos5_server_enable="YES"
kadmind5_server_enable="YES"
```

A következõ lépésben vegyük szemügyre a Kerberos beállításait tartalmazó /etc/krb5.conf állományt:

```
[libdefaults]
    default_realm = EXAMPLE.ORG
[realms]
    EXAMPLE.ORG = {
        kdc = kerberos.example.org
        admin_server = kerberos.example.org
    }
[domain_realm]
    .example.org = EXAMPLE.ORG
```

Vegyük észre, hogy az itt szereplõ /etc/krb5.conf állomány szerint a kulcselosztónk teljes hálózati neve `kerberos.example.org`. Ha a kulcselosztónknak nem ez a neve, akkor a zónákat leíró állományba vegyünk még fel egy ilyen CNAME (álnév) bejegyzést.

Ha egy nagyobb hálózatban vagyunk, ahol a DNS szervert is megfelelõen beállították, akkor az iménti példa ennyire leszûkíthetõ:

```
[libdefaults]
      default_realm = EXAMPLE.ORG
```

Itt már a következõ sorokat hozzáadták `example.org` zónát leíró állományhoz:

```
_kerberos._udp      IN  SRV     01 00 88 kerberos.example.org.
_kerberos._tcp      IN  SRV     01 00 88 kerberos.example.org.
_kpasswd._udp       IN  SRV     01 00 464 kerberos.example.org.
_kerberos-adm._tcp  IN  SRV     01 00 749 kerberos.example.org.
_kerberos           IN  TXT     EXAMPLE.ORG
```

A kliensek csak akkor lesznek képesek elérni a Kerberos szolgáltatásait, ha vagy *kötelezõ jelleggel* megadunk egy teljesen beállított /etc/krb5.conf állományt, vagy egy minimális /etc/krb5.conf állományt *és* egy helyesen beállított DNS szervert használunk.

Ezután létrehozzuk a Kerberos adatbázisát. Ez az adatbázis tartalmazza az összes szereplõ kulcsát a mesterkulcssal titkosítva. Erre a jelszóra nem kell feltétlenül emlékeznünk, mivel ez egy állományban tárolódik (/var/heimdal/m-key). A mesterkulcsot a `kstash` parancs kiadásával és egy jelszó megadásával tudjuk létrehozni.

Ahogy a mesterkulcs elkészült, a `kadmin` parancs `-l` (mint "lokális", azaz helyi) opciójával inicializálni tudjuk az adatbázist. Ez az opció arra utasítja a `kadmin` programot, hogy ne a `kadmind` hálózati szolgáltatást használja, hanem közvetlenül az adatbázis állományait módosítsa. Ezzel oldható meg az adatbázis kezdeti létrehozásának problémája. Miután megkaptuk a `kadmin` parancssorát, az övezetünkhöz tartozó adatbázis inicializálásához adjuk ki az `init` parancsot.

Végül, még mindig a `kadmin` parancssorát használva, az `add` paranccsal hozzuk létre az elsõ szereplõnket. Egyelõre érjük be az alapértelmezett értékekkel, a `modify` paranccsal késõbb úgyis meg tudjuk változtatni ezeket. Hozzátesszük, hogy itt a `?` parancs segítségével bármikor lekérhetjük az opciók ismertetését.

Példa egy adatbázis létrehozására:

```
# kstash
Master key: xxxxxxxx
Verifying password - Master key: xxxxxxxx

# kadmin -l
```

```
kadmin> init EXAMPLE.ORG
Realm max ticket life [unlimited]:
kadmin> add tillman
Max ticket life [unlimited]:
Max renewable life [unlimited]:
Attributes []:
Password: xxxxxxxx
Verifying password - Password: xxxxxxxx
```

Most már ideje elindítani a KDC szolgáltatásait. Ezeket az `/etc/rc.d/kerberos start` és `/etc/rc.d/kadmind start` parancsok kiadásával tudjuk felhozni. Megjegyezzük, hogy most még semmilyen kerberizált démont nem kell elindítanunk. Ellenben igyekezzünk ellenõrizni a KDC mûködõképességét azzal, hogy KDC parancssorából kérünk egy jegyet a frissen hozzáadott szereplõnknek (felhasználónknak) és kilistázzuk:

```
% kinit tillman
tillman@EXAMPLE.ORG's Password:

% klist
Credentials cache: FILE:/tmp/krb5cc_500
    Principal: tillman@EXAMPLE.ORG

  Issued           Expires          Principal
Aug 27 15:37:58  Aug 28 01:37:58  krbtgt/EXAMPLE.ORG@EXAMPLE.ORG
```

Miután végeztünk, nyugodtan törölhetjük a jegyet:

```
% kdestroy
```

14.8.3. Szerverek kerberizálása a Heimdal használatával

Ehhez elõször is szükségünk lesz a Kerberos konfigurációs állományának, az /etc/krb5.conf másolatára. Ezt úgy tudjuk megtenni, ha egyszerûen átmásoljuk a kulcselosztóról az egyik kliensre valamilyen megbízható módon (vagy az scp(1) programhoz hasonló hálózati segédprogramok, vagy például fizikailag egy floppy lemez használatával).

Ezután szükségünk lesz egy /etc/krb5.keytab nevû állományra. Ez az alapvetõ különbség a kerberizált démonokat felkínáló szerver és egy munkaállomás közt - a szervernek rendelkeznie kell egy keytab állománnyal. Ez az állomány tartalmazza a szerver kulcsát, amivel így a kulcselosztóval kölcsönösen azonosítani tudják egymást. Ezt a szerverre biztonságosan kell eljuttatnunk, mivel ennek napvilágra kerülésével a szerver védelme komoly veszélybe kerül. Tehát, ha egy titkosítás nélküli csatornán, például FTP-n keresztül visszük át, akkor kifejezetten rossz ötlet.

A szerverre általában a `kadmin` program használatával érdemes átvinni a keytab állományt. Ez azért is hasznos, mert ehhez a `kadmin` segítségével létre kell hoznunk a befogadó szereplõt is (a kulcselosztó a krb5.keytab állomány végén).

Vegyük észre, hogy már kaptunk egy jegyet és ezzel a jeggyel jogosultaknak kell lennünk a kadmind.acl állomány `kadmin` felület használatára. A hozzáférést vezérlõ listák (ACL-ek) tervezésével kapcsolatban olvassuk el Heimdal info oldalán található "Remote administration" címû szakaszt (`info heimdal`). Amennyiben nem kívánjuk engedélyezni a `kadmin` távoli elérését, egyszerûen csak csatlakozzunk valamilyen biztonságos módon (helyi konzolon, ssh(1) vagy egy kerberizált telnet(1) használatával) a kulcselosztóhoz, és a `kadmin -l` paranccsal végezzük el helyben az adminisztrációt.

Miután telepítettük az /etc/krb5.conf állományt, a Kerberos szerverrõl el tudjuk érni a `kadmin` felületét. Az `add --random-key` paranccsal most már hozzáadhatjuk a szerver befogadó szereplõjét és az `ext` paranccsal ki tudjuk vonni a szerver befogadó szereplõjét a saját keytab állományából. Például:

```
# kadmin
kadmin> add --random-key host/myserver.example.org
Max ticket life [unlimited]:
Max renewable life [unlimited]:
Attributes []:
kadmin> ext host/myserver.example.org
kadmin> exit
```

Itt jegyeznénk meg, hogy az `ext` parancs (az "extract" rövdítése) a kivont kulcsot alapértelmezés szerint az /etc/krb5.keytab állományba menti ki.

Ha a kulcselosztón nem fut a `kadmind` szolgáltatás (valószínûleg biztonsági okokból) és ezért távolról nem tudjuk elérni a `kadmin` felületét, akkor így tudjuk közvetlenül hozzáadni a befogadó szereplõt (`host/myserver.EXAMPLE.ORG`), majd kivonatolni azt egy ideiglenes állományba (elkerülve az /etc/krb5.keytab felülírását):

```
# kadmin
kadmin> ext --keytab=/tmp/example.keytab host/myserver.example.org
kadmin> exit
```

Ezután valamilyen biztonságos eszközzel (például `scp` vagy floppy használatával) át tudjuk másolni keytab állományt a szerverre. A kulcselosztón levõ keytab felülírását elkerülendõ, ne feledkezzünk el egy megfelelõ név megadásáról sem.

Ezen a ponton már a szerver képes felvenni a kapcsolatot a kulcselosztóval (a krb5.conf állomány miatt) és bizonyítani a személyazonosságát (a krb5.keytab állomány miatt). Így tehát készen állunk a szolgáltatások kerberizálására. Ebben a példában most a `telnet` szolgáltatást vesszük célba úgy, hogy elõször az /etc/inetd.conf állományba berakjuk az alábbi sort, majd újraindítjuk az inetd(8) szolgáltatást az `/etc/rc.d/inetd restart` paranccsal:

```
telnet    stream  tcp     nowait  root    /usr/libexec/telnetd  telnetd -a user
```

Itt az a legfontosabb, hogy az `-a` (mint authentication, azaz hitelesítés) paramétert a "user"

beállítással adjuk meg. A telnetd(8) man oldalán olvashatunk ennek pontos részleteirõl.

14.8.4. Kliensek kerberizálása a Heimdal használatával

A kliensek beállítása szinte majdnem gyerekjáték. A Kerberos beállításához egyedül az /etc/krb5.conf állományra lesz szükségünk. Valamilyen biztonságos eszközzel másoljuk át a kulcselosztóról a kliensre.

Úgy tudjuk leteszteln klienst, ha megpróbáljuk róla kiadni a `kinit`, `klist` és `kdestroy` parancsokat a fentebb létrehozott szereplõ jegyének megszerzéséhez, lekérdezéséhez és megsemmisítéséhez. A Kerberos használatával megpróbálkozhatunk csatlakozni valamelyik kerberizált szerverre is, ha viszont ez nem mûködik még egy jegy megszerzése után sem, akkor a gond többnyire a szerverrel van, nem pedig a klienssel vagy a kulcselosztóval.

Amikor egy `telnet` vagy egy hozzá hasonló alkalmazást tesztelünk, egy csomaglehallgató (mint amilyen például a tcpdump(1)) elindításával gyõzödjünk meg róla, hogy a jelszavak ilyenkor titkosítva mennek át. Próbáljuk meg titkosítani a teljes kommunikációt a `telnet -x` paraméterével (hasonlóan az `ssh` parancshoz).

Alapból még számos más kiegészítõ Kerberos kliensalkalmazás is telepítõdik. Ezeken érezhetõ meg valójában az alaprendszerhez tartozó Heimdal változat "minimalitása": ebben a `telnet` az egyedüli kerberizált szolgáltatás.

A Heimdal port igyekszik pótolni a hiányzó klienseket a kerberizált `ftp`, `rsh`, `rcp`, `rlogin` és néhány kevéséb ismert program telepítésével. Az MIT változat portja szintén tartalmazza a Kerberos kliensek teljes kelléktárát.

14.8.5. A felhasználók konfigurációs állományai: a .k5login és a .k5users

Általában az övezetben található felhasználók mindegyikéhez tartozik egy Kerberos-szereplõ (mint például a `tillman@EXAMPLE.ORG`), ami a felhasználó helyi hozzáférésére mutat (mint például a `tillman` nevû helyi hozzáférés). A `telnet` és a hozzá hasonló kliensalkalmazások általában nem igényelnek felhasználót vagy szereplõt.

Elõfordulhat azonban, hogy valaki olyan szeretné elérni egy helyi felhasználó hozzáférését, aki nem rendelkezik a hozzá tartozó Kerberos-szereplõvel. Például a `tillman@EXAMPLE.ORG` nevû felhasználó el szeretné érni a helyi számítógépen levõ `webdevelopers` hozzáférést. Más szereplõk is elérhetik a helyi hozzáféréseket.

A probléma megoldásához a felhasználók könyvtárában található .k5login és a .k5users állományok használhatóak a .host és .rhosts állományok kombinációjához hasonlóan. Például a .k5login így néz ki:

```
tillman@example.org
jdoe@example.org
```

Ezt a `webdevelopers` nevû helyi felhasználó könyvtárában kell elhelyeznünk, így a felsorolt szereplõt megosztott jelszó használata nélkül képesek elérni a hozzáférést.

Az említett parancsok man oldalának elolvasása ajánlott. Megjegyezzük, hogy a `ksu` man oldal foglalkozik a .k5users állománnyal.

14.8.6. Tippek, trükkök a Kerberos használatáról és hibaelhárítás

- Akár a Kerberos Heimdal vagy az MIT változatát használjuk, ne felejtsük úgy beállítani a `PATH` környezeti változóban felsorolt elérési utakat, hogy a kliensalkalmazások kerberizált változatai a rendszerben használatos verziók elé kerüljenek.
- Az övezetben minden számítógép órája ugyanúgy jár? Ha nem, akkor a hitelesítés csõdöt mondhat. A Az órák egyeztetése az NTP használatávalból tudhatjuk meg hogyan szinkronizáljunk órákat az NTP segítségével.
- Az MIT és a Heimdal verziók a `kadmin` kivételével remekül megvannak egymással, mivel az általa használt protokollt még nem szabványosították.
- Ha megváltoztatjuk a gépünk hálózati nevét, akkor a ugyanígy a `host/` szereplõnket is meg kell változtatni és frissíteni a keytab állományunkat. Ez olyan speciális keytab bejegyzésekre is vonatkozik, mint például az Apache www/mod_auth_kerb moduljához tartozó `www/` szereplõ.
- Az övezetünkben levõ összes számítógépnek (mind a két irányba) feloldható DNS névvel kell rendelkeznie (vagy legalább egy /etc/hosts állománnyal). Erre a CNAME rekord megfelelõ, de az A és PTR rekordoknak mindenképpen rendben kell lenniük. Az ilyenkor keletkezõ hibaüzenet nem éppen fogja meg a lényeget: `Kerberos5 refuses authentication because Read req failed: Key table entry not found`.
- A kulcselosztó számára kliensként viselkedõ bizonyos operációs rendszerek nem állítják be megfelelõen a `ksu` engedélyeit, ezért nem lehet `root` jogokkal futtatni. Ezért a `ksu` parancs nem fog mûködni, ami alapvetõen nem egy rossz ötlet, de idegesítõ. Ez nem a kulcselosztó hibája.
- Ha a KerberosMIT változatát használjuk és a meg akarjuk hosszabbítani a szereplõknek kiadott jegyek élettartamát az alapértelmezett tíz óráról, akkor a `kadmin` felületén a `modify_principal` paranccsal tudjuk megváltoztatni mind a kérdéses szereplõ, mind pedig a `krbtgt` jegyeinek élettartamának maximumát. Ezt követõen a szereplõ a `kinit -l` opciójával tud egy nagyobb élettartammal rendelkezõ jegyet kérni.

Amikor egy kulcselosztóval kapcsolatos hibát próbálunk felderíteni a csomagok lehallgatásával, és a munkaállomásunkról kiadjuk a `kinit` parancsot, akkor arra lehetünk figyelmesek, hogy a TGT már egybõl a `kinit` indításakor átküldésre kerül - még mielõtt egyáltalán megadtuk volna a jelszavunkat! Ezt azzal lehet magyarázni, hogy a Kerberos szerver bármilyen hitelesítetlen kérésre elküld egy TGT-t (Jegyadó jegy, azaz Ticket Granting Ticket). Azonban mindegyik ilyen TGT a felhasználó jelszavából származtatott kulccsal titkosítódik. Ezért amit a felhasználó jelszóként megad, nem megy el a kulcselosztónak, hanem vele a `kinit` a már megkapott TGT-t kódolja ki. Amennyiben a visszakódolás egy érvényes idõbélyeggel rendelkezõ, használható jegyet eredményez, akkor a felhasználó érvényes Kerberos hitelesítést szerez. Ez a hitelesítés magában foglal egy kulcsot, amellyel a késõbbiekben a Kerberos szerverekkel tudjuk felvenni biztonságos módon a kapcsolatot, és rajta kívül egy újabb jegyadó jegyet, amelyet a Kerberos szerver a saját kulcsával titkosított. A titkosítás második vonala a felhasználó számára ismeretlen, de segítségével a Kerberos szerer képes ellenõrizni az egyes

jegyadó jegyek hitelességét.

- Ha a jegyeket hosszabb (például egyhetes) élettartammal akarjuk használni és a jegyeket tároló géphez OpenSSH segítségével csatlakozunk, akkor mindenképpen ellenõrizzük, hogy az sshd_config állományban a Kerberos `TicketCleanup` beállításának értéke `no`, máskülönben a kijelentkezés után automatikusan törlõdnek a jegyeink.
- Ne hagyjuk figyelmen kívül azt sem, hogy a befogadó szereplõk is rendelkezhetnek nagyobb élettartamú jegyekkel. Ha a felhasználónkhoz tartozó szereplõ jegye például egy hét alatt évül el, de a számítógép, amire bejelentkezük, csupán kilenc óráig tartja életben ezeket, akkor a jegyeket tároló gyorsítótárunkban hamarabb elévül a hozzá tartozó jegy, ami miatt pedig hibák keletkeznek.
- Ha a rossz jelszavak használata ellen beállítjuk a krb5.dic állományt (errõl a `kadmind` man oldalán találunk egy rövid leírást), akkor nem szabad elfelejteni, hogy ez csak olyan szereplõkre vonatkozik, akiknek a jelszavára is állítottunk be szabályozásokat. A krb5.dict állományok felépítési nem bonyolult: minden sorban egyetlen karakterlánc szerepel. Érdemes lehet például létrehozni ezen a néven egy szimbolikus linket a /usr/shared/dict/words állományra.

14.8.7. Eltérések az MIT porttól

A Heimdal és az MIT változatok közti egyik legnagyobb eltérés a `kadmin` programmal kapcsolatban van, ami eltérõ (de egyébként ekivalens) parancskészlettel rendelkezik és más protokollt használ. Ennek komoly következménye, hogy ha az MIT-féle kulcselosztót használjuk, akkor azt a Heimdal `kadmin` felületével nem tudjuk távolról adminisztrálni (és vica versa).

A kliensalkalmazások paraméterezése is eltérhet ugyanazon feladatoknál. Ezért velük kapcsolatban az MITKerberos honlapja (http://web.mit.edu/Kerberos/www/) a mérvadó. Vigyázzunk az elérési utakkal: az MIT port magát alapértelmezés szerint a /usr/local könyvtárba telepíti, ezért az általuk kiváltani kívánt "normális" rendszerprogramokat esetleg hamarabb találja meg a rendszer, ha nem jól állítottuk be a `PATH` környezeti változónkat.

Ha nem értjük, hogy miért mûködnek olyan furcsán a `telnetd` és a `klogind` által kezelt bejelentkezések, akkor olvassuk el a FreeBSD security/krb5 portjával települõ MIT változat /usr/local/shared/doc/krb5/README.FreeBSD állományt (angolul). Az a legfontosabb, hogy a `incorrect permissions on cache file` hiba eltüntetéséhez a `login.krb5` binárist kell használnunk, így a továbbított jogosultságoknak megfelelõen át tudja állítani a tulajdonost.

Az rc.conf állományt is módosítani kell a következõ beállítás kialakításához:

```
kerberos5_server="/usr/local/sbin/krb5kdc"
kadmind5_server="/usr/local/sbin/kadmind"
kerberos5_server_enable="YES"
kadmind5_server_enable="YES"
```

Erre azért van szükség, mert a KerberosMIT változata a /usr/local könyvtáron belülre telepíti fel a hozzá tartozó alkalmazásokat.

14.8.8. A Kerberosban talált korlátozások enyhítése

A Kerberos a "mindent vagy semmit" megközelítést követi

A hálózaton minden szolgáltatást módosítanunk kell ahhoz, hogy együtt tudjanak mûködni a Kerberosszal (vagy valamilyen más módon védenünk kell ezeket a támadások ellen), különben a felhasználók jogait el lehet lopni vagy újra fel lehet használni. Erre jó példa lehet az összes távoli parancssoros elérés (például az `rsh` valamint a `telnet`) kerberizálása, de a jelszavakat titkosítatlanul küldõ POP3 levelezõ szerver kihagyása.

A Kerberos az egyfelhasználós munkaállomások számára készült

Többfelhasználós környezetben a Kerberos már nem annyira biztonságos. Ez azért mondható el, mert a jegyeket a mindenki által olvasható /tmp könyvtárban tárolja. Ha az adott felhasználó számítógépét egyszerre több emberrel is megosztja (tehát többfelhasználós), akkor a felhasználó jegyeit egy másik felhasználó bármikor lemásolhatja (ellophatja).

Ezt a `-c` opció után megadott állománynévvel vagy (inkább) a `KRB5CCNAME` környezeti változó megfelelõ beállításával tudjuk áthidalni, habár ezt ritkán teszik is meg. Ha a felhasználók könyvtárában és a megfelelõ engedélyekkel tároljuk ezeket a jegyeket, akkor némileg visszaszoríthatjuk a probléma kockázatát.

A kulcselosztó a rendszer legsebezhetõbb pontja

A rendszer kialakításából fakadóan a kulcselosztónak legalább annyira megbízhatónak kell lennie, mint a rajta levõ központi jelszóadatbázisnak. A kulcselosztón semmilyen más szolgáltatás nem futhat és fizikailag is biztonságba kell helyezni. A kockázat nagy, mivel a Kerberos az összes jelszót ugyanazzal a kulcssal (a "mesterkulcssal") titkosítja, amelyet a kulcselosztó egy állományban tárol.

Széljegyzet gyanánt hozzátesszük, hogy a mesterkulcs elvesztése nem annyira rossz, mint azt elsõ gondolnánk. A mesterkulcsot csupán a véletlenszám-generátor inicializálásához használják a Kerberos adatbázisának titkosításakor. Amíg a kulcselosztóhoz nem tudnak illetéktelenek hozzáférni, addig nem tudnak sokat kezdeni a mesterkulccsal.

Mellesleg ha a kulcselosztó nem elérhetõ (talán pontosan egy DoS támadás vagy éppen hálózati problémák miatt), akkor a hitelesítés nem végezhetõ el, mivel így a hozzá szükséges hálózati szolgáltatások sem használhatóak. Ez remek eszköz egy DoS támadáshoz. Ezen több (egy központi és egy vagy több alárendelt) kulcselosztó telepítésével, valamint a másodlagos vagy tartalékként használt hitelesítési eszközök (a PAM erre tökéletes) körültekintõ megvalósításával enyhíthetünk.

A Kerberos hiányosságai

A Kerberos révén a felhasználók, számítógépek és szolgáltatások tudják egymást hitelesíteni. Ellenben semmilyen eszközt nem kínál fel a kulcselosztó hitelességének ellenõrzésére. Így tehát (például) egy eltérített `kinit` képes ellopni az összes felhasználói nevet és jelszót. Az ilyen incidensek elkerülésére a security/tripwire és a hozzá hasonló segédprogramok segítségével lehet megõrizni a rendszer sértelenségét.

14.8.9. Erõforrások és további információk

- A Kerberos GYIK (angolul)
- Egy hitelesítési rendszer kidolgozása: párbeszéd négy színben (angolul)
- RFC 1510: A Kerberos hálózati hitelesítési szolgáltatás (V5) (angolul)
- Az MIT Kerberos holnapja (angolul)
- A Heimdal Kerberos honlapja (angolul)

14.9. OpenSSL

A FreeBSD-hez adott OpenSSL az egyik olyan tényezõ, amit a legtöbb felhasználó figyelmen kívül hagy. Az OpenSSL egy titkosítási réteget nyújt a hagyományos kommunikációs csatorna felett, így rengeteg hálózati alkalmazásba és szolgáltatásba bele lehet szõni.

Az OpenSSL felhasználható többek közt a levelezõ kliensek titkosított hitelesítésére, hitelkártyás fizetések weben keresztüli lebonyolítására alkalmas, és még sok minden másra. Sok port, köztük a www/apache13-ssl és a mail/sylpheed-claws is felajánlja az OpenSSL felhasználását.

A legtöbb esetben a Portgyûjtemény megpróbálja lefordítani a security/openssl portot, hacsak a `WITH_OPENSSL_BASE` változót határozottan a "yes" értékre nem állítjuk.

A FreeBSD-hez mellékelt OpenSSL ismeri a Secure Sockets Layer v2/v3 (SSLv2/SSLv3) és Transport Layer Security v1 (TLSv1) hálózatbiztonsági protokollokat, és általános célú titkosítási könyvtárként is alkalmazható.

Noha az OpenSSL ismeri az IDEA algoritmusát is, az Egyesült Államokban érvényben levõ szabadalmak miatt alapértelmezés szerint nem engedélyezett. A használatához el kell olvasni a hozzá tartozó licencet, és ha elfogadjuk a benne foglaltakat, akkor állítsuk be a `MAKE_IDEA` változót a make.conf állományban.

Az OpenSSL-t leginkább a szoftverek tanúsítványainak elkészítéséhez használják. Ilyen tanúsítvánnyokkal lehet szavatolni, hogy az érte felelõs cég vagy egyén valóban megbízható és nem szélhámos. Amennyiben a kérdéses tanúsítványt nem vizsgálta be valamelyik "tanúsítványok hitelesítésével foglalkozó hatóság" (Certificate Authority, vagy CA), akkor errõl általában kap egy figyelmeztetést a felhasználó. A tanúsítványokat hitelesítõ cégek, mint például a VeriSign, írják alá ezeket a tanúsítványokat és ezzel érvényesítik az egyes cégek vagy egyének megbízhatóságát. Ez ugyan pénzbe kerül, de használatuk egyáltalán nem is kötelezõ. Azonban az átlagosnál paranoidabb felhasználók számára megnyugvást jelenthet.

14.9.1. Tanúsítványok elõállítása

A tanúsítványok létrehozására a következõ parancs áll rendelkezésre:

```
# openssl req -new -nodes -out req.pem -keyout cert.pem
Generating a 1024 bit RSA private key
```

```
.................++++++
.....................................++++++
writing new private key to 'cert.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:országnév (kétbetűs kóddal)
State or Province Name (full name) [Some-State]:állam vagy tartomány teljes neve
Locality Name (eg, city) []:település neve
Organization Name (eg, company) [Internet Widgits Pty Ltd]:szervezet neve
Organizational Unit Name (eg, section) []:szervezeti egység neve
Common Name (eg, YOUR name) []:általános név (hálózati név!)
Email Address []:e-mail cím

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:VALAMILYEN JELSZÓ
An optional company name []:egy másik szervezet neve
```

Az adatok bekérésére elõtt megjelenõ figyelmeztetõ üzenet fordítása:

```
Itt a tanúsítvány igénylésével kapcsolatos információkat kell
megadnunk. Itt egy ún. "ismertetőnevet" (Distinguished
Name, DN) kell megadnunk. Ezen kívül van még néhány más mező is, de
ezeket akár üresen is hagyhatjuk. Néhány mezőnek van alapértelmezett
értéke, de ha oda egy pontot írunk, akkor kitöröljük.
```

A "Common Name" mezõnél ellenõrzési okokból egy hálózati nevet, tehát a szerverünk nevét kell megadnunk. Ha nem így járunk el, akkor lényegében egy használhatatlan tanúsítványt kapunk. További opciók is elérhetõek, mint például a lejárati idõ (expire time) megadása, a titkosítási algoritmus megváltoztatása stb. Ezek teljes listája megtalálható az openssl(1) man oldalon.

Az elõbbi parancs kiadása után két állománynak kell létrejönnie az aktuális könyvtárban. A tanúsítványkérést, vagyis az req.pem állományt kell eljuttatnunk a tanúsítványok hitelesítésével foglakozó szervhez, aki majd érvényesíti az imént megadott adatainkat. A második, cert.pem nevû állomány a tanúsítványhoz tartozó privát kulcs, amit semmilyen körülmények között sem szabad kiadnunk. Ha ez mások kezébe kerül, akkor el tudnak játszani bennünket (vagy a szerverünket).

Amikor a hitelesítõ szerv aláírása nem feltétlenül szükséges, akkor készíthetünk egy saját magunk által aláírt tanúsítványt is. Ehhez elõször is generálnunk kell egy RSA-kulcsot:

```
# openssl dsaparam -rand -genkey -out saját_RSA.kulcs 1024
```

Most pedig készítsünk el a hitelesítõ szerv kulcsát is:

```
# openssl gendsa -des3 -out hitelesítõ.kulcs saját_RSA.kulcs
```

Ezzel a kulccsal most gyártsunk le egy tanúsítványt:

```
# openssl req -new -x509 -days 365 -key hitelesítõ.kulcs -out új.tanúsítvány
```

Ekkor két új állomány keletkezik a könyvtárunkban: a hitelesítõ szerv aláírása, a hitelesítõ.kulcs és maga a tanúsítvány, az új.tanúsítvány állomány. Ezeket tegyük az /etc könyvtáron belül egy olyan könyvtárba, amelyet csak a `root` tud olvasni. A `chmod` paranccsal állítsunk be rá 0700-as kódú engedélyeket.

14.9.2. Példa a tanúsítványok használatára

Mire is jók ezek az állományok? Például kitûnõen alkalmazhatóak a Sendmail levelezõ szerverhez beérkezõ kapcsolatot titkosítására. Így lényegében felszámoljuk minden olyan felhasználó titkosítatlan módon zajló hitelesítését, aki a helyi levelezõ szerveren keresztül küldi a leveleit.

Ez általában nem a legjobb megoldás, mivel egyes levelezõ kliensek hibát jeleneznek a felhasználónak, ha nem rendelkezik a tanúsítvánnyal. A tanúsítványok telepítésével kapcsolatban olvassuk el a szoftverhez adott leírást.

A helyi .mc állományba ezeket a sorokat kell beletenni:

```
dnl SSL Options
define(`confCACERT_PATH',`/etc/certs')dnl
define(`confCACERT',`/etc/certs/új.tanúsítvány')dnl
define(`confSERVER_CERT',`/etc/certs/új.tanúsítvány')dnl
define(`confSERVER_KEY',`/etc/certs/hitelesítõ.kulcs')dnl
define(`confTLS_SRV_OPTIONS', `V')dnl
```

Itt a /etc/certs/ az a könyvtár, amit tanúsítványok és kulcsok helyi tárolására használunk. Végezetül még újra kell generálnunk a helyi .cf állományokat. Ezt a /etc/mail könyvtárban a `make install` parancs kiadásával könnyen elvégezhetjük. Miután ez megtörtént, akkor Sendmailhoz tartozó démont a `make restart` paraméterével indíthatjuk újra.

Ha minden jól ment, akkor a /var/log/maillog állományban nem találunk egyetlen hibaüzenetet sem, és a Sendmail is megjelenik a futó programok között.

A telnet(1) segédprogrammal így probálhatjuk ki a levelezõ szervert:

```
# telnet example.com 25
Trying 192.0.34.166...
Connected to example.com.
Escape character is '^]'.
```

```
220 example.com ESMTP Sendmail 8.12.10/8.12.10; Tue, 31 Aug 2004 03:41:22 -0400 (EDT)
ehlo example.com
250-example.com Hello example.com [192.0.34.166], pleased to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-AUTH LOGIN PLAIN
250-STARTTLS
250-DELIVERBY
250 HELP
quit
221 2.0.0 example.com closing connection
Connection closed by foreign host.
```

Ha itt megjelenik a "STARTTLS" sor, akkor mindent sikerült beállítanunk.

14.10. VPN IPsec felett

VPN létrehozása FreeBSD átjárók használatával két olyan hálózat között, amelyeket egymástól az internet választ el.

14.10.1. Az IPsec bemutatása

Ebben a szakaszban az IPsec beállításának folyamatát vázoljuk fel. Az IPsec beállításához elengedhetetlen, hogy tisztában legyünk egy saját rendszermag fordításának alapjaival (lásd A FreeBSD rendszermag testreszabása).

Az *IPsec* egy olyan protokoll, amely az Internet Protocol (IP) rétegére épül. Segítségével két vagy több számítógép képes biztonságos módon tartani egymással a kapcsolatot (innen ered a neve). A FreeBSD IPsec "hálózati protokollkészlete" a KAME implementációjára épül, mely egyaránt támogatja az IPv4 és IPv6 protokollcsaládokat.

Az IPsec két alprotokollból tevõdik össze:

- *A hasznos adat biztonságos becsomagolása (Encapsulated Security Payload, ESP)* során egy szimmetrikus kriptográfiai algoritmussal (mint például Blowfish, 3DES) titkosítjuk az IP-csomagok tartalmát, ezáltal megvédjük ezeket az illetéktelenektõl.
- A *Hitelesítési fejléc (Authentication Header, AH)* használatával megakadályozzuk, hogy az illetéktelenek meghamisítsák az IP csomagok fejlécét. Ezt úgy érjük el, hogy kiszámolunk egy kriptográfiai ellenõrzõ összeget és az IP-csomagok fejlécének mezõire egy biztonságos függvénnyel generálunk valamilyen ujjlenyomatot. Az ez után következõ kiegészítõ fejléc tartalmazza ezt az ujjlenyomatot, amellyel a csomag hitelesíthetõ.

Az ESP és az AH az alkalmazástól függõen használható együtt vagy külön-külön.

Az IPsec akár közvetlenül is használható két számítógép forgalmának titkosítására (ezt *Szállítási*

módnak (Transport Mode) nevezik), vagy két alhálózat között építhetünk ki vele "virtuális tunneleket", ami remekül alkalmas két vállalati hálózat kommunikációjának bebiztosítására (ez a *Tunnel mód (Tunnel Mode)*). Ez utóbbit egyszerûen csak *Virtuális magánhálózatként (Virtual Private Network, VPN)* emlegetik. A FreeBSD IPsec alrendszerérõl az ipsec(4) man oldalon találhatunk további információkat.

A rendszermag IPsec támogatásának aktiválásához a következõ paramétereket kell beletennünk a konfigurációs állományba:

```
options   IPSEC        # IP biztonság
device    crypto
```

Ha szükségünk van a IPsec nyomkövetésére, a következõ beállítást is hozzátehetjük:

```
options   IPSEC_DEBUG  # az IP biztonság nyomkövetése
```

14.10.2. A probléma

Semmilyen szabvány nem fogalmazza meg mi is számít VPN-nek. A virtuális magánhálózatok tucatnyi különbözõ technológiával valósíthatóak meg, de mindegyiknek megvan a maga erõssége és gyengesége. Ebben a szakaszban körvonalazunk egy ilyen helyzetet, valamint a benne felépített VPN megvalósításához alkalmazott stratégiákat.

14.10.3. A forgatókönyv: adott egy otthoni és egy vállalati hálózat, amelyek külön-külön csatlakoznak az internetre, és VPN használatával ezeket egyetlen hálózatként szeretnénk használni

Elõfeltételezéseink a következõek:

- legalább két hálózatunk van;
- magán belül mind a két hálózat IP-t használ;
- mind a két hálózat egy FreeBSD átjárón keresztül csatlakozik az internethez;
- a hálózatok átjárói legalább egy publikus IP-címmel rendelkeznek;
- a hálózatok belsõ címei lehetnek publikus vagy privát IP-címek, nem számít. Fontos viszont, hogy ezek ne ütközzenek, vagyis ne használja egyszerre mind a kettõ a `192.168.1.x` címtartományt.

14.10.4. Az IPsec beállítása FreeBSD alatt

Kezdésképpen a Portgyûjteményböl telepítenünk kell a security/ipsec-tools portot. Ez a programcsomag rengeteg olyan alkalmazást tartalmaz, amely segítségünkre lehet a beállítások elvégzése során.

A következõ lépésben létre kell hoznunk két gif(4) típusú pszeudoeszközt, melyeken keresztül a két hálózat között egy tunnel segítségével ki tudjuk építeni a szükséges kapcsolatot. Ehhez `root`

felhasználóként futtassuk a következõ parancsokat (a *belsõ* és *külsõ* megnevezésû paramétereket cseréljük ki a valós belsõ és külsõ átjárók címeire):

```
# ifconfig gif0 create
```

```
# ifconfig gif0 belsõ1 belsõ2
```

```
# ifconfig gif0 tunnel külsõ1 külsõ2
```

Tekintsük például, hogy a vállalati LAN publikus IP-címe `172.16.5.4`, valamint a privát IP-címe `10.246.38.1`. Az otthoni LAN publikus IP-címe legyen most `192.168.1.12`, valamint a belsõ privát IP-címe pedig `10.0.0.5`.

Elsõre ez talán még nem teljesen érthetõ, ezért az ifconfig(8) parancs használatával is nézzük meg a példában szereplõ hálózatok konfigurációját:

```
Az elsõ átjáró:

gif0: flags=8051 mtu 1280
tunnel inet 172.16.5.4 --> 192.168.1.12
inet6 fe80::2e0::81ff:fe02:5881%gif0 prefixlen 64 scopeid 0x6
inet 10.246.38.1 --> 10.0.0.5 netmask 0xffffff00

A második átjáró:

gif0: flags=8051 mtu 1280
tunnel inet 192.168.1.12 --> 172.16.5.4
inet 10.0.0.5 --> 10.246.38.1 netmask 0xffffff00
inet6 fe80::250:bfff:fe3a:c1f%gif0 prefixlen 64 scopeid 0x4
```

Miután elvégeztük az iménti beállításokat, a ping(8) paranccsal már mind a két privát IP-tartománynak elérhetõnek kell lennie, ahogy azt az alábbi példa is érzékeltetni kívánja:

```
otthoni-halo# ping 10.0.0.5
PING 10.0.0.5 (10.0.0.5): 56 data bytes
64 bytes from 10.0.0.5: icmp_seq=0 ttl=64 time=42.786 ms
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=19.255 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=20.440 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=21.036 ms
--- 10.0.0.5 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 19.255/25.879/42.786/9.782 ms

vallalati-halo# ping 10.246.38.1
PING 10.246.38.1 (10.246.38.1): 56 data bytes
```

```
64 bytes from 10.246.38.1: icmp_seq=0 ttl=64 time=28.106 ms
64 bytes from 10.246.38.1: icmp_seq=1 ttl=64 time=42.917 ms
64 bytes from 10.246.38.1: icmp_seq=2 ttl=64 time=127.525 ms
64 bytes from 10.246.38.1: icmp_seq=3 ttl=64 time=119.896 ms
64 bytes from 10.246.38.1: icmp_seq=4 ttl=64 time=154.524 ms
--- 10.246.38.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 28.106/94.594/154.524/49.814 ms
```

Az elvárásainknak megfelelõen tehát a privát címeken mind a két oldalnak képesnek kell lennie ICMP csomagokat küldenie és fogadnia. A következõ lépésben meg kell mondanunk az átjáróknak hogyan irányítsák a csomagokat a két hálózat közti forgalom megfelelõ áramlásához. Ezt az alábbi paranccsal elérhetjük el:

```
# vallalati-halo# route add 10.0.0.0 10.0.0.5 255.255.255.0
```

```
# vallalati-halo# route add net 10.0.0.0: gateway 10.0.0.5
```

```
# otthoni-halo# route add 10.246.38.0 10.246.38.1 255.255.255.0
```

```
# otthoni-halo# route add host 10.246.38.0: gateway 10.246.38.1
```

Itt már a belsõ gépeket az átjárókról és az átjárók mögül egyaránt el tudjuk érni. A következõ példa alapján errõl könnyedén meg is tudunk gyõzõdni:

```
vallalati-halo# ping 10.0.0.8
PING 10.0.0.8 (10.0.0.8): 56 data bytes
64 bytes from 10.0.0.8: icmp_seq=0 ttl=63 time=92.391 ms
64 bytes from 10.0.0.8: icmp_seq=1 ttl=63 time=21.870 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=63 time=198.022 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=63 time=22.241 ms
64 bytes from 10.0.0.8: icmp_seq=4 ttl=63 time=174.705 ms
--- 10.0.0.8 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 21.870/101.846/198.022/74.001 ms

otthoni-halo# ping 10.246.38.107
PING 10.246.38.1 (10.246.38.107): 56 data bytes
64 bytes from 10.246.38.107: icmp_seq=0 ttl=64 time=53.491 ms
64 bytes from 10.246.38.107: icmp_seq=1 ttl=64 time=23.395 ms
64 bytes from 10.246.38.107: icmp_seq=2 ttl=64 time=23.865 ms
64 bytes from 10.246.38.107: icmp_seq=3 ttl=64 time=21.145 ms
64 bytes from 10.246.38.107: icmp_seq=4 ttl=64 time=36.708 ms
--- 10.246.38.107 ping statistics ---
```

```
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 21.145/31.721/53.491/12.179 ms
```

A tunnelek beállítása volt igazából a könnyebb rész, egy biztonságos összeköttetés kialakítása azonban már valamivel komolyabb folyamatot rejt magában. A most következõ konfigurációban erre "elõre ismert" (vagyis pre-shared, PSK) RSA-kulcsokat fogunk használni. A konkrét IP-címektõl eltekintve az átjárókon a /usr/local/etc/racoon/racoon.conf állományok hasonlóan fognak kinézni, nagyjából valahogy így:

```
path    pre_shared_key "/usr/local/etc/racoon/psk.txt"; # az ismert kulcsot tartalmazó
állomány helye
log     debug;  # a naplózás részletességének beállítása: ha végeztünk a teszteléssel
és a hibakereséssel, akkor állítsuk át a 'notify' értékre

padding  # ezeket ne nagyon változtassuk meg
{
        maximum_length  20;
        randomize       off;
        strict_check    off;
        exclusive_tail  off;
}

timer   # idõzítési beállítások, állítsuk be igény szerint
{
        counter         5;
        interval        20 sec;
        persend         1;
#       natt_keepalive  15 sec;
        phase1          30 sec;
        phase2          15 sec;
}

listen  # cím [port], ahol a racoon majd válaszolni fog
{
        isakmp          172.16.5.4 [500];
        isakmp_natt     172.16.5.4 [4500];
}

remote  192.168.1.12 [500]
{
        exchange_mode   main,aggressive;
        doi             ipsec_doi;
        situation       identity_only;
        my_identifier   address 172.16.5.4;
        peers_identifier        address 192.168.1.12;
        lifetime        time 8 hour;
        passive         off;
        proposal_check  obey;
#       nat_traversal   off;
        generate_policy off;
```

```
                        proposal {
                                encryption_algorithm    blowfish;
                                hash_algorithm          md5;
                                authentication_method   pre_shared_key;
                                lifetime time           30 sec;
                                dh_group                1;
                        }
}

sainfo  (address 10.246.38.0/24 any address 10.0.0.0/24 any) # address
$hálózat/$hálózati_maszk $típus address $hálózat/$hálózati_maszk $típus
          # (a $típus lehet "any" vagy "esp")
{         # a $hálózat a két összekapcsolni kívánt belsõ hálózat legyen
        pfs_group       1;
        lifetime        time    36000 sec;
        encryption_algorithm    blowfish,3des,des;
        authentication_algorithm        hmac_md5,hmac_sha1;
        compression_algorithm   deflate;
}
```

A példában szereplõ összes opció részletes kifejtése jóval meghaladná ezen leírás kereteit, ezért a bõvebb információkkal kapcsolatban inkább a racoon beállításaihoz tartozó man oldal elolvasását javasoljuk.

A gépek közti hálózati forgalom titkosításához be kell még állítanunk egy SPD házirendet is, így a FreeBSD és a racoon képes kódolni és dekódolni a csomagokat.

Ezt a most következõ, a vállalati átjárón találhatóhoz hasonló egyszerû shell szkripttel tudjuk elvégezni. Ezt az állományt a rendszer indításakor fogjuk felhasználni, melyet /usr/local/etc/racoon/setkey.conf néven mentsünk el:

```
flush;
spdflush;
# Az otthoni hálózati felé
spdadd 10.246.38.0/24 10.0.0.0/24 any -P out ipsec esp/tunnel/172.16.5.4-
192.168.1.12/use;
spdadd 10.0.0.0/24 10.246.38.0/24 any -P in ipsec esp/tunnel/192.168.1.12-
172.16.5.4/use;
```

Ahogy ezzel megvagyunk, a racoon az egyes átjárókon a következõ paranccsal indítható el:

```
# /usr/local/sbin/racoon -F -f /usr/local/etc/racoon/racoon.conf -l
/var/log/racoon.log
```

A parancs eredménye ennek megfelelõen nagyjából a következõ lesz:

```
vallalati-halo# /usr/local/sbin/racoon -F -f /usr/local/etc/racoon/racoon.conf
Foreground mode.
2006-01-30 01:35:47: INFO: begin Identity Protection mode.
2006-01-30 01:35:48: INFO: received Vendor ID: KAME/racoon
2006-01-30 01:35:55: INFO: received Vendor ID: KAME/racoon
2006-01-30 01:36:04: INFO: ISAKMP-SA established 72.16.5.4[500]-192.168.1.12[500]
spi:623b9b3bd2492452:7deab82d54ff704a
2006-01-30 01:36:05: INFO: initiate new phase 2 negotiation:
72.16.5.4[0]192.168.1.12[0]
2006-01-30 01:36:09: INFO: IPsec-SA established: ESP/Tunnel 92.168.1.12[0]-
>172.16.5.4[0] spi=28496098(0x1b2d0e2)
2006-01-30 01:36:09: INFO: IPsec-SA established: ESP/Tunnel 172.16.5.4[0]-
>192.168.1.12[0] spi=47784998(0x2d92426)
2006-01-30 01:36:13: INFO: respond new phase 2 negotiation:
172.16.5.4[0]192.168.1.12[0]
2006-01-30 01:36:18: INFO: IPsec-SA established: ESP/Tunnel 192.168.1.12[0]-
>172.16.5.4[0] spi=124397467(0x76a279b)
2006-01-30 01:36:18: INFO: IPsec-SA established: ESP/Tunnel 172.16.5.4[0]-
>192.168.1.12[0] spi=175852902(0xa7b4d66)
```

A tunnel megfelelõ mûködését úgy tudjuk ellenõrizni, ha átváltunk egy másik konzolra és a tcpdump(1) program segítségével figyeljük a hálózati forgalmat. A példában szereplõ `em0` interfészt természetesen ne felejtsük el kicserélni a megfelelõ eszköz nevére.

```
# tcpdump -i em0 host 172.16.5.4 and dst 192.168.1.12
```

Ennek hatására az alábbiakhoz hasonló adatoknak kellene megjelennie a konzolon. Amennyiben nem ez történik, valamilyen hiba történt, ezért meg kell keresnünk azt a visszakapott adatok alapján.

```
01:47:32.021683 IP vallalatihalozat.com > 192.168.1.12.otthonihalozat.com:
ESP(spi=0x02acbf9f,seq=0xa)
01:47:33.022442 IP vallalatihalozat.com > 192.168.1.12.otthonihalozat.com:
ESP(spi=0x02acbf9f,seq=0xb)
01:47:34.024218 IP vallalatihalozat.com > 192.168.1.12.otthonihalozat.com:
ESP(spi=0x02acbf9f,seq=0xc)
```

Itt már mind a két hálózatnak elérhetõnek kell lennie és egyként kell látszódnia. A hálózatokat ezen felül még érdemes külön védeni egy tûzfallal is. Ilyenkor a csomagok két hálózati közti zavartalan oda-vissza vándorlásához további szabályokat kell még felvennünk a tûzfal szabályrendszerébe. A ipfw(8) tûzfal esetén ez a következõ sorok hozzáadását jelenti a tûzfal konfigurációs állományához:

```
ipfw add 00201 allow log esp from any to any
ipfw add 00202 allow log ah from any to any
ipfw add 00203 allow log ipencap from any to any
```

```
ipfw add 00204 allow log udp from any 500 to any
```

A szabályok számozását mindig az adott gép aktuális beállításainak megfelelõen kell módosítani.

A pf(4) és ipf(8) felhasználók számára ehhez a következõ parancsot javasoljuk:

```
pass in quick proto esp from any to any
pass in quick proto ah from any to any
pass in quick proto ipencap from any to any
pass in quick proto udp from any port = 500 to any port = 500
pass in quick on gif0 from any to any
pass out quick proto esp from any to any
pass out quick proto ah from any to any
pass out quick proto ipencap from any to any
pass out quick proto udp from any port = 500 to any port = 500
pass out quick on gif0 from any to any
```

Végezetül a következõ sor hozzáadásával engedélyezzük az /etc/rc.conf állományban a VPN indítását a rendszer indítása során:

```
ipsec_enable="YES"
ipsec_program="/usr/local/sbin/setkey"
ipsec_file="/usr/local/etc/racoon/setkey.conf" # engedélyezzük az spd házirend
beállítását a rendszer indításakor
racoon_enable="yes"
```

14.11. OpenSSH

Az OpenSSH olyan hálózati kapcsolódási eszközök összessége, amivel biztonságos módon érhetünk el távoli számítógépeket. Az `rlogin`, `rsh`, `rcp` és a `telnet` direkt kiváltására használható. Emellett SSH-n keresztül TCP/IP kapcsolatok is biztonságosan bújtathatóak vagy küldhetõek tovább.

Az OpenSSH-t az OpenBSD projekt tartja karban, és az SSH 1.2.12 verziójára épül hibajavításokkal és frissítésekkel egyetemben. Az SSH 1 és 2 protokollokkal egyaránt kompatibilis.

14.11.1. Az OpenSSH használatának elõnyei

A hétköznapi esetben, vagyis amikor a telnet(1) vagy rlogin(1) alkalmazásokat használjuk, az adatok titkosítatlan formában közlekednek a hálózaton. A szerver és a kliens közé bárhova becsatlakozó hálózati kíváncsiskodók így könnyedén el tudják lopni a felhasználói nevünket és jelszavunkat, vagy lényegében bármilyen adatot, ami az adott munkamenetben megfordul. Az OpenSSH ennek kivédésére kínál fel különféle hitelesítési és titkosítási eszközöket.

14.11.2. Az sshd engedélyezése

Az sshd a FreeBSD telepítésekor jelentkezõ `Standard` lehetõségek egyike. Az sshd engedélyezését úgy tudjuk kideríteni, ha az rc.conf állományban megkeressük a következõ sort:

```
sshd_enable="YES"
```

Ez tölti be a rendszer indításakor az sshd(8)-t, az OpenSSH démonát. Vagy az /etc/rc.d/sshd rc(8) szkript segítségével is elindíthatjuk az OpenSSH-t:

```
/etc/rc.d/sshd start
```

14.11.3. Az SSH kliens

Az ssh(1) segédprogram az rlogin(1) programhoz hasonlóan mûködik.

```
# ssh felhasználó@gép.hu
Host key not found from the list of known hosts.  Are you sure you
want to continue connecting (yes/no)?  yes Host
'gép.hu' added to the list of known hosts.
felhasználó@gép.hu's password:
*******
```

Az üzenetek fordítása:

```
Nem találtam meg a gépet az ismert gépek között.  Biztosan csatlakozni
akarunk hozzá (igen/nem)?  igen A 'gép.hu'
felkerült az ismert gépek közé.
Adja meg a felhasználó@gép.hu jelszavát:
```

Bejelentkezés után minden ugyanolyan, mintha az `rlogin` vagy a `telnet` programokat használtuk volna. Az SSH egy kulcs segítségével próbálja azonosítani a számítógépeket, ezzel ellenõrzi a szerver hitelességét a kliensek csatlakozásakor. A felhasználónak ilyenkor elõször mindig `yes` választ kell adnia. A késõbbi bejelentkezési kísérletek pedig majd mindig az így kapott kulccsal történnek. Ha eltérne a kulcs, akkor az SSH kliens erre figyelmeztetni fog minket. A kulcsok a ~/.ssh/known_hosts vagy az SSH v2 protokoll esetén a ~/.ssh/known_hosts2 állományba kerülnek elmentésre.

Alapértelmezés szerint az OpenSSH szerverek csak SSH v2 kapcsolatokat fogadnak el. Lehetõség szerint a kliens is ezt a változatot fogja használni, de ha nem sikerül, akkor megpróbálkozik a v1-el. A kliensnek a `-1` vagy `-2` opciók segítségével elõ is lehet írni, hogy az elsõ vagy a második változatot használja. A kliensben az elsõ változat támogatását csupán a régebbi verziók kompatibilitása miatt tartják karban.

14.11.4. Biztonságos másolás

Az scp(1) parancs az rcp(1) parancshoz hasonlóan mûködik: egyik géprõl másol a másikra, biztonságosan.

```
#  scp felhasználó@gép.hu:/COPYRIGHT COPYRIGHT
felhasználó@gép.hu's password: *******
COPYRIGHT            100% |*****************************|  4735
00:00
#
```

Mivel a kulcsot már ismerjük ehhez a távoli géphez (az elõbbi példából), ezért az scp(1) használatakor már ezzel hitelesítünk.

Az scp(1) paraméterei hasonlóak a cp(1) parancséhoz: elsõ helyen az állomány vagy állományok neveit adjuk meg, a másodikon pedig a célt. Mivel az állományokat a hálózaton SSH-n keresztül küldik át, ezért az állományok neveit `felhasználó@gép:_elérési_út_` formában kell megadni.

14.11.5. Beállítások

Az OpenSSH démon és kliens rendszerszintû konfigurációs állományai az /etc/ssh könyvtárban találhatóak.

Az ssh_config tartalmazza a kliens beállításait, miközben az sshd_config tartalmazza a démonét.

Emellett az rc.conf állományban megadható `sshd_program` (ez alapból a /usr/sbin/sshd) és `sshd_flags` opciókkal további beállítási szinteket nyújtanak.

14.11.6. ssh-keygen

Jelszavak helyett az ssh-keygen(1) programmal a felhasználók azonosítására DSA- vagy RSA-kulcsokat tudunk készíteni:

```
% ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/felhasználó/.ssh/id_dsa):
Created directory '/home/felhasználó/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/felhasználó/.ssh/id_dsa.
Your public key has been saved in /home/felhasználó/.ssh/id_dsa.pub.
The key fingerprint is:
bb:48:db:f2:93:57:80:b6:aa:bc:f5:d5:ba:8f:79:17 felhasználó@gép.hu
```

Az ssh-keygen(1) ekkor a hitelesítésre létrehoz egy publikus és egy privát kulcsból álló párt. A privát kulcs a ~/.ssh/id_dsa vagy ~/.ssh/id_rsa állományba kerül, miközben a publikus kulcs a ~/.ssh/id_dsa.pub vagy ~/.ssh/id_rsa.pub lesz attól függõen, hogy DSA vagy RSA a kulcs típusa. A módszer mûködéséhez a publikus DSA- vagy RSA-kulcsot a távoli számítógép ~/.ssh/authorized_keys

állományába kell bemásolni.

Így tehát a távoli számítógépre jelszavak alkalmazása helyett SSH-kulccsal tudunk belépni.

Ha az ssh-keygen(1) parancsnak megadunk egy jelmondatot is, akkor a felhasználó a privát kulcsát csak ennek megadásával tudja használni. A hosszú jelmondatok állandó beirogatásától a Az ssh-agent és az ssh-add szakaszban hamarosan bemutatásra került ssh-agent(1) igyekszik megkímélni minket.

A különbözõ opciók és állományok eltérhetnek a számítógépünkre telepített OpenSSH verziójától függõen. Ilyen esetben javasolt felkeresni az ssh-keygen(1) man oldalát.

14.11.7. Az ssh-agent és az ssh-add

Az ssh-agent(1) és ssh-add(1) segédprogramokkal be tudjuk tölteni az SSH-kulcsokat a memóriába, amivel elkerülhetjük a jelmondat állandó begépelését.

A hitelesítést az ssh-agent(1) program kezeli a betöltött privát kulcsok alapján. Az ssh-agent(1) használatával egy másik programot is elindhatunk, egy parancsértelmezõtõl kezdve egy ablakkezelõig szinte bármit.

Az ssh-agent(1) programot úgy tudjuk egy parancsértelmezõben használni, hogy elõször is elindítjuk vele az adott parancsértelmezõt. Ezután az ssh-add(1) lefuttatásával hozzá kell adnunk egy identitást, annak jelmondatának megadásával. Miután ezeket megtettük, a felhasználó bármelyik olyan távoli gépre be tud jelentkezni, ahol a publikus kulcsát ismerik. Például:

```
% ssh-agent csh
% ssh-add
Enter passphrase for /home/felhasználó/.ssh/id_dsa:
Identity added: /home/felhasználó/.ssh/id_dsa (/home/felhasználó/.ssh/id_dsa)
%
```

Az ssh-agent(1) programot X11-el úgy tudjuk használni, ha az ~/.xinitrc állományba tesszük bele. Ezzel az ssh-agent(1) az összes X11-ben indított program számára rendelkezésre áll. Példának vegyük ezt az ~/.xinitrc állományt:

```
exec ssh-agent startxfce4
```

Így az X11 indulásakor mindig elindul az ssh-agent(1), amely pedig elindítja az XFCE alkalmazást. Miután átírtuk a saját állományunkat, a rendszer életbeléptetéséhez indítsuk újra az X11-et, az ssh-add(1) futtatásával pedig töltsük be az összes SSH-kulcsunkat.

14.11.8. Tunnelezés SSH-val

Az OpenSSH-val létre tudunk hozni egy tunnelt, amellyel egy másik protokoll adatait tudjuk titkosított módon becsomagolni.

Az alábbi parancs arra utasítja az ssh(1) programot, hogy hozzon létre egy tunnelt a telnet használatához:

```
% ssh -2 -N -f -L 5023:localhost:23 felhasználó@izé.mizé.hu
%
```

Az `ssh` parancsnak a következõ kapcsolókat adtuk meg:

`-2`
: Az `ssh` parancs a protokoll második változatát használja. (Ne adjuk meg, ha régi SSH szerverekkel dolgozunk.)

`-N`
: Tunnel létrehozása. Ha nem adjuk meg, akkor az `ssh` egy hagyományos munkamenet felépítését kezdi meg.

`-f`
: Az `ssh` a háttérben fusson.

`-L`
: Egy helyi tunnel a *helyiport:távoligép:távoliport* felírásban.

`felhasználó@izé.mizé.hu`
: A távoli SSH szerver.

Az SSH által létrehozott járatok úgy mûködnek, hogy létrehozunk egy csatlakozást a `localhost` (a helyi gép) megadott portján. Ezután minden olyan kapcsolatot, ami a helyi gép adott portjára érkezik, SSH-n keresztül átirányítunk a távoli gép portjára.

Ebben a példában a helyi gép *5023* portját átirányítjuk a helyi gép *23* portjára. Mivel a *23* a telnet portja, ezért az így definiált SSH járattal egy biztonságos telnet munkamenetet hozunk létre.

Ezen a módon tetszõleges nem biztonságos TCP protokollt, például SMTP-t, POP3-at, FTP-t stb. be tudunk csomagolni.

Példa 20. Biztonságos tunnel létrehozása SSH-val SMTP-hez

```
% ssh -2 -N -f -L 5025:localhost:25 felhasználó@levelező.szerver.hu
felhasználó@levelező.szerver.hu's password: *****
% telnet localhost 5025
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 levelező.szerver.hu ESMTP
```

Az ssh-keygen(1) és további felhasználói hozzáférések alkalmazásával ezen a módon ki tudunk alakítani egy minden további problémától és zûrtõl mentes SSH tunnelezési környezetet. A

jelszavak helyett kulcsokat használunk és minden tunnel külön felhasználóként is futtatható.

Gyakorlati példák a tunnelek használatára

Egy POP3 szerver biztonságos elérése

Tegyük fel, hogy a munkahelyünkön van egy SSH szerver, amire kívülrõl lehet csatlakozni, illetve vele egy hálózatban van egy POP3 levelezõ szerver is. A munkahelyünk és az otthonunk között levõ hálózati útvonalat részben vagy teljesen nem tartjuk megbízhatónak. Ezért az e-mailjeinket valamilyen biztonságos módon szeretnénk elérni. Ezt úgy tudjuk megvalósítani, ha otthonról csatlakozunk a munkahelyen levõ SSH szerverre és ezen keresztül érjük a levelezõ szervert.

```
% ssh -2 -N -f -L 2110:levél.gép.hu:110 felhasználó@ssh-szerver.gép.hu
felhasználó@ssh-szerver.gép.hu's password: ******
```

Miután a tunnel létrejött és mûködõképes, állítsuk be a levelezõ kliensünkben, hogy a POP3 kéréseket a `localhost` 2110 portjára küldje. Innen pedig biztonságos módon megy tovább a `levél.gép.hu` címre.

Egy szigorú tûzfal megkerülése

Egyes hálózati adminisztrátorok túlságosan szigorú szabályokat adnak meg a tûzfalban, és nem csak a bejövõ kapcsolatokat szûrik, hanem a kimenõket is. A távoli gépekhez csak a 22 (SSH) és 80 (böngészés) portjaikon tudunk csatlakozni.

Mi viszont szeretnénk más (nem egészen a munkánkkal kapcsolatos) szolgáltatásokat is elérni, például egy Ogg Vorbis szerverrõl zenét hallgatni. Ehhez a szerverhez viszont csak akkor tudnánk csatlakozni, ha a 22 vagy 80 portokon üzemelne.

Ezt a problémát úgy oldhatjuk meg, ha felépítünk egy SSH kapcsolatot a hálózatunk tûzfalán kívül levõ számítógéppel és segítségével átbújunk az Ogg Vorbis szerverhez.

```
% ssh -2 -N -f -L 8888:zene.gép.hu:8000 felhasználó@tűzfalazatlan-rendszer.gép.org
felhasználó@tűzfalazatlan-rendszer.gép.org's password: *******
```

A zenelejátszó kliensüknek adjuk meg a `localhost` 8888 portját, amely pedig a tûzfal sikeres kijátszásával továbbítódik a `zene.gép.hu` 8000-res portjára.

14.11.9. Az `AllowUsers` felhasználói beállítás

Gyakran nem árt korlátozni a felhasználók bejelentkezését. Az `AllowUsers` erre tökéletesen megfelel. Például, ha csak `192.168.1.32` címrõl engedjük bejelentkezni a `root` felhasználót, akkor ehhez valami ilyesmit kell beírnunk az /etc/ssh/sshd_config állományba:

```
AllowUsers root@192.168.1.32
```

Ezzel pedig csupán nevének megadásával engedélyezzük az `admin` felhasználó bejelentkezését (bárhonnan):

```
AllowUsers admin
```

Egy sorban több felhasználó is megadható, mint például:

```
AllowUsers root@192.168.1.32 admin
```

Ilyenkor ne felejtsük el megadni az összes bejelentkezésre (valamilyen formában) jogosult felhasználót megadni, máskülönben kizárjuk ezeket.

Miután elvégeztük a szükséges változtatásokat az /etc/ssh/sshd_config állományban, utasítsuk az sshd(8) démont a konfigurációs állományok újraolvasására:

```
# /etc/rc.d/sshd reload
```

14.11.10. Ajánlott olvasnivalók (angolul)

OpenSSH

ssh(1) scp(1) ssh-keygen(1) ssh-agent(1) ssh-add(1) ssh_config(5)

sshd(8) sftp-server(8) sshd_config(5)

14.12. Az állományrendszerek hozzáféréseit vezérlõ listák

A FreeBSD 5.0 és késõbbi változatai különbözõ fejlesztéseket hoztak az állományrendszerekben, például a pillanatképek készítése vagy a hozzáférés-vezérlési listák (Access Control List, ACL-ek) támogatása.

A hozzáférés-vezérlési listák a szabványos UNIX®-os engedély modellt bõvítik ki egy igen kompatibilis (POSIX®.1e) módon. Használatával a rendszergazdák egy sokkal kifinomultabb biztonsági modellt tudhatnak a kezük ügyében.

Az UFS állományrendszerek ACL támogatását úgy tudjuk engedélyezni, ha a rendszermagot az

```
options UFS_ACL
```

paraméterrel fordítjuk le. Amennyiben ezt nem fordítottuk bele, akkor az ACL támogatással rendelkezõ állományrendszerek csatlakoztatása során egy figyelmeztetést kapunk. Ez az opció a GENERIC rendszermag része. Az ACL az állományrendszeren engedélyezett kiterjesztett tulajdonságokra támaszkodik. Ezeket a kiterjesztett tulajdonságokat a következõ generációs UNIX®

állományrendszer, az UFS2 már alapból ismeri.

UFS1 típusú állományrendszereken sokkal nagyobb a kiterjesztett tulajdonságok kezelésének költsége, mint az UFS2 esetében. Az UFS2 jóval nagyobb teljesítménnyel képes dolgozni a kiterjesztett tulajdonságokkal. Emiatt a hozzáférés-vezérlési listák használatához az UFS2 sokkal inkább ajánlott, mint az UFS1.

Az ACL használatát a csatlakoztatáskor megadott `acls` beállítással engedélyezhetjük, amelyet érdemes felvennünk az /etc/fstab állományba. Ha a tunefs(8) segédprogrammal az állományrendszer fejlécében levõ szuperblokk ACL kapcsolóját átírjuk, akkor ez a beállítás automatikussá tehetõ. A szuperblokk használata több okból is ajánlatos:

- A csatlakoztatáskor megadott ACL beállítás nem változtatható egy egyszerû újracsatlakoztatással (mount(8) `-u`), csak egy teljes leválasztással (umount(8)) és egy friss csatlakoztatással (mount(8)). Ennek értelmében az ACL-ek a rendszerindító állományrendszeren a rendszer indulása után nem engedélyezhetõek. Ám ez azt is jelenti, hogy egy már használatban levõ állományrendszer beállításai sem változtathatóak meg.
- Ha a kapcsolót a szuperblokkban állítjuk be, akkor az állományrendszert még akkor is ACL támogatással csatlakoztatja a rendszer, ha azt nem adtuk meg az fstab állományban vagy az eszközeink átrendezõdtek. Így az állományrendszereket még véletlenül sem tudjuk ACL használata nélkül csatlakoztatni, ami egyébként így komoly biztonsági problémákat okozhatna.

Beállíthatjuk úgy is ACL kezelését, hogy egy friss csatlakoztatás nélkül is bekapcsolható legyen, azonban az ilyen állományrendszerek ACL nélküli csatlakoztatását nem ajánljuk senkinek, mivel ha egyszer már engedélyeztük a használatukat, majd kikapcsoljuk ezeket és végül a kiterjesztett tulajdonságok törlése nélkül újra engedélyezzük, akkor nagyon könnyen pórul járhatunk. Ha elkezdtük használni az ACL-eket egy állományrendszeren, akkor ne tiltsuk le ezeket, mert az így keletkezõ állományvédelem nem feltétlenül lesz kompatibilis a felhasználók által beállítottakkal, és az ACL újraengedélyezése a változásaik elõtti korábbi ACL engedélyeket fogja visszaállítani az állományokra, aminek hatása kiszámíthatatlan.

A hozzáférés-vezérlési listákat használó állományrendszerek esetén egy `+` (plusz) jellel ábrázolják a kiterjesztett engedélyeket. Például:

```
drwx------  2 robert  robert  512 Dec 27 11:54 private
drwxrwx---+ 2 robert  robert  512 Dec 23 10:57 könyvtár1
drwxrwx---+ 2 robert  robert  512 Dec 22 10:20 könyvtár2
drwxrwx---+ 2 robert  robert  512 Dec 27 11:57 könyvtár3
drwxr-xr-x  2 robert  robert  512 Nov 10 11:54 public_html
```

Láthatjuk, hogy a könyvtár1, könyvtár2 és könyvtár3 könyvtárakhoz tartoznak ACL típusú engedélyek, míg a public_html könyvtárhoz nem.

14.12.1. Az ACL-ek használata

Az állományrendszerben található ACL engedélyeket a getfacl(1) segédprogrammal nézhetjük meg. Például a próba állomány ACL engedélyeit a következõ paranccsal tudjuk megnézni:

```
% getfacl próba
    #file:próba
    #owner:1001
    #group:1001
    user::rw-
    group::r--
    other::r--
```

Egy állomány ACL engedélyeit a setfacl(1) segédprogrammal tudjuk megváltoztatni. Figyeljük meg:

```
% setfacl -k próba
```

A `-k` opció törli az összes ACL alapú engedélyt egy állományról vagy állományrendszerrõl. Ennél viszont sokkal hasznosabb a `-b` opció használata, mivel az meghagyja az ACL mûködéséhez szükséges alapvetõ mezõket.

```
% setfacl -m u:trhodes:rwx,group:web:r--,o::--- próba
```

Ebben a fenti parancsban a `-m` opciót pedig arra használtuk, hogy módosítsuk az alapértelmezett ACL bejegyzéseket. Mivel az ezt megelõzõ parancsban teljesen töröltük még az elõredefiniált bejegyzéseket is, ez a parancs a megadott paraméterekkel kiegészítve ezeket vissza fogja állítani. Ügyeljünk arra, hogy ha olyan felhasználót vagy csoportot adunk meg, ami nem létezik a rendszerben, akkor a szabvány kimenetre egy `Invalid argument` hibaüzenetet kapunk.

14.13. A külsõ programok biztonsági problémáinak figyelése

Az utóbbi években a biztonsági kérdésekkel foglalkozó világban számos fejlesztésre került sor a sebezhetõségi figyelmeztetések feldolgozásában. Manapság tulajdonképpen bármilyen operációs rendszer fokozott veszélynek teszik ki magát a külsõ programok telepítésével és használatával.

A sebezhetõségekrõl beszámoló értesítések a biztonság egyik alapköve, azonban a FreeBSD projekt nem tud ilyen jelentéseket kiadni a FreeBSD alaprendszerén kívül minden egyes külsõ alkalmazáshoz. Azonban lehetõségünk van enyhíteni a külsõ csomagok sebezhetõségén és figyelmeztetni a rendszergazdákat az ismert biztonsági problémákra. A FreeBSD-nek van egy Portaudit nevû segédprogramja, amit kizárólag erre a célra hoztak létre.

A ports-mgmt/portaudit port egy adatbázist használ, ahol a FreeBSD biztonsági csapata és a portok fejlesztõi tartják karban az ismert biztonsági problémákat.

A Portaudit használatának megkezdéséhez telepítsük a Portgyûjteményböl:

```
# cd /usr/ports/ports-mgmt/portaudit && make install clean
```

A telepítési folyamat során a periodic(8) konfigurációs állományai is frissítõdnek, így a Portaudit is lefut a napi biztonsági ellenõrzések folyamán. Gondoskodjunk róla, hogy a `root` felhasználónak levélben elküldött a napi biztonsági értesítéseket rendesen elolvassuk. Nincs szükségünk további beállításokra.

A telepítés után a rendszergazda a következõ paranccsal tudja frissíteni a saját adatbázispéldányát és megnézni a pillanatnyilag telepített csomagok ismert sebezhetõségeit:

```
# portaudit -Fda
```

Ez az adatbázis a periodic(8) minden egy futásakor magától frissül, ezért ez a parancs lényegében elhagyható. Egyedül a soronkövetkezõ példákhoz kell kiadni.

A Portgyûjteményböl telepített külsõ alkalmazások megbízhatóságának ellenõrzését az alábbi parancs kiadásával bármikor elvégezhetjük:

```
# portaudit -a
```

A Portaudit ennek hatására valahogy így fogja megjeleníteni a sebezhetõ csomagokat:

```
Affected package: cups-base-1.1.22.0_1
Type of problem: cups-base -- HPGL buffer overflow vulnerability.
Reference: <http://www.FreeBSD.org/ports/portaudit/40a3bca2-6809-11d9-a9e7-
0001020eed82.html>

1 problem(s) in your installed packages found.

You are advised to update or deinstall the affected package(s) immediately.
```

Fordítása:

```
Érintett csomag: cups-base-1.1.22.0_1
A probléma jellege: cups-base -- HPGL puffer túlcsordulási sebezhetõség.
Link: <http://www.FreeBSD.org/ports/portaudit/40a3bca2-6809-11d9-a9e7-
0001020eed82.html>

A telepített csomagokkal kapcsolatban 1 problémát találtam.

Javasoljuk, hogy az érintett csomagokat azonnal frissítse vagy távolítsa el.
```

Ha a böngészõnket az itt megadott címre irányítjuk, akkor megismerhetjük a kérdéses sebezhetõség pontosabb részleteit. Ezen az oldalon megtalálhatjuk a hiba által érintett verziókat a FreeBSD portok verziója szerint, illetve más olyan honlapokat, ahol biztonsági figyelmeztetéseket találhatunk.

Röviden összefoglalva, a Portaudit egy komoly segédeszköz és hitetlenül hasznos kiegészítõje a Portupgrade portnak.

14.14. A FreeBSD biztonsági figyelmeztetései

A FreeBSD több más kereskedelmi minõségû operációs rendszerhez hasonlóan "Biztonsági figyelmeztetéseket" (Security Advisory) ad ki. Ezek a figyelmeztetések általában megjelennek a biztonsággal foglalkozó levelezési listákon és a hivatkozott hibák kijavítása után a megfelelõ kiadások hibajegyzékében is. Ebben a szakaszban megismerjük és értelmezzük ezeket a figyelmeztetéseket, valamint megtudhatjuk, milyen lépéseket kell megtennünk a rendszerünk kijavításához.

14.14.1. Hogyan épül fel egy figyelmeztetés?

A FreeBSD biztonsági figyelmeztetései az alább látható formában jelennek meg, amit mi most a FreeBSD security notifications levelezési lista levelezési listáról kölcsönöztünk.

```
=============================================================================
FreeBSD-SA-XX:XX.UTIL                                          Security Advisory
                                                               The FreeBSD Project

Topic:          denial of service due to some problem①

Category:       core②
Module:         sys③
Announced:      2003-09-23④
Credits:        Person@EMAIL-ADDRESS⑤
Affects:        All releases of FreeBSD⑥
                FreeBSD 4-STABLE prior to the correction date
Corrected:      2003-09-23 16:42:59 UTC (RELENG_4, 4.9-PRERELEASE)
                2003-09-23 20:08:42 UTC (RELENG_5_1, 5.1-RELEASE-p6)
                2003-09-23 20:07:06 UTC (RELENG_5_0, 5.0-RELEASE-p15)
                2003-09-23 16:44:58 UTC (RELENG_4_8, 4.8-RELEASE-p8)
                2003-09-23 16:47:34 UTC (RELENG_4_7, 4.7-RELEASE-p18)
                2003-09-23 16:49:46 UTC (RELENG_4_6, 4.6-RELEASE-p21)
                2003-09-23 16:51:24 UTC (RELENG_4_5, 4.5-RELEASE-p33)
                2003-09-23 16:52:45 UTC (RELENG_4_4, 4.4-RELEASE-p43)
                2003-09-23 16:54:39 UTC (RELENG_4_3, 4.3-RELEASE-p39)⑦
CVE Name:   CVE-XXXX-XXXX⑧

For general information regarding FreeBSD Security Advisories,
including descriptions of the fields above, security branches, and the
following sections, please visit
http://www.FreeBSD.org/security/.
```

```
I.   Background⑨

II.  Problem Description⑩

III. Impact⑪

IV.  Workaround⑫

V.   Solution⑬

VI.  Correction details⑭

VII. References⑮
```

① A `Topic` mezõben olvashatjuk pontosan mi is maga a probléma. Alapvetõen bemutatja az érintett biztonsági figyelmeztetést és megemlíti a sebezhetõ segédprogramot.

② A `Category` mezõ hivatkozik a rendszer azon részére, amelyre a hiba kihatással lehet. Értéke lehet `core`, `contrib` vagy `ports`. A `core` kategória azt jelzi, hogy a sebezhetõség a FreeBSD legfontosabb komponenseit érinti. A `contrib` kategória a FreeBSD projekt számára felajánlott szoftverek, mint például a sendmail sebezhetõségére utal. Végezetül a `ports` kategória jelzi, hogy a sebezhetõség valamelyik, a Portgyûjteményben szereplõ szoftverre érvényes.

③ A `Module` mezõ a sebezhetõ komponens helyét nevezi meg, például `sys`. Ebben a példában azt láthatjuk, hogy a `sys` modul a hibás. Ezért a sebezhetõség egy rendszermagban használt komponenst érint.

④ Az `Announced` mezõ a biztonsági figyelmeztetés kiadásának vagy széleskörû kihirdetésének dátumát rögzíti. Ez azt jelenti, hogy a biztonsági csapat meggyõzõdött a probléma létezésérõl és a hibát orvosoló javítás már felkerült a FreeBSD forráskódjába.

⑤ A `Credits` mezõ azokat az egyéneket vagy szervezeteket említi meg, akik észlelték a sebezhetõséget és jelentették.

⑥ Az `Affects` mezõben megadják, hogy a FreeBSD melyik kiadásaira van hatással a sebezhetõség. Ha a rendszermag esetén lefuttatjuk az `ident` parancsot az érintett állományokra, akkor megtudhatjuk a pontos revíziójukat. A portoknál a verziószám a port neve után szerepel a /var/db/pkg könyvtárban. Ha a rendszerünket nem frissítettük CVS-rõl és fordítottuk újra, akkor nagy a valószínûsége, hogy a sebezhetõség minket is érint.

⑦ A `Corrected` mezõ tartalmazza a a kijavítás dátumát, idejét, idõzónáját és az ezt tartalmazó kiadást.

⑧ Az ismert sebezhetõségek adatbázisában (Common Vulnerabilities Database, CVD) használt azonosítási információk alapján végzett keresések számára fenntartott.

⑨ A `Background` mezõ adja meg részleteiben a sebezhetõ programmal kapcsolatos tudnivalókat. Az esetek többségében itt írják le, hogy miért jött létre az adott eszköz a FreeBSD-ben, mire használják és hogyan keletkezett.

⑩ A `Problem Description` mezõ a biztonsági rést részletezi. Ebben a részben szerepelhet a hibás kódrészlet vagy akár még az is, hogy miként kell vele elõidézni a hibát.

⑪ Az `Impact` mezõ a probléma lehetséges hatásait írja körül a rendszerben. Ez például lehet egy DoS támadás, speciális engedélyek ellopása vagy akár a rendszeradminisztrátori jogok megszerzése.

⑫ A `Workaround` mezõ igyekszik elfogadható megoldást nyújtani a rendszerük frissítésére képtelen rendszergazdák számára. Ennek oka lehet az idõ rövidsége, a hálózati elérhetõség vagy más okokból fakadó elcsúszás. Ennek ellenére a biztonsági kérdéseket sosem szabad félvállról venni, ezért a sebezhetõ rendszereket vagy ki kell javítani vagy valamilyen módon meg kell kerülni a biztonsági rés kialakulását.

⑬ A `Solution` mezõ utasításokkal segít a rendszer kijavítását. Ez egy lépésrõl lépésre tesztelt és ellenõrzött módszer, amellyel a rendszerünket megfelelõen ki tudjuk javítani és biztonságossá tenni.

⑭ A `Correction Details` mezõ mutatja a CVS-ág vagy kiadás nevét, amelyben a pontokat aláhúzásra cserélték. Ezenkívül még az egyes ágakban az érintett állományok revízióját is mutatja.

⑮ A `References` mezõ általában a témával kapcsolatos további forrásokat kínálja fel URL, könyv, levelezési lista vagy hírcsoport formájában.

14.15. A futó programok nyilvántartása

A futó programok nyilvántartása olyan biztonsági módszer, ahol a rendszergazda figyelemmel kíséri a rendszer használatban levõ erõforrásait, a felhasználók közti megoszlását, gondoskodik a rendszer felügyeletérõl és valamennyire nyomon követi a felhasználók parancsait.

Ennek a módszernek egyaránt megvannak a maga elõnyei és hátrányai. Az egyik elõnye, hogy a használatával a behatolás egészen a betörés pontjáig visszakövethetõ. Hátranya viszont, hogy a futó programok nyilvántartása rengeteg mennyiségû naplót generál és ehhez sok lemezterületre lesz szükségünk. Ebben a szakaszban végigjárjuk a programok nyilvántartásának alapjait.

14.15.1. A futó programok nyilvántartásának engedélyezése és használata

A futó programok nyilvántartását elõször engedélyeznünk kell. Ehhez a következõ parancsokat kell kiadnunk:

```
# touch /var/account/acct

# accton /var/account/acct

# echo 'accounting_enable="YES"' >> /etc/rc.conf
```

Miután aktiváltuk, a nyilvántartást elkezdi számbavenni a processzor kihasználtságát, a parancsokat stb. A nyilvántartás emberek számára nem olvasható formátumban készül, ezért csak az sa(8) segédprogrammal tudjuk megnézni. Ha nem adunk meg neki semmilyen opciót, akkor az `sa` kilistázza a felhasználónkénti hívásokat, az összes eltelt idõt percben, a teljes processzor- és felhasználói idõt percben, az I/O mûveletek átlagos számát stb.

A kiadott parancsokról a lastcomm(1) programmal tudunk tájékozódni. A `lastcomm` segítségével ki tudjuk íratni a felhasználók adott terminálon kiadott parancsait is, mint például:

```
# lastcomm ls
    trhodes ttyp1
```

Ezzel megjelenik a `trhodes` nevû felhasználó `ttyp1` terminálon kiadott összes ismert `ls` parancsa.

Számos hasznos beállítást és hozzájuk tartozó leírást találhatunk még a lastcomm(1), acct(5) és sa(8) man oldalakon.

Chapter 15. A jail alrendszer

15.1. Áttekintés

Ez a fejezet a FreeBSD-ben található jail alrendszert, valamint annak használatát mutatja be közelebbrõl. Az jail, melyet gyakran csak úgy emlegetnek, mint a *chroot környezetek* továbbfejlesztését, a rendszergazdák számára ajánlott, nagyon sokoldalú eszköz, de a haladó felhasználók is hasznosnak találhatják.

A fejezet elolvasása során megismerjük:

- mi is az a jail, milyen célra használható a FreeBSD-ben;
- hogyan hozzunk létre, indítsunk el és állítsunk le jaileket;
- a létrehozott jailek karbantartásainak alapjait, a jailek belülrõl és kívülrõl egyaránt.

A jail alrendszerrõl még több hasznos információt a következõ helyekrõl tudhatunk meg:

- A jail(8) man oldal. Ez tartalmazza a `jail` segédprogram teljes referenciáját - ez az a karbantartásra használható eszköz, amellyel el tudjuk indítani, le tudjuk állítani és vezérelni tudjuk a jaileket a FreeBSD-ben.
- A levelezési listák és azok archívumai. A FreeBSD general questions levelezési lista archívuma és a FreeBSD lista szerveren található többi levelezési lista rengeteg olvasnivalót tartogat a jailekkel kapcsolatban. Mindig érdemes keresni ezekben az archívumokban, vagy beküldeni a kérdésünket a freebsd-questions levelezési listára.

15.2. A jail alrendszerhez kapcsolódó fogalmak

A fejezet további részében a következõ fogalmakat fogjuk használni, hogy a FreeBSD jailekhez tartozó egyes részeit és azok belsõ mûködését, valamint kapcsolatukat a rendszer többi részével még inkább érthetõvé tegyük:

chroot(8) (parancs)

Egy segédprogram, amely a FreeBSD chroot(2) rendszerhívásán keresztül egy program és annak leszármazottjainak futtatásához megváltoztatja a rendszer gyökérkönyvtárát (change root).

chroot(2) (környezet)

A "chroot" módban futó programok környezete. Olyan erõforrásokat foglal magában, mint mondjuk az állományrendszer látható része, az elérhetõ felhasználói és csoport azonosítók, hálózati csatolók és egyéb folyamatok közti kommunikációs mechanizmusok stb.

jail(8) (parancs)

Az a rendszerkarbantartó segédprogram, amely lehetõvé teszi program elindítását elzárt környezetben.

befogadó (rendszer, program, felhasználó stb.)

Az elzárt környezetet irányító rendszer. A befogadó rendszer hozzá tud férni az összes elérhetõ

hardveres erõforráshoz, képes az elzárt környezeten kívül és belül futó programokat vezérelni. Az egyik legfontosabb különbség a befogadó és az elzárt rendszer között, hogy azok a korlátozások, amelyek az elzárt környezetben rendszeradminisztrátori jogokkal futó programokra vonatkoznak, nem feltétlenül érvényesek a befogadó rendszerben futóakra.

befogadott (rendszer, program, felhasználó stb.)

Olyan program, felhasználó vagy más egyéb egyed, amely csak egy jailen keresztül, korlátozottan tud hozzáférni az erõforrásokhoz.

15.3. Bevezetés

Mivel a rendszeradminisztráció egy nehéz és zavarba ejtõ feladat, rengeteg komoly eszköz jött létre a rendszergazdák életének megkönnyítésére. Ezek az eszközök többnyire a rendszerek telepítését, beállítását és karbantartását igyekeznek valamilyen módon jobbá tenni. A rendszergazdák egyik feladata úgy gondoskodni a biztonságról, hogy közben a rendszer képes legyen ellátni eredeti feladatát.

A FreeBSD rendszerek biztonságosságának növelését hivatott egyik ilyen eszköz a *jails*. Elõször a FreeBSD 4.X verziójában bukkant fel, de jelentõs fejlõdésen ment keresztül a FreeBSD 5.X verziókban, aminek köszönhetõen sokkal erõteljesebb és rugalmasabb alrendszerré vált. A fejlesztése természetesen most is folytatódik tovább, állandóan fejlõdik a használhatósága, teljesítménye, megbízhatósága és biztonságossága.

15.3.1. Mi is az a jail?

A BSD-szerû operációs rendszerekben már a 4.2BSD óta megtalálható volt a chroot(2). A chroot(8) segédprogrammal meg tudjuk megváltoztatni adott programok számára a gyökérkönyvtárat, és ezzel egy biztonságos környezetet teremteni, távol a rendszer többi részétõl. A chroot-tal kialakított környezetben elinduló programok nem tudnak hozzáférni a rajta kívül található állományokhoz és erõforrásokhoz. Ennek okán, ha egy ilyen környezetben futó szolgáltatást megtámadnak, az önmagában még nem teszi lehetõvé a támadó számára, hogy elérhesse az egész rendszert. A chroot(8) remekül használható olyan egyszerûbb feladatok megoldására, amelyek nem igényelnek túlságosan sok rugalmasságot vagy bonyolult és fejlett támogatást. A chroot ötletének felmerülése óta azonban számos kiskaput találtak már az általa létrehozott környezetekben, és habár ezek mindegyikét javították a FreeBSD újabb változataiban, teljesen egyértelmûvé vált, hogy a chroot(2) nem biztosít járható utat a szolgáltatások biztonságossá tételéhez. Erre a feladatra egy új alrendszert kellett kiépíteni.

Ez az egyik oka annak, amiért az *jaileket* kifejlesztették.

A jailek által képviselt elzárás ötlete több szempontból is a hagyományos chroot(2) környezet elvén alapszik. Egy hagyományos chroot(2) környezetben futó programok korlátozása csupán abban merül ki, hogy az állományrendszer melyik részét láthatják. A rendszer többi erõforrása (mint mondjuk a felhasználók, futó programok vagy a hálózati alrendszer) azonban továbbra is megosztva marad a chroot környezetben és a befogadó rendszerben futó programok között. A jailek által alkalmazott megoldás kibõvíti ezt a modellt, és nem csak az állományrendszerre vonatkozó hozzáférést virtualizálja, hanem több más dolog mellett kiterjeszti ezt a felhasználókra és a FreeBSD hálózati alrendszerére is. Az elzárt környezetek beállításaihoz elérhetõ finomhangolási

lehetõségekrõl bõvebben a Finomhangolás és karbantartásban esik szó.

A jaileket az alább négy elem írja le:

- A könyvtárszerkezet egy részfája - attól a résztõl indulva, ahonnan a jail kezdõdik. A jailen belül futó programok nem léphetnek ki ebbõl a részfából. Az eredeti chroot(2) kialakításában merengõ biztonsági hibák lehetõségei nem veszélyeztetik a többi FreeBSD jailt.
- A rendszer neve - a név, amelyet a jailen belül használunk. Mivel a jaileket elsõsorban hálózati szolgáltatások kordában tartására használjuk, a jailekhez tartozó beszédes rendszernevek sokat tudnak segíteni a rendszergazdák munkájában.
- Egy IP-cím - a jailhez tartozik és nem változtatható meg a mûködése során. Egy jail IP-címe általában egy már létezõ hálózati csatoló másik címe, de ez nem szükségszerûen igaz minden esetben.
- Egy parancs - annak a programnak az elérési útja, amelyet elzártan kívánunk futtatni. Az elzárt környezet gyökerétõl mérve relatívan adjuk meg, és az adott környezet típusától függõen eltérõ lehet.

Ezektõl eltekintve a jailek rendelkezhetnek saját felhasználókkal és lehetnek saját `root` felhasználóik is. Természetesen a `root` hatásköre csak az elzárt környezetre korlátozódik, és a befogadó rendszer szemszögébõl az elzárt `root` nem mindenható. Ráadásul az elzárt `root` felhasználó nem hajthat végre semmilyen kritikus mûveletet a saját jail(8) környezetén kívül. A `root` további képességeirõl és korlátozásairól lentiekben bõvebben is említést teszünk a Finomhangolás és karbantartásban.

15.4. A jailek létrehozása és vezérlése

Egyes rendszergazdák a jaileket a következõ két típusba sorolják: "teljes" jail, mely egy valódi FreeBSD rendszerre emlékeztet, és a "szolgáltatás" jail, mely egyetlen, feltehetõen kiemelt jogokkal futó alkalmazás vagy szolgáltatás számára van elõkészítve. Ez a besorolás csupán fogalmi szintû, a jail felépítésének módját nem befolyásolja. A jail(8) man oldal részletesen ismerteti a jailek létrehozását:

```
# setenv D /itt/lesz/a/jail
# mkdir -p $D ①
# cd /usr/src
# make buildworld ②
# make installworld DESTDIR=$D ③
# make distribution DESTDIR=$D ④
# mount -t devfs devfs $D/dev ⑤
```

① Érdemes elõször a jail helyét megválasztani. Itt fog fizikailag helyet foglalni a befogadó rendszer állományrendszerén belül a jail. Jó választás lehet erre a /usr/jail/jailnév, ahol a *jailnév* a jailt azonosító rendszernév. A /usr/ állományrendszeren általában elegendõ hely jut a jail állományrendszerének, ami egy "teljes" jail esetén lényegében a FreeBSD alaprendszer alapértelmezett telepítésében megtalálható összes állomány másolatát tartalmazza.

② Ha korábban már a `make world` vagy a `make buildworld` parancs segítségével újrafordítottuk az

alaprendszert, akkor ezt a lépést ki is hagyhatjuk és telepítsük az új alaprendszert közvetlenül az új jailbe.

③ Ez a parancs fogja felmásolni a jail fizikai helyének választott könyvtár-részfába a mûködéshez szükséges programokat, függvénykönyvtárakat, man oldalakat és így tovább.

④ A make paramétereként megadott `distribution` cél gondoskodik az összes szükséges konfigurációs állomány felmásolásáról. Magyarán szólva, átmásolja az összes telepíthetõ állományt a /usr/src/etc/ könyvtárból a jail /etc alkönyvtárába, vagyis a $D/etc/ könyvtárba.

⑤ A jaileken belül a devfs(8) csatlakoztatása nem kötelezõ. Másrészt azonban majdnem mindegyik alkalmazás, a feladatától függõen, legalább egy eszközhöz hozzá akar férni. Nagyon fontos, hogy a kezünkbe vegyük a eszközök hozzáférésének irányítását a jaileken belül, mivel a helytelen beállítások révén a támadók csúnya dolgokat tudnak majd mûvelni. A devfs(8) mûködését a devfs(8) és devfs.conf(5) man oldalakon is ismertetett szabályrendszerek irányítják.

Ahogy a jailt telepítettük, a jail(8) segédprogrammal tudjuk elindítani. A jail(8) négy kötelezõ paramétert vár, melyekre a Mi is az a jail?ban ki is térünk. Más paramétereket is megadhatunk, például azt, hogy az elzárt program egy adott felhasználó jogaival fusson. A `command` paraméter használata a jail típusától függ: egy *virtuális rendszer* esetében a /etc/rc jó választásnak bizonyulhat, mivel ennek segítségével egy valódi FreeBSD rendszerindítási folyamatát játszhatjuk le. Amennyiben elzárt *szolgáltatás*ról van szól, az adott szolgáltatástól vagy alkalmazástól függ.

A jaileket gyakran már a rendszerindítás során elindítják, amit a FreeBSD rc mechanizmusa nagyban meg is könnyít.

1. A rendszer indítása során aktiválandó jailek listáját vegyük hozzá a rc.conf(5) állományhoz:

```
jail_enable="YES"   # Ide NO-t írjunk, ha ki akarjuk kapcsolni
jail_list="www"     # Szóközzel elválasztva soroljuk fel a jaileket
```

A `jail_list` értékeként felsorolt jailek nevei csak betûket és számjegyeket tartalmazhatnak.

2. A `jail_list`-ben szereplõ összes jailt meg kell adnunk az ezeket leíró rc.conf(5)-beli beállításokat:

```
jail_www_rootdir="/usr/jail/www"     # a jail gyökérkönyvtára
jail_www_hostname="www.example.org"  # a jail neve
jail_www_ip="192.168.0.10"           # a jail IP-címe
jail_www_devfs_enable="YES"          # legyen-e devfs a jailen belül
jail_www_devfs_ruleset="www_ruleset" # az alkalmazott devfs szabályrendszer
```

Az rc.conf(5) állományban szereplõ jailek esetén a /etc/rc szkript fut le, tehát feltételezi, hogy az így megadott jail egy teljes virtuális rendszer. A szolgáltatások jailbe foglalásához meg kell változtatnunk a jail alapértelmezett parancsát is. Ezt a `jail`*jailnév*`exec_start` opció megfelelõ beállításával tudjuk megtenni.

Az összes itt elérhetõ opciót a rc.conf(5) man oldalon találhatjuk meg.

Ha léteznek a megfelelõ bejegyzések az rc.conf állományban, akkor az /etc/rc.d/jail szkript is használható arra, hogy a jaileket kézzel indítsuk el vagy állítsuk le:

```
# /etc/rc.d/jail start www
# /etc/rc.d/jail stop www
```

A jail(8) leállítására jelen pillanatban még nem érhetõ el szabályos módszer. Ez azért van, mert a szabályos rendszerleállítást elvégzõ parancsok nem használhatóak a jailen belül. Emiatt a jaileket a legtisztábban úgy tudjuk leállítani, ha kiadjuk az alábbi parancsot magában a jailben vagy pedig a jexec(8) segédprogrammal a jailen kívülrõl:

```
# sh /etc/rc.shutdown
```

Errõl a témáról többet a jail(8) man oldalon olvashatunk.

15.5. Finomhangolás és karbantartás

Számos opció állítható be a jaileknél, és sokféle módon vegyíthetjük a befogadó FreeBSD rendszerünket a jailekkel, ami által magasabb szintû alkalmazásokat hozhatunk létre. Ebben a részben bemutatunk:

- Néhány olyan beállítást, amellyel finomhangolhatjuk a telepített jailek által megvalósított biztonsági megszorítások viselkedését.
- A jailek kezelésére alkalmas néhány olyan magasabb szintû alkalmazást, amelyek elérhetõek a FreeBSD Portgyûjteményén keresztül, és általános jail alapú megoldások kialakításához használhatóak.

15.5.1. A FreeBSD-ben található finomhangoló eszközök

A jailek beállításainak finomhangolását túlnyomórészt sysctl(8) változókkal végezhetjük el. A sysctl-en belül egy speciális részfában találhatunk erre alkalmas beállításokat: ez a a FreeBSD rendszermag opciói között megtalálható `security.jail.*`. Itt közüljük a jailekre vonatkozó fontosabb sysctl változók listáját, az alapértelmezett értékeikkel együtt. A nevek minden bizonnyal sokat elárulnak, de ha többet szeretnénk tudni róluk, lapozzuk fel a jail(8) és sysctl(8) man oldalakat.

- `security.jail.set_hostname_allowed: 1`
- `security.jail.socket_unixiproute_only: 1`
- `security.jail.sysvipc_allowed: 0`
- `security.jail.enforce_statfs: 2`
- `security.jail.allow_raw_sockets: 0`

- `security.jail.chflags_allowed: 0`
- `security.jail.jailed: 0`

Ezekkel a változókkal a *befogadó rendszer* rendszergazdája tud hozzátenni vagy elvenni a `root` felhasználó alapértelmezett határaihoz. Vegyük azonban észre, hogy egyes korlátozások azonban semmiképpen sem szüntethetõek meg. A `root` nem csatlakoztathat és választhat le állományrendszereket a jail(8) környezetben. Az elzárt `root` nem tölthet be és törölhet devfs(8) szabályrendszereket, tûzfal szabályokat sem, ill. nem végezhet semmilyen olyan bármilyen más karbantartási feladatot, amely a rendszermag adataiban módosítást vonna maga után, például nem állíthatja a rendszermag `securelevel` (biztonsági szintjének) értékét.

A FreeBSD alaprendszere tartalmazza azokat a segédeszközöket, amelyekkel a rendszerben aktív jailek információt tudjuk megjeleníteni, vagy csatlakozni tudunk hozzájuk. A jls(8) és jexec(8) parancsok részei az alap FreeBSD rendszernek, segítségükkel elvégezhetõek az alábbi egyszerû feladatokat:

- Ki tudjuk íratni az aktív jailek és hozzájuk tartozó azonosítókat (JID-eket), IP-címeket, neveket és útvonalakat.
- A befogadó rendszerbõl hozzá tudunk csatlakozni egy futó jailhez, és parancsokat tudunk futtatni a jailen belül vagy karbantartási feladatokat tudunk elvégezni magán a jailen belül. Ez különösen hasznosnak bizonyulhat, amikor a `root` felhasználó szabályosan le akarja állítani a jailt. A jexec(8) segédprogrammal el tudunk indítani egy parancsértelmezõt a jailen belül, amibõl aztán irányíthatjuk. Példa:

```
# jexec 1 tcsh
```

15.5.2. Magasszintû karbantartó eszközök a FreeBSD Portgyûjteményében

A sok külsõ karbantartó eszköz közül az egyik legteljesebb és leghasznosabb a sysutils/jailutils. Sok kisebb alkalmazást tartalmaz, melyek kibõvítik a jail(8) irányíthatóságát. Bõvebb információkért kérjük, látogassa meg a hozzá tartozó honlapot.

15.6. A jailek alkalmazása

15.6.1. Szolgáltatások jailbe foglalása

Ez a rész eredetileg Simon L. B. Nielsen <simon@FreeBSD.org> http://simon.nitro.dk/service-jails.html oldalon található írásán, valamint Ken Tom (locals@gmail.com) átdolgozott cikkén alapul. Itt megismerhetjük, hogyan állítsunk be a FreeBSD rendszerünkben egy biztonsági réteget a jail(8) felhasználásával. Továbbá feltételezzük, hogy ez a rendszer legalább RELENG_6_0 verziójú és a fejezetben korábban tárgyaltakat az olvasó teljes mértékben megértette.

A kialakítás

A jailek egyik legnagyobb gondja a frissítés folyamatának lebonyolítása. Azért jelent ez egyre inkább gondot, mert minden egyes jailt újra fel kell építenünk a frissítése során. Ez többnyire nem

okoz gondot egyetlen jail használata során, mivel maga a frissítési folyamat meglehetõsen egyszerû, azonban igen idõigényessé és fárasztóvá tud válni több jail esetében.

Ez a példa a FreeBSD képességeinek haladó szintû ismeretét követeli meg. Amennyiben az itt bemutatott lépesek túlságosan is bonyolultnak tûnnének, érdemes olyan egyszerûbb rendszerek után nézni, mint mondjuk a sysutils/ezjail, amely egy egyszerûbb módszert kínál fel a FreeBSD-ben használt jailek karbantartására, és nem is annyira bonyolult, mint ez a példa.

A bemutatandó példa célja, hogy feloldja az ilyen jellegû problémákat, és ezért igyekszik a jailek között mindent megosztani, ami csak lehetséges. Mindezt biztonságosan éri el - írásvédett mount_nullfs(8) állományrendszer használatával, aminek köszönhetõen a frissítés maga egyszerûbbé, az egyes szolgáltatások különzárása pedig vonzóbbá válik. Ráadásul egyúttal egy nagyon egyszerû módszert mutat az új jailek hozzáadására és a régi törlésére ugyanúgy, mint a frissítésükre.

Például ilyen szolgáltatásokat kívánunk szabályozni: egy HTTP szervert, egy DNS szervert, egy SMTP szervert és így tovább.

Az itt szereplõ beállítás céljai:

- Készítsünk egy egyszerûen és könnyen átlátható jailkezelési rendszert. Ebbõl tehát következik, hogy *ne* kelljen lefuttatni a teljes rendszer telepítését minden egyes jailre.
- Könnyítsük meg az új jailek hozzáadását és a régiek eltávolítását.
- Könnyítsük meg a már létezõ jailek frissítését és cseréjét.
- Tegyük lehetõvé saját FreeBSD ágak futtatását.
- Legyünk különösen körültekintõek a biztonság tekintetében, és igyekezzünk minél jobban csökkenteni veszély kockázatát.
- Takarékoskodjunk a tárhellyel és az állományrendszerrel, amennyire csak lehet.

Ahogy azt már korábban is említettük, ez a kialakítás nagyban építkezik egyetlen fõ sablonra, amely írásvédetten kerül csatlakoztatásra (nullfsen keresztül) az egyes jailekben, valamint jailenként egy-egy írható-olvasható eszközre. Ez az eszköz lehet egy külön fizikai lemez, egy partíció vagy egy vnode alapú md(4) eszköz. Ebben a példában írható-olvasható nullfs csatlakozásokat használunk.

Az állományrendszer kiosztása a most következõ listában szerepel:

- Minden jailt a /home/j könyvtárban csatlakoztatunk.
- A /home/j/mroot lesz az összes jail sablonja és mindegyikük számára írásvédett.
- Minden jailnek létrehozunk egy üres alkönyvtárat a /home/j könyvtárban.
- Minden jailnek lesz egy /s alkönyvtára, amelyet a rendszer írható-olvasható részére irányítunk.
- Minden jailnek lesz egy saját írható-olvasható része, amely a /home/j/skel könyvtáron alapszik.
- Mindegyik elzárt terület (a jailek írható-olvasható része) a /home/js könyvtárban jön létre.

Ez a kiosztás feltételezi, hogy a jaileket a /home partíción hozzuk létre. Ez természetesen bármi másra megváltoztatható, de akkor figyelnünk kell erre minden egyes parancs kiadása elõtt.

A sablon létrehozása

Ez a rész leírja a fõ sablon létrehozásához szükséges lépéseket. Ez a jailek számára írásvédett lesz.

Érdemes mindig frissíteni a FreeBSD rendszerünket a legújabb -RELEASE ágra. Ehhez olvassuk el az ide tartozó fejezetet a kézikönyvbõl. Abban az esetben, ha a frissítés nem lenne megoldható, egy `make buildworld` parancsot mindenképpen le kell tudnunk futtatni. Ezenfelül a sysutils/cpdup csomagra is szükségünk van. Használni fogjuk a portsnap(8) segédprogramot is a FreeBSD Portgyûjtemény letöltéséhez. Akik nem ismernék, a kézikönyv errõl szóló fejezetében olvashatnak róla.

1. Elõször is, készítsük el az írásvédett állományrendszer könyvtárszerkezetét, amely majd tartalmazni fogja a jailek által használt FreeBSD-s programokat. Ezután lépjünk be a FreeBSD forrásfájának könyvtárába és telepítsük fel az írásvédett állományrendszert a sablonba:

```
# mkdir /home/j /home/j/mroot
# cd /usr/src
# make installworld DESTDIR=/home/j/mroot
```

2. Ezt követõen készítsük elõ a jailek számára a FreeBSD Portgyûjteményt és FreeBSD forrásfát, melyek kellenek a mergemaster használatához:

```
# cd /home/j/mroot
# mkdir usr/ports
# portsnap -p /home/j/mroot/usr/ports fetch extract
# cpdup /usr/src /home/j/mroot/usr/src
```

3. Hozzuk létre a rendszer írásvédett részének vázát:

```
# mkdir /home/j/skel /home/j/skel/home /home/j/skel/usr-X11R6
/home/j/skel/distfiles
# mv etc /home/j/skel
# mv usr/local /home/j/skel/usr-local
# mv tmp /home/j/skel
# mv var /home/j/skel
# mv root /home/j/skel
```

4. Használjuk a mergemastert a hiányzó konfigurációs állományok telepítésére. Szabaduljunk meg a mergemaster által készített felesleges könyvtáraktól:

```
# mergemaster -t /home/j/skel/var/tmp/temproot -D /home/j/skel -i
# cd /home/j/skel
# rm -R bin boot lib libexec mnt proc rescue sbin sys usr dev
```

5. Most pedig szimbolikusan linkeljük az írható-olvasható állományrendszert az írásvédett állományrendszerre. Ellenõrizzük, hogy a szimbolikus linkek a megfelelõ s/ könyvtárakban jöttek létre. Valós vagy rossz helyen létrehozott könyvtárak használata esetén a telepítés nem fog sikerülni.

```
# cd /home/j/mroot
# mkdir s
# ln -s s/etc etc
# ln -s s/home home
# ln -s s/root root
# ln -s ../s/usr-local usr/local
# ln -s ../s/usr-X11R6 usr/X11R6
# ln -s ../../s/distfiles usr/ports/distfiles
# ln -s s/tmp tmp
# ln -s s/var var
```

6. Utolsó lépésként hozzunk létre egy /home/j/skel/etc/make.conf állományt az alábbi tartalommal:

```
WRKDIRPREFIX?=  /s/portbuild
```

A WRKDIRPREFIX beállításával lehetõvé válik a FreeBSD portok jaileken belüli fordítása. Ne felejtsük el, hogy a portokat tartalmazó könyvtár az írásvédett rendszer része! Az átállított WRKDIRPREFIX azonban megengedi, hogy a fordítások az egyes jailek írható-olvasható részeiben történjenek.

A jailek létrehozása

Most, miután teljesen elkészült a FreeBSD jailek sablonja, be is tudjuk állítani és hozzá is tudjuk venni ezeket az /etc/rc.conf állományhoz. Ebben a példában 3 jail létrehozását láthatjuk: "NS", "MAIL" és "WWW".

1. Írjuk bele a következõ sorokat az /etc/fstab állományba, aminek köszönhetõen az egyes jailek számára elérhetõvé válik az írásvédett sablon és a hozzájuk tartozó írható-olvasható területek:

```
/home/j/mroot   /home/j/ns     nullfs  ro  0   0
/home/j/mroot   /home/j/mail   nullfs  ro  0   0
/home/j/mroot   /home/j/www    nullfs  ro  0   0
/home/js/ns     /home/j/ns/s   nullfs  rw  0   0
```

```
/home/js/mail   /home/j/mail/s nullfs  rw  0   0
/home/js/www    /home/j/www/s  nullfs  rw  0   0
```

Az elsõ helyen nullával jelölt partíciókat a fsck(8) nem fogja ellenõrizni a rendszer indulása során, a második helyen nullával jelölt partíciókat pedig nem fogja menteni a dump(8). Mi egyáltalán nem akarjuk, hogy az fsck ellenõrizze vagy a dump lementse a jailjeinkhez tartozó írásvédett nullfs-partícióinkat. Ezért szerepel végig "0 0" a fentebb szereplõ fstab -bejegyzések utolsó két oszlopában.

2. Állítsuk be a jaileket az /etc/rc.conf-ban:

```
jail_enable="YES"
jail_set_hostname_allow="NO"
jail_list="ns mail www"
jail_ns_hostname="ns.example.org"
jail_ns_ip="192.168.3.17"
jail_ns_rootdir="/usr/home/j/ns"
jail_ns_devfs_enable="YES"
jail_mail_hostname="mail.example.org"
jail_mail_ip="192.168.3.18"
jail_mail_rootdir="/usr/home/j/mail"
jail_mail_devfs_enable="YES"
jail_www_hostname="www.example.org"
jail_www_ip="62.123.43.14"
jail_www_rootdir="/usr/home/j/www"
jail_www_devfs_enable="YES"
```

Azért állítottuk a `jail`névrootdir változó értékét a /usr/home könyvtárra a /home könyvtár helyett, mert a FreeBSD alaptelepítésében a /home könyvtár fizikailag a /usr/home könyvtárral egyezik meg. A `jail`névrootdir változó értékeként megadott könyvtár *nem* tartalmazhat szimbolikus linket, máskülönben a jailek nem lesznek hajlandóak létrejönni. Ennek megállapításában a realpath(1) segédprogram lehet segítségünkre. A korlátozás részleteirõl a FreeBSD-SA-07:01.jail biztonsági figyelmeztetésben olvashatunk.

3. Hozzuk létre az egyes jailek írásvédett állományrendszereihez szükséges csatlakozási pontokat:

```
# mkdir /home/j/ns /home/j/mail /home/j/www
```

4. Telepítsük az írható-olvasható sablont az egyes jailekbe. Figyeljük meg a sysutils/cpdup használatát, amellyel az egyes könyvtárak pontos másolatait hozhatjuk létre:

```
# mkdir /home/js
```

```
# cpdup /home/j/skel /home/js/ns
# cpdup /home/j/skel /home/js/mail
# cpdup /home/j/skel /home/js/www
```

5. Ebben a fázisban a jailek már elkészültek és készen állnak a futásra. Elõször csatlakoztassuk az egyes jailekhez szükséges állományrendszereket, majd indítsuk el ezeket a /etc/rc.d/jail szkripttel:

```
# mount -a
# /etc/rc.d/jail start
```

A jailek most már futnak. Az elindulásuk ellenõrzéséhez használjuk a jls(8) parancsot. Valami ilyesmit láthatunk a kiadása után:

```
# jls
   JID  IP Address      Hostname                      Path
     3  192.168.3.17    ns.example.org                /home/j/ns
     2  192.168.3.18    mail.example.org              /home/j/mail
     1  62.123.43.14    www.example.org               /home/j/www
```

Itt már be tudunk jelentkezni az egyes jailekbe, új felhasználókat tudunk készíteni vagy démonokat tudunk beállítani. A `JID` oszlop mutatja az egyes jailek azonosítási számát. A 3-as `JID` számú jailben az alábbi parancs használatával karbantartási feladatokat elvégezni:

```
# jexec 3 tcsh
```

Frissítés

Idõrõl idõre adódhat, hogy frissítenünk kell a rendszert a FreeBSD egy újabb változatára, vagy egy biztonsági hiba javítása miatt, vagy pedig a már meglevõ jailek számára hasznos újítások bevezetése miatt. Ez a kialakítás megkönnyíti a korábban létrehozott jailjeink frissítését. Továbbá igyekszik minimalizálni a kiesésüket is, mivel a jaileket csak a legutolsó pillanatban fogjuk leállítani. Sõt, még az is lehetõvé válik, hogy visszaállítsuk a korábbi verziót, ha véletlenül valami rosszul sülne el menetközben.

1. Elsõ lépéseként frissítsük magát a befogadó rendszert a megszokott módon. Ezután hozzunk létre egy új írásvédett sablont a /home/j/mroot2 könyvtárban.

```
# mkdir /home/j/mroot2
# cd /usr/src
# make installworld DESTDIR=/home/j/mroot2
# cd /home/j/mroot2
# cpdup /usr/src usr/src
```

```
# mkdir s
```

A `installworld` lefuttatása létrehoz néhány felesleges könyvtárat, melyeket takarítsunk is el:

```
# chflags -R 0 var
# rm -R etc var root usr/local tmp
```

2. Hozzuk újra létre az írható-olvasható szimbolikus linkjeinket a fõ állományrendszerre:

```
# ln -s s/etc etc
# ln -s s/root root
# ln -s s/home home
# ln -s ../s/usr-local usr/local
# ln -s ../s/usr-X11R6 usr/X11R6
# ln -s s/tmp tmp
# ln -s s/var var
```

3. Most érkezett el az idõ, hogy leállítsuk a jaileket:

```
# /etc/rc.d/jail stop
```

4. Válasszuk le az eredeti állományrendszereket:

```
# umount /home/j/ns/s
# umount /home/j/ns
# umount /home/j/mail/s
# umount /home/j/mail
# umount /home/j/www/s
# umount /home/j/www
```

Az írható-olvasható állományrendszerek hozzá vannak kapcsolva az írásvédett állományrendszerhez (/s), ezért azokat elõször le kell választani.

5. Mozgassuk el az útból a régi írásvédett állományrendszerünket és váltsuk fel az újjal. Így biztonsági mentésként és a régi írásvédett rendszer archívumaként továbbra is rendelkezésre áll, ha valami baj történne. Az itt használt elnevezés az újonnan létrehozott írásvédett állományrendszer dátumából ered. Mozgassuk át az eredeti FreeBSD Portgyûjteményt az új állományrendszerre, hogy megtakarítsunk némi tárhelyet és állományleírót:

```
# cd /home/j
# mv mroot mroot.20060601
```

```
# mv mroot2 mroot
# mv mroot.20060601/usr/ports mroot/usr
```

6. Most már készen áll az új írásvédett sablon, így már csak az állományrendszerek újracsatlakoztatása és a jailek újraindítása maradt:

```
# mount -a
# /etc/rc.d/jail start
```

A jls(8) használatával ellenõrizzük, hogy a jailek rendesen elindultak. Ne felejtsük el jailenként lefuttatni a mergemastert sem. A konfigurációs állományokat és az rc.d szkripteket is frissítenünk kell majd.

Chapter 16. Kötelező hozzáférés-vezérlés (MAC)

16.1. Áttekintés

A FreeBSD 5.X változata új biztonsági bõvítéseket vett át a TrustedBSD projektbõl a POSIX®.1e nyomán. A két legjelentõsebb új biztonsági mechanizmus az állományrendszerekben megtalálható hozzáférés-vezérlési listák (Access Control List, ACL) és a kötelezõ hozzáférés-vezérlés (Mandatory Access Control, MAC). A kötelezõ hozzáférés-vezérlés segítségével olyan új hozzáférés-vezérlési modulok tölthetõek be, amelyek új biztonsági házirendeket implementálnak. Némelyek közülük védelmet nyújtanak a rendszer egy szûk részének, amivel így egy adott szolgáltatást bástyáznak alá. Mások minden részletre kiterjedõ címkézett biztonságot szolgáltatnak alanyokon és objektumokon keresztül. A meghatározás "kötelezõ" része onnan fakad, hogy a szabályok betartatását a rendszergazdák és a rendszer végzik, és nem bízzák a felhasználókra, ahogy azt a System V típusú rendszerekben a szabványos állományokra és IPC-re érvényes engedélyeken keresztül a tetszés szerinti hozzáférés-vezérlés (Discretionary Access Control, DAC) teszi.

Ebben a fejezetben a kötelezõ hozzáférés-vezérlést övezõ keretrendszerre (MAC Framework) és a különbözõ biztonsági házirendeket megvalósító, beilleszthetõ modulokra fogunk összpontosítani.

A fejezet elolvasása során megismerjük:

- hogy a FreeBSD jelen pillanatban milyen modulokat tartalmaz a MAC rendszeren belül és milyen mechanizmusok tartoznak hozzájuk;
- hogy a MAC biztonsági házirendjeit képezõ modulok miket valósítanak meg, valamint mi a különbség a címkézett és címkézetlen házirendek között;
- hogyan kell hatékonyan beállítani és használni rendszerünkben a MAC rendszert;
- hogyan állítsuk be a MAC rendszerben található különféle biztonsági házirendeket képezõ modulokat;
- hogyan hozzunk létre a MAC rendszer segítségével egy biztonságosabb környezetet, amire példákat is mutatunk;
- hogyan teszteljük le a MAC rendszer beállításait és bizonyosodjunk meg mûködésének helyességérõl.

A fejezet elolvasásához ajánlott:

- a UNIX® és a FreeBSD alapjainak ismerete (A UNIX alapjai)
- a rendszermag beállításának és lefordításának ismerete (A FreeBSD rendszermag testreszabása)
- tisztában lenni az alapvetõ biztonsági kérdésekkel és azok hatásával a FreeBSD-n belül (Biztonság)

Az itt ismertetésre kerülõ információk helytelen alkalmazása a rendszer hozzáférhetõségének teljes elvesztését, a felhasználók bosszantását vagy az X11 által felkínált lehetõségek kirekesztését eredményezheti. Ami viszont ennél is

> fontosabb, hogy a MAC rendszerre nem úgy kell tekinteni, mint amitõl a rendszerünk tökéletesen biztonságossá válik. A MAC segítségével csupán a meglevõ biztonsági házirendeket gyarapítjuk. A szilárd biztonsági rutin és a rendszeres ellenõrzések elvégzése nélkül a rendszerünk valójában sosem lesz teljesen biztonságos.
>
> Hozzá kell tennünk, hogy a fejezetben bemutatott példák tényleg csak példák. Senkinek sem tanácsoljuk, hogy az itt említett beállításokat egy éles rendszerre is kiterjessze. A különbözõ biztonsági modulok felépítése rengeteg gondolkodást és próbálgatást igényel. Aki nem érti meg az egész mûködését, könnyen azon kaphatja magát, hogy újra végig kell mennie a rendszeren és egyenként be kell állítania minden könyvtárat és állományt.

16.1.1. Amivel itt nem foglalkozunk

Ebben a fejezetben a MAC rendszerrel kapcsolatban rengeteg biztonsági kérdéssel foglalkozni fogunk. Az új MAC biztonsági modulok kifejlesztését azonban már nem érintjük. Számos olyan biztonsági modul található a MAC rendszerben, amelyek rendelkeznek az új modulok kialakításához és teszteléséhez szükséges jellemzõkkel. Ilyenek többek közt a mac_test(4), mac_stub(4) és a mac_none(4). Ezekrõl a biztonsági modulokról és az általuk szolgáltatott mechnanizmusokról a man oldalaik tudnak bõvebb tájékoztatást adni.

16.2. A fejezet fontosabb fogalmai

A fejezet tartalmának kifejtéséhez szükségünk lesz néhány fontosabb alapfogalom tisztázására. Segítségükkel vélhetõen sikerül eloszlatni a téma feldolgozása során felmerülõ félreértéseket, illetve elkerülni az új fogalmak és információk váratlan felbukkanását.

- *alany*: Alanynak tekintünk a rendszerben minden olyan aktív egyedet, amely információt áramoltat az *objektumok*, tehát a felhasználók, a processzorok, a rendszerben futó programok stb. között. A FreeBSD-ben majdnem minden esetben a felhasználók egy szálon keresztül vezérlik a futó programokat.
- *címke*: A címke egy olyan biztonsági tulajdonság, ami vonatkozhat állományokra, könyvtárakra vagy a rendszer más elemeire. Egy címke tekinthetõ a bizalmasságot jelzõ pecsétnek is: ha egy állományra címkét teszünk, akkor benne megadjuk a rá vonatkozó biztonsági jellemzõket, és csak a hozzá hasonló biztonsági beállításokkal rendelkezõ állományok, felhasználók, erõforrások stb. érhetik el. A címkék jelentését és értelmezését a házirendek beállítása határozza meg: míg egyes házirendek a címkéket egy objektum sértetlenségének vagy titkosságának tekintik, addig mások a hozzáféréssel kapcsolatos szabályokat rögzítik bennük.
- *egycímkés*: Egycímkés esetrõl akkor beszélünk, amikor az adat áramlásának szabályozására az egész állományrendszer egyetlen címkét alkalmaz. Ha ezt beállítjuk egy állományrendszernél, de nem adjuk meg vele együtt a `multilabel` opciót, akkor az összes állományra ugyanaz a címke érvényes.
- *erõs vízjel*: Az erõs vízjel házirendje szerint a biztonsági szint akkor növelhetõ, ha magasabb szintû információkhoz akarunk hozzájutni. A legtöbb esetben a folyamatok befejezõdése után visszaállítódik az eredeti szint. A FreeBSD MAC rendszere pillanatnyilag ehhez nem tartalmaz

házirendet, de a teljesség kedvéért megadtuk ennek a definícióját is.

- *gyenge vízjel*: A gyenge vízjel házirendje szerint a biztonsági szint csökkenthetõ az alacsonyabb szintû információk elérése érdekében. A legtöbb esetben a folyamatok befejezõdése után visszaállítódik az eredeti szint. A FreeBSD-ben ezt a házirendet egyedül a mac_lomac(4) alkalmazza.
- *házirend*: Szabályok olyan gyûjteménye, amely megadja, hogy miként kell a célokat teljesíteni. Egy *házirend* általában az egyes elemek kezelését rögzíti. Ebben a fejezetben a *házirend* kifejezés alatt a *biztonsági házirendet* értjük, tehát olyan szabályok gyûjteményét, amelyek az adatok és az információ áramlását határozzák meg, továbbá megadják, hogy közülük ki mihez férhet hozzá.
- *kényesség*: Általában az MLS tárgyalásakor kerül elõ. Az kényesség szintjével az adatok fontosságát vagy titkosságát szokták jelölni. A kényességi szint növekedésével növekszik az adat titkosságának vagy bizalmasságának szintje.
- *objektum*: Objektum vagy rendszerobjektum minden olyan egyed, amelyen információ folyik keresztül az *alanyok* irányításával. Ezek lehetnek többek közt könyvtárak, állományok, mezõk, képernyõk, billentyûzetek, a memória, mágneses tárolóeszközök, nyomtatók vagy bármilyen más adattároló/hordozó eszköz. Az objektumok alapvetõen adattárolók vagy a rendszer erõforrásai. Egy *objektum* elérésén gyakorlatilag az adatok elérését értjük.
- *rekesz*: Egy rekeszbe soroljuk az elrekeszteni vagy elkülöníteni kívánt programok és adatok összeségét, ahol a felhasználók explicit módon képesek hozzáférni a rendszer bizonyos komponenseihez. Emellett a rekesz utalhat egy tetszõleges csoportosításra is, például munkacsoportra, osztályra, projektre vagy témára. A rekeszek használata elengedhetetlen a biztonsági házirendek kialakításához.
- *sértetlenség*: A sértetlenség, mint kulcsfogalom, az adatok megbízhatóságának szintje. Minél sértetlenebb az adat, annál inkább tekinthetjük megbízhatónak.
- *szint*: Egy biztonsági tulajdonság megnövelt vagy lecsökkentett beállítása. A szint növekedésével együtt a biztonság mértéke is növekszik.
- *többcímkés*: A `multilabel` vagyis többcímkés jellemzõ az állományrendszerek esetén fordulhat elõ, és a tunefs(8) segédprogrammal állítható be egyfelhasználós módban vagy a rendszer indítása során az fstab(5) állományon keresztül, esetleg egy új állományrendszer létrehozásakor. Ezzel a beállítással a rendszergazda különféle MAC címkéket rendelhet különbözõ objektumokhoz. Ez a beállítás természetesen csak olyan biztonsági modulok esetén él, amelyek tudnak címkézni.

16.3. A MAC ismertetése

Az imént definiált új fogalmak tükrében most nézzük meg, hogy a MAC rendszer alkalmazásával miként javíthatunk rendszerünk biztonságán. A MAC rendszerhez készített különbözõ biztonsági modulok alkalmasak a hálózat és az állományrendszerek védelmére, valamint segítségükkel megakadályozhatjuk, hogy a felhasználók elérhessenek bizonyos portokat és socketeket stb. A házirendeket formázó modulokat talán együttesen tudjuk a leghatékonyabban alkalmazni, és ha egyszerre több modul betöltésével egy többrétegû védelmi rendszert alakítunk ki. Ez nem ugyanaz, mint a rendszer megerõsítése, ahol a rendszer összetevõit jellemzõ módon csak bizonyos célok tekintetében edzzük meg. A módszer egyedüli hátulütõi a többszörös állományrendszeri címkékkel,

a felhasználónként beállítandó hálózati eléréssel stb. járó adminisztrációs költségek.

Ezek a hátrányok azonban eltörpülnek a létrehozott rendszer tartósságával szemben. Például, ha képesek vagyunk megmondani, hogy az adott konfigurációban milyen házirendek alkalmazására van szükség, akkor ezzel az adminisztrációs költségek visszaszoríthatóak. A szükségtelen házirendek eltávolításával még növelhetjük is a rendszer összteljesítményét, valamint az így felkínált rugalmasságot. Egy jó kialakításban figyelembe kell venni az összes biztonsági elõírást, és hatékonyan megvalósítani ezeket a rendszer által felajánlott különféle biztonsági modulokkal.

Ezért tehát a MAC lehetõségeit kihasználó rendszerekben legalább annyit meg kell tudni oldani, hogy a felhasználók ne változtathassák kedvükre a biztonsági tulajdonságokat. Az összes felhasználói segédprogramnak, programnak és szkriptnek a kiválasztott biztonsági modulokban szereplõ hozzáférési szabályokkal kifeszített kereten belül kell mozognia. A MAC totális irányítása pedig a rendszergazda kezében van.

A rendszergazda így egyedül csak a megfelelõ biztonsági modulok gondos összeválogatásáért felelõs. Bizonyos környezetekben szükséges lehet a hálózaton keresztüli hozzáférések korlátozása is. Ilyen esetekben a mac_portacl(4), mac_ifoff(4) vagy a mac_biba(4) moduloktól érdemes elindulnunk. Más esetekben az állományrendszerek objektumainak bizalmasságát kell csupán megõriznünk. Erre a célra a mac_bsdextended(4) és mac_mls(4) modulok a legalkalmasabbak.

A házirendekhez kapcsolódó döntések a hálózati beállítások alapján is meghozhatóak. Elképzelhetõ, hogy csak bizonyos felhasználók férhetnek hozzá az ssh(1) szolgáltatásain keresztül a hálózathoz vagy az internethez. A mac_portacl(4) pontosan ilyen helyzetekben tud a segítségünkre sietni. Mit tegyünk viszont az állományrendszerek esetén? Vágjunk el adott felhasználókat vagy csoportokat bizonyos könyvtáraktól? Vagy korlátozzuk a felhasználók vagy segédprogramok hozzáférését adott állományokhoz bizonyos objektumok bizalmassá tételével?

Az állományrendszerek esetében az objektumokat néhány felhasználó elérheti, mások pedig nem. Például egy nagyobb fejlesztõcsapat kisebb csoportokra bontható. Az A projektben résztvevõ fejlesztõk nem férhetnek hozzá a B projektben dolgozó fejlesztõk munkájához. Ellenben szükségük lehet a C projekten munkálkodó fejlesztõk által létrehozott objektumokra. Ez egy igen érdekes helyzet. A MAC rendszer által felkínált különbözõ biztonsági modulokra építkezve azonban könnyedén csoportokba tudjuk szervezni a felhasználókat, és a megfelelõ területekhez az információ kiszivárgása nélkül hozzá tudjuk õket engedni.

Ennek következtében minden egyes biztonsági modul a maga módján gondoskodik az egész rendszer biztonságáról. A céljainknak megfelelõ modulokat egy jól átgondolt biztonsági házirend alapján válasszuk ki. Sok esetben az egész házirendet át kell tekinteni és újra kell alkalmazni a rendszerben. A MAC által felajánlott különbözõ biztonsági modulok megértése segít a rendszergazdáknak megválasztani az adott helyzetben legjobban alkalmazható házirendeket.

A FreeBSD rendszermagja alapból nem tartalmazza a MAC rendszert. Ezért a fejezetben szereplõ példák vagy az itt leírtak kipróbálásához az alábbi beállítást kell hozzátennünk a rendszermag beállításait tartalmazó állományhoz:

```
options MAC
```

Majd fordítsuk és telepítsük újra a rendszermagot.

Miközben a MAC rendszerhez készült különbözõ modulok a saját man oldalaik szerint igénylik a beépítésüket, vigyázzunk velük, mert ezzel a rendszerüket pillanatok alatt ki tudjuk zárni a hálózatból és így tovább. A MAC alapú védelem felépítése leginkább egy tûzfal összeállításához hasonlítható, ahol ugyanígy számolni kell azzal, hogy egy óvatlan paranccsal kizárhatjuk magunkat a rendszerbõl. Valamilyen módon mindig próbáljunk gondoskodni a rendszer elõzõ állapotának visszaállíthatóságáról, és a MAC távoli adminisztrációját mindig nagyfokú körültekintéssel végezzük.

16.4. Bõvebben a MAC címkéirõl

A MAC-címke egy olyan biztonsági tulajdonság, amelyet a rendszerben található alanyokhoz és objektumokhoz rendelhetünk.

Egy címke beállításához a felhasználónak pontosan ismernie kell, hogy ilyenkor mi történik. Az objektumokhoz tartozó tulajdonságok a betöltött moduloktól függenek, és az egyes modulok eltérõ módon értelmezik ezeket a tulajdonságokat. Ha a precíz megértésük hiányában helytelenül állítjuk be ezeket, vagy nem vagyunk képesek tisztázni a velük járó következményeket, akkor az a rendszerünk kiszámíthatatlan és valószínûleg kedvezõtlen viselkedését eredményezi.

A házirendek az objektumhoz rendelt biztonsági címkéket a hozzáféréssel kapcsolatos döntések meghozásában használják fel. Bizonyos házirendek esetében már maga a címke elegendõ információt tartalmaz a döntés megformálásához. Máshol viszont a címkék egy nagyobb szabályrendszer részeként dolgozódnak fel stb.

Például, ha egy állományra beállítjuk a `biba/low` címkét, akkor az arra fog utalni, hogy a címkét a Biba nevû biztonsági modul kezeli és értéke "low".

Az a néhány modul, amely a FreeBSD-ben támogatja a címkézés lehetõségét, három speciális címkét definiál elõre. Ezek rendre a "low" (alacsony), "high" (magas) és "equal" (egyezõ) címkék. Habár az egyes modulok esetén eltérõ módon képesek vezérelni a hozzáférést, azt mindig biztosra vehetjük, hogy a "low" a legalacsonyabb érték, az "equal" címke hatására az adott alanyt vagy objektumot érintetlenül hagyják, és a "high" értékû címke a Biba és MLS modulok esetében a legmagasabb beállítást jelenti.

Az egycímkés állományrendszerek használata során az egyes objektumonkhoz csak egyetlen címkét rendelhetünk hozzá. Ezzel az egész rendszerben csak egyfajta engedélyt alkalmazunk, ami sok esetben pontosan elegendõ. Létezik néhány különleges eset, amikor az állományrendszerben levõ alanyokhoz vagy objektumokhoz egyszerre több címkét is hozzá kell rendelnünk. Ilyenkor a `multilabel` opciót kell átadnunk a tunefs(8) segédprogramnak.

A Biba és az MLS esetében elõfordulhat, hogy egy numerikus címkével fogjuk jelölni a hierarchikus irányítás pontos szintjét. A numerikus szintek használatával tudjuk az információt különbözõ csoportokba szétosztani vagy elrendezni, például úgy, hogy csak az adott szintû vagy a felette álló csoportok számára engedélyezzük a hozzáférést.

Az esetek többségében a rendszergazdának csak egyetlen címkét kell beállítania az egész

állományrendszerre.

Hé, álljunk csak meg! Akkor ez viszont pont olyan, mint a DAC! Én azt hittem, hogy a MAC szigorúan a rendszergazda kezébe adja az irányítást. Ez az állítás továbbra is fennáll, mivel bizonyos értelemben a `root` lesz az, aki beállítja a házirendeket, tehát õ mondja meg, hogy a felhasználók milyen kategóriákba vagy hozzáférési szintekbe sorolódnak. Sajnos, sok biztonsági modul még magát a `root` felhasználót is korlátozza. Az objektumok feletti irányítás ilyenkor a csoportra száll, de a `root` bármikor visszavonhatja vagy módosíthatja a beállításokat. Ezzel a hierarchikus/engedély alapú modellel a Biba és az MLS nevû házirendek foglalkoznak.

16.4.1. A címkék beállítása

A címkézéshez kapcsolódó összes beállítást gyakorlatilag az alapvetõ rendszerprogramokkal végezhetjük el. Ezek a parancsok az objektumok és az alanyok szabályozásához, valamint a konfiguráció módosításához és ellenõrzéséhez adnak egy egyszerû kezelõfelületet.

Az összes konfigurációs beállítást a setfmac(8) és setpmac(8) segédprogramokkal végezhetjük el. A `setfmac` segítségével a rendszerszintû objektumokhoz tudunk hozzárendelni a MAC-címkéket, míg a `setpmac` paranccsal a rendszerben levõ alanyokhoz tudunk címkéket rendelni. Vegyük például ezt:

```
# setfmac biba/high próba
```

Amennyiben az iménti parancs hibátlanul lefutott, visszakapjuk a paranccsort. Ezek a parancsok csak olyankor maradnak nyugodtan, amikor semmilyen hiba nem történt. Mûködésük hasonló a chmod(1) és chown(8) parancsokéhoz. Bizonyos esetekben `Permission denied` (`A hozzáférés nem engedélyezett`) hibát kapunk, ami általában akkor bukkan fel, ha egy korlátozott objektummal kapcsolatban próbálunk meg címkét beállítani vagy módosítani . A rendszergazda a következõ paranccsal tudja feloldani az ilyen helyzeteket:

```
# setfmac biba/high próba
Permission denied
# setpmac biba/low setfmac biba/high próba
# getfmac próba
próba: biba/high
```

Ahogy az itt tetten is érhetõ, a `setpmac` használható a modul beállításainak felülbírálására úgy, hogy a meghívott programban egy másik címkét állít be. A `getpmac` segédprogram általában a sendmailhez hasonló háttérben futó programok esetében alkalmazható: ilyenkor a konkrét parancs helyett a futó program azonosítóját kell megadnunk, de mûködése ugyanaz. Ha a felhasználók a hatókörükön túl levõ állományokat próbálnak meg módosítani, akkor a betöltött modulok szabályainak megfelelõen a `mac_set_link` függvény `Operation not permitted` (`A mûvelet nem engedélyezett`) hibát fog adni.

Gyakori címketípusok

A mac_biba(4), mac_mls(4) és mac_lomac(4) moduloknál használhatunk címkéket. Értékük lehet "high", "equal" vagy "low", melyek rövid magyarázata a következõ:

- A `low` címke az objektumra vagy alanyra érvényes leggyengébb beállítást jelenti. Az ilyen címkéjû objektumok vagy alanyok nem érhetik el a "high" címkéjûeket.
- Az `equal` címke használható minden olyan objektum vagy alany esetében, amelyeket ki akarunk vonni az adott házirend hatálya alól.
- A `high` címke adja az objektumhoz vagy alanyhoz tartozó legerõsebb beállítást.

Az egyes moduloktól függõen ezek az értékek az információ áramoltatásának különbözõ irányait írhatják le. A megfelelõ man oldalak elolvasásával még jobban megismerhetjük az egyes címketípusok beállításának jellegzetességeit.

A címkék beállításáról részletesebben

A numerikus osztályozó címkék `összehasonlítás:rekesz+rekesz` alakban használatosak, tehát a

```
biba/10:2+3+6(5:2+3-20:2+3+4+5+6)
```

kifejezés így értelmezhetõ:

"A Biba házirend címkéje"/"10 osztály" :"2, 3 és 6 rekeszek": ("5 osztály...")

Ebben a példában az elsõ osztály tekinthetõ "valódi osztálynak", amely a "valódi rekeszeket" jelenti, a második osztály egy alacsonyabb besorolás, míg az utolsó egy magasabb szintû. A legtöbb konfigurációban nem lesz szükségünk ennyire összetett beállításokra, noha képesek vagyunk felírni ezeket.

Ha ezt kivetítjük a rendszer objektumaira, akkor a rendszerben levõ alanyokat illetõen csupán az aktuális osztály/rekeszek számítanak, mivel a rendszerben és hálózati csatolófelületeken elérhetõ hozzáférés-vezérlési jogokat tükrözi.

Az alany-objektum párokban megadott osztályzatok és rekeszek használhatóak fel egy olyan kapcsolat kiépítésére, amit "dominanciának" nevezünk. Ilyenkor egy alany ural egy objektumot, vagy egy objektum ural egy alanyt, vagy egyikük sem uralja a másikat, esetleg mind a kettõ uralja egymást. A "kettõs dominancia" esete akkor forog fenn, amikor a két címke megegyezik. A Biba információáramoltatási sajátosságaiból adódóan jogunk van rekeszeket létrehozni, "tudunk kell", hogy ezek projekteknek feleltethetõek meg, de az objektumok is rendelkezhetnek rekeszekkel. A felhasználók ilyenkor csak úgy tudnak elérni egyes objektumokat, ha az `su` vagy a `setpmac` használatával leszûkítik a jogaikat egy olyan rekeszre, ahol már nem érvényesülnek rájuk korlátozások.

A felhasználók és címkék kapcsolata

Maguknak a felhasználóknak is szükségük van címkékre, mivel csak ezek segítségével tudnak az állományaik és programjaik megfelelõ módon együttmûködni a rendszerben érvényes biztonsági házirenddel. Ezt a login.conf állományban megadható bejelentkezési osztályokkal állíthatjuk be. Minden címkéket használó modulban a felhasználóknak is van címkéjük.

Lentebb látható egy ilyen minta bejegyzés, amely minden modulhoz tartalmaz beállítást:

```
default:\
    :welcome=/etc/motd:\
    :setenv=MAIL=/var/mail/$,BLOCKSIZE=K:\
    :path=~/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin:\
    :manpath=/usr/shared/man /usr/local/man:\
    :nologin=/usr/sbin/nologin:\
    :cputime=1h30m:\
    :datasize=8M:\
    :vmemoryuse=100M:\
    :stacksize=2M:\
    :memorylocked=4M:\
    :memoryuse=8M:\
    :filesize=8M:\
    :coredumpsize=8M:\
    :openfiles=24:\
    :maxproc=32:\
    :priority=0:\
    :requirehome:\
    :passwordtime=91d:\
    :umask=022:\
    :ignoretime@:\
    :label=partition/13,mls/5,biba/10(5-15),lomac/10[2]:
```

Itt a `label` opciót használtuk a felhasználói osztályhoz tartozó alapértelmezett címkék beállításához, amit majd a MAC betartat. A felhasználók nem módosíthatják ezt az értéket, ezért ez a felhasználók számára nem tekinthetõ tetszõlegesen elhagyható beállításnak. Egy valós konfigurációban azonban a rendszergazda valószínûleg nem akarja majd egyszerre az összes modult használni. Javasoljuk, hogy mielõtt egy ilyen jellegû konfigurációt adnánk meg, olvassuk el az egész fejezetet.

A felhasználók ezt a címkét meg tudják változtatni az elsõ bejelentkezés után, de csak a házirend keretein belül. A fenti példában úgy állítjuk be a Biba házirendet, hogy a futó programok sértetlenségi foka legalább 5, legfeljebb 15 lehet, de az alapértéke 10. Tehát a programok egészen addig 10-es szinten futnak, amíg a programok a Biba bejelentkezéskor megadott tartományában meg nem változtatják ezt a címkét, feltehetõen a `setpmac` parancs hatására.

Mindig, amikor megváltozatjuk a login.conf beállításait, a `cap_mkdb` paranccsal újra kell generálni a bejelentkezési osztályokhoz tartozó adatbázist, amire a késõbbi példákban vagy részekben igyekszünk is mindig felhívni a figyelmet.

Nem árt hozzátennünk, hogy sok rendszerben kifejezetten sok felhasználót kell kezelnünk, amihez több különbözõ bejelentkezési osztályra is szükségünk lehet. Mivel késõbb már csak egyre jobban bonyolódni fog a felhasználók kezelése, ezért soha ne felejtsünk el komolyan elõre tervezni.

A FreeBSD következõ változataiban meg fognak jelenni más módszerek is a felhasználók és címkék közti kapcsolatok kezelésére. A FreeBSD 5.3 elõtt azonban ez még semmiképpen sem várható.

A hálózati csatolófelületek és a címkék kapcsolata

A hálózati csatlakozások esetében is állíthatunk be címkéket, melyek a hálózaton keresztül folyó adatok áramlását határozzák meg. Minden esetben ugyanúgy mûködnek, mint ahogy a házirendek az objektumokra. Például a `biba` esetében a magas beállításokkal rendelkezõ felhasználók nem férhetnek hozzá az alacsonyabb címkéjû hálózati csatolófelületekhez.

Ha MAC-címkéket akarunk rendelni egy hálózati felülethez, akkor az `ifconfig` parancsnak adjuk meg a `maclabel` paramétert. Például a

```
# ifconfig bge0 maclabel biba/equal
```

parancs beállítja a `biba/equal` MAC-címkét a bge(4) felületre. A `biba/high(low-high)` alakú címkéket átadásukhoz idézõjelek közé kell tenni, különben hibát kapunk.

Minden címkézést támogató modulhoz tartoznak futási idõben állítható paraméterek, amelyekkel akár le is tudjuk tiltani a MAC-címkéket a hálózati csatolófelületeken. Ugyanezt jelenti egyébként, ha `equal` értéket adunk meg a címkének. Ezt behatóbban úgy ismerhetjük meg, ha kielemezzük a `sysctl` parancs kimenetét, a megfelelõ modul man oldalát vagy a fejezetben további részében található, erre vonatkozó információkat.

16.4.2. Egy címke vagy több címke?

Alapértelmezés szerint a rendszer a `singlelabel` beállítást használja. Ez vajon mit tartogat a rendszergazda számára? Számos olyan eltérést, aminek megvannak a saját elõnyei és hátrányai a rendszer védelmi modelljének rugalmassága szempontjából.

A `singlelabel` beállítás minden alany vagy objektum esetében csupán egyetlen címke, például a `biba/high` használatát engedi. Kevesebb adminisztrációs költséggel jár, azonban csökkenteni a címkézést támogató modulok testreszabhatóságát. Ezért sok rendszergazda inkább a `multilabel` beállítást választja a biztonsági házirend kialakítása során.

A `multilabel` beállítás lehetõvé teszi, hogy mindegyik alanyhoz és objektumhoz a szabványos `singlelabel` beállítás lehetõségeivel szemben egymástól függetlenül külön-külön rendelhessünk címkéket a partíciókon. Az egy- és többcímkés opciónak csak olyan modulok esetében van értelme, amelyek támogatják a címkézést, mint például a Biba, Lomac, MLS és a SEBSD házirendek.

Sokszor egyáltalán nincs is szükségünk a `multilabel` használatára. Tekintsük például a következõ helyzetet és biztonsági modellt:

- Adott egy FreeBSD webszerver, ahol a MAC rendszert több biztonsági házirenddel alkalmazzuk.
- A gépen egyedül csak a `biba/high` címkére van szükségünk mindenhez a rendszerben. Itt egyszerûen csak nem adjuk meg az állományrendszernek a `multilabel` beállítást, mivel az egycímkés rendszer mindig rendelkezésünkre áll.
- Mivel azonban erre a gépre telepíteni akarunk egy webszervert is, ilyenkor a `biba/low` címke használatával igyekszünk korlátozni a szerver feldolgozási képességeit. A Biba házirendrõl és annak mûködésérõl csak a késõbbiekben fogunk írni, ezért ha az elõbbi megjegyzést még nem teljesen értjük, akkor egyszerûen csak olvassunk tovább és térjünk vissza ide. A szerver futása

alatt, vagy legalább is idejének nagy részében egy külön partíciót használhatna, amire a `biba/low` címkét állítanánk be. Természetesen ez a példa korántsem teljes, hiszen hiányoznak belõle az adatokra érvényes korlátozások, a konfigurációs és felhasználói beállítások. Ez csupán az iménti gondolatmenet gyors illusztrációja.

Amennyiben címkézést nem támogató modulokat alkalmazunk, a `multilabel` beállításra szinte sosem lesz szükségünk. Ilyenek például a `seeotheruids`, `portacl` és `partition` házirendek.

A `multilabel` opció használata és így speciális, többcímkés védelmi modell létrehozása képes elbonyolítani a rendszer karbantartását, mert ilyenkor az állományrendszerben mindennek lennie kell címkéjének: könyvtáraknak, állományok és még az eszközleíróknak is.

A most következõ paranccsal beállítjuk az állományrendszerre a `multilabel` opciót. Ez csak egyfelhasználós módban tehetõ meg:

```
# tunefs -l enable /
```

A lapozópartíció esetében erre nincs szükség.

Elõfordulhat, hogy néhány felhasználónak nem sikerül a `multilabel` opciót beállítania a rendszerindító partícióra. Ha ez történne, akkor olvassuk el a fejezet A hibák elhárítása a MAC rendszerbenát.

16.5. A védelem megtervezése

Mindig hasznos idõt szánni a tervezésre, amikor nekilátunk egy új technológia alkalmazásához. A tervezés közben a rendszergazdának "egyben kell látnia a képet", lehetõleg az alábbiak figyelembevételével:

- Elvárások a modell felé
- A modell célkitûzései

Továbbá a MAC használata esetén:

- Miként osztályozzuk a célrendszeren rendelkezésre álló információt és erõforrásokat
- Milyen információt vagy erõforrást kell korlátoznunk és milyen típusú korlátozást alkalmazzunk rájuk
- A MAC melyik moduljain keresztül tudjuk elérni céljainkat

Habár mindig módunkban áll megváltoztatni és újra konfigurálni a rendszerben található erõforrásokat és biztonsági beállításokat, sokszor azért igen kényelmetlen utánanézni a rendszerben és állítgatni az állományok, illetve felhasználói hozzáférések paramétereit. A beállításainkat valamint azok konfigurációját *elõször* külön próbáljuk ki, mielõtt a MAC alapú megvalósításunkat egy éles rendszeren kezdjük el használni. Ennek elhagyása szinte biztosan kudarcra ítél minket.

A különbözõ környezetek igényei és elvárásai eltérnek. Egy alaposan és minden részletében

átgondolt védelmi profil megalapozása csökkenti a rendszer üzembehelyezése után elvégzendõ módosítások számát. Mint olyanokra, a következõ szakaszokban kitérünk a rendszergazdák számára elérhetõ modulokra, bemutatjuk a használatukat és beállításukat és egyes esetekben betekintést is adunk olyan helyzetekbe, ahol a legjobban kiaknázhatóak a képességeik. Például egy webszerver esetén hasznos lehet a mac_biba(4) és mac_bsdextended(4) házirendek alkalmazása. Más esetekben, például egy kevés felhasználóval mûködõ számítógépen, a mac_partition(4) modul lehet jó választás.

16.6. A modulok beállítása

A MAC rendszerben megtalálható összes modul a korábban leírtak szerint beépíthetõ a rendszermagba vagy menet közben is betölthetõ modulként. A használni kívánt modulokat a /boot/loader.conf állományba javasolt felvenni, így azok be tudnak töltõdni a rendszer indítása folyamán.

A soron következõ szakaszokban a különbözõ MAC-modulokat dolgozzuk fel és foglaljuk össze a lehetõségeiket. Továbbá a fejezet szeretne szólni ezek alkalmazásáról speciális helyzetekben is. Egyes modulokkal címkézni is tudunk, aminek révén a hozzáféréseket címkékkel szabályozzuk, például úgy, hogy megmondjuk "mit szabad és mit nem". A címkék beállításait tartalmazó állomány vezérli az állományok elérését, a hálózati kommunikációt és még sok minden mást. Az elõzõ szakaszban már megismerhettük, hogy a `multilabel` opció segítségével hogyan állíthatjuk be az állományonkénti vagy partíciónkénti hozzáférés-vezérlést.

Az egycímkés konfigurációban az egész rendszerben csupán egyetlen címke használatára nyílik mód, ezért is hívják a `tunefs` beállítását `multilabel`nek.

16.7. A seeotheruids MAC-modul

A modul neve: mac_seeotheruids.ko

A rendszermag konfigurációs beállítása: `options MAC_SEEOTHERUIDS`

Rendszerindítási beállítás: `mac_seeotheruids_load="YES"`

A mac_seeotheruids(4) modul a `security.bsd.see_other_uids` és `security.bsd.see_other_gids sysctl` -változókat utánozza és terjeszti ki. A használatához semmilyen címkét nem kell beállítani és transzparens módon képes együttmûködni a többi modullal.

A modult betöltése után az alábbi `sysctl`-változókkal tudjuk vezérelni:

- A `security.mac.seeotheruids.enabled` engedélyezi a modult és az alapértelmezett beállításokat használja. Alapértelmezés szerint egyik felhasználó sem láthatja a többiek futó programjait és csatlakozásait.
- A `security.mac.seeotheruids.specificgid_enabled` egy adott csoportot mentesít a házirend szabályozásai alól. Tehát ki akarunk vonni egy csoportot a házirend alkalmazásából, akkor állítsuk be a `security.mac.seeotheruids.specificgid=XXX sysctl`-változót, ahol az *XXX* a mentesíteni kívánt csoport numerikus azonosítója.
- A `security.mac.seeotheruids.primarygroup_enabled` segítségével adott elsõdleges csoportokat

vonhatunk ki a házirend hatálya alól. Ezt a változót nem használhatjuk a `security.mac.seeotheruids.specificgid_enabled` változóval együtt.

16.8. A bsdextended MAC-modul

A modul neve: mac_bsdextended.ko

A rendszermag konfigurációs beállítása: `options MAC_BSDEXTENDED`

Rendszerindítási beállítás: `mac_bsdextended_load="YES"`

A mac_bsdextended(4) modul segítségével egy állományrendszer szintjén mûködõ tûzfalat tudunk kialakítani. Ez a modul a szabványos állományrendszeri engedély alapú modelljét bõvíti ki, lehetõvé téve, hogy a rendszergazda tûzfalszerû szabályokkal nyújtson védelmet a könyvtárszerkezetben található állományoknak, segédprogramoknak és könyvtáraknak. Amikor egy állományrendszerbeli objektumhoz próbálunk meg hozzáférni, a modul illeszti ezt egy szabályrendszerre, amiben vagy talál egy hozzá tartozó szabályt vagy kifut belõle. Ez a viselkedés a `security.mac.bsdextended.firstmatch_enabled` sysctl(8) paraméter segítségével változtatható meg. Hasonlóan a FreeBSD-ben található többi tûzfalmodulhoz, az állományok elérését definiáló szabályok a rendszerindítás során egy rc.conf(5) változóból olvasódnak be.

A szabályokat a ugidfw(8) segédprogrammal adhatjuk meg, amelynek a formai szabályai hasonlóak az ipfw(8) programéhoz. A libugidfw(3) függvénykönyvtár felhasználásával azonban további segédprogramok is írhatóak hozzá.

A modul használata során igyekezzünk minél jobban odafigyelni, mert helytelen alkalmazásával el tudjuk vágni magunkat az állományrendszer bizonyos részeitõl.

16.8.1. Példák

Miután sikerült betölteni a mac_bsdextended(4) modult, a következõ paranccsal tudjuk lekérdezni a jelenleg érvényes szabályokat:

```
# ugidfw list
0 slots, 0 rules
```

Ahogy az várható is volt, pillanatnyilag még egyetlen szabályt sem adtunk meg. Ennek értelmében tehát mindent el tudunk érni. A következõ paranccsal tudunk olyan szabályt létrehozni, ahol a `root` kivételével elutasítjuk az összes felhasználó hozzáférését:

```
# ugidfw add subject not uid root new object not uid root mode n
```

Ez egyébként egy nagyon buta ötlet, mivel így a felhasználók még a legegyszerûbb parancsokat, mint például az `ls`-t, sem tudják rájuk kiadni. Ennél sokkal humánusabb lesz, ha:

```
# ugidfw set 2 subject uid felhasználó1 object uid felhasználó2 mode n
```

```
# ugidfw set 3 subject uid felhasználó1 object gid felhasználó2 mode n
```

Ilyenkor a `felhasználó1` nevû felhasználótól megvonjuk a `felhasználó2` felhasználói könyvtárának összes hozzáférését, beleértve a listázhatóságot is.

A `felhasználó1` helyett megadhatjuk a `not uid felhasználó2` opciót is. Ebben az esetben egy felhasználó helyett az összes felhasználóra ugyanaz a korlátozás fog érvényesülni.

A `root` felhasználóra ezek a beállítások nem vonatkoznak.

Ezzel felvázoltuk, miként lehet a mac_bsdextended(4) modult felhasználni az állományrendszerek megerõsítésére. Részletesebb információkért járuljunk a mac_bsdextended(4) és ugidfw(8) man oldalakhoz.

16.9. Az ifoff MAC-modul

A modul neve: mac_ifoff.ko

A rendszermag konfigurációs beállítása: `options MAC_IFOFF`

Rendszerindítási beállítás: `mac_ifoff_load="YES"`

A mac_ifoff(4) modul kizárólag abból a célból készült, hogy segítségével menet közben le tudjuk tiltani bizonyos hálózati csatolófelületek beállítását a rendszerindítás közben. Sem címkékre, sem pedig a többi MAC-modulra nincs szükségünk a használatához.

A vezérlést nagyrészt az alábbi `sysctl`-változókkal tudjuk megoldani.

- A `security.mac.ifoff.lo_enabled` engedélyezi vagy letiltja a (lo(4)) helyi loopback felületen az összes forgalmat.
- A `security.mac.ifoff.bpfrecv_enabled` engedélyezi vagy letiltja a Berkeley csomagszûrõ (BPF, Berkeley Packet Filter) felületén az összes forgalmat.
- A `security.mac.ifoff.other_enabled` engedélyezi vagy letiltja az összes többi csatolófelületen az összes forgalmat.

A mac_ifoff(4) modult általában olyan környezetek monitorozásakor szokták használni, ahol a rendszer indítása során még nem szabad hálózati forgalomnak keletkeznie. Vagy például a security/aide porttal együtt használva automatikusan el tudjuk zárni a rendszerünket, ha a védett könyvtárakban új állományok keletkeznek vagy megváltoznak a régiek.

16.10. A portacl MAC-modul

A modul neve: mac_portacl.ko

A rendszermag konfigurációs beállítása: `MAC_PORTACL`

Rendszerindítási beállítás: `mac_portacl_load="YES"`

A mac_portacl(4) modul a helyi TCP és UDP portok kiosztásának korlátozását teszi lehetõvé különféle sysctl-változókon keresztül. A mac_portacl(4) segítségével lényegében a nem-root felhasználók is használhatnak privilegizált, tehát 1024 alatti portokat.

Miután betöltöttük, a modul az összes csatlakozásra alkalmazza a MAC-házirendet. Ezután az alábbi változókkal hangolhatjuk a viselkedését:

- A security.mac.portacl.enabled totálisan engedélyezi vagy letiltja a házirend használatát.
- A security.mac.portacl.port_high megadja azt a legmagasabb portot, amelyre még kiterjed a mac_portacl(4) védelme.
- Ha a security.mac.portacl.suser_exempt változónak nem nulla értéket adunk meg, akkor azzal a root felhasználót kivonjuk a szabályozások alól.
- A security.mac.portacl.rules az érvényes mac_portacl házirendet adja meg, lásd lentebb.

A security.mac.portacl.rules változó által megadott aktuális mac_portacl házirend formátuma a következõ: szabály[,szabály,…], ahol ezen a módon tetszõleges számú szabályt adhatunk meg. Az egyes szabályok pedig így írhatóak fel: azonosítótípus: azonosító: protokoll: port. Az azonosítótípus értéke uid vagy gid lehet, amivel megadjuk, hogy az azonosító paraméter felhasználóra vagy csoportra hivatkozik. A protokoll paraméter adja meg, hogy a szabályt TCP vagy UDP típusú kapcsolatra értjük, és ennek megfelelõen az értéke is tcp vagy udp lehet. A sort végül a port paraméter zárja, ahol annak a portnak számát adjuk meg, amelyhez az adott felhasználót vagy csoportot akarjuk kötni.

Mivel a szabályokat közvetlenül maga a rendszermag dolgozza fel, ezért a felhasználók illetve csoportok azonosítója, valamint a port értéke kizárólag numerikus érték lehet. Tehát a szabályokban név szerint nem hivatkozhatunk felhasználókra, csoportokra vagy szolgáltatásokra.

A UNIX®-szerû rendszereken alapértelmezés szerint az 1024 alatti portokat csak privilegizált programok kaphatják meg és használhatják, tehát a root felhasználó neve alatt kell futniuk. A mac_portacl(4) azonban a nem privilegizált programok számára is lehetõvé teszi, hogy elfoglalhassanak 1024 alatti portokat, amihez viszont elõször le kell tiltani ezt a szabvány UNIX®-os korlátozást. Ezt úgy érhetjük el, ha a net.inet.ip.portrange.reservedlow és net.inet.ip.portrange.reservedhigh változókat egyaránt nullára állítjuk.

A mac_portacl(4) mûködésének részleteirõl a példákon keresztül vagy a megfelelõ man oldalakból tudhatunk meg többet.

16.10.1. Példák

A következõ példák az iméntieket igyekeznek jobban megvilágítani:

```
# sysctl security.mac.portacl.port_high=1023
# sysctl net.inet.ip.portrange.reservedlow=0 net.inet.ip.portrange.reservedhigh=0
```

Elsõként beállítjuk, hogy a mac_portacl(4) vegye át a szabványos privilegizált portok vezérlését és letiltjuk a normál UNIX®-os korlátozásokat.

```
# sysctl security.mac.portacl.suser_exempt=1
```

A `root` felhasználót azonban nem akarjuk kitenni a házirendnek, ezért a `security.mac.portacl.suser_exempt` változónak egy nem nulla értéket adunk meg. A mac_portacl(4) modul most pontosan ugyanúgy mûködik, mint a UNIX®-szerû rendszerek alapértelmezés szerint.

```
# sysctl security.mac.portacl.rules=uid:80:tcp:80
```

A 80-as azonosítóval rendelkezõ felhasználó (aki általában a `www`) számára engedélyezzük a 80-as port használatát. Így a `www` felhasználó anélkül képes webszervert futtatni, hogy szüksége lenne a `root` jogosultságaira.

```
# sysctl security.mac.portacl.rules=uid:1001:tcp:110,uid:1001:tcp:995
```

Az 1001-es azonosítóval rendelkezõ felhasználónak megengedjük, hogy elfoglalhassa a 110-es ("pop3") és 995-ös ("pop3s") portokat. Ennek köszönhetõen az adott felhasználó el tud indítani egy szervert, amihez a 110-es és 995-ös portokon lehet kapcsolódni.

16.11. A partition MAC-modul

A modul neve: mac_partition.ko

A rendszermag konfigurációs beállítása: `options MAC_PARTITION`

Rendszerindítási beállítás: `mac_partition_load="YES"`

A mac_partition(4) házirend a futó programokat címkéjük szerint adott "partíciókra" osztja szét. Ezt leginkább egy speciális jail(8) megoldásként tudjuk elképzelni, noha teljesen felesleges összehasonlítani a kettõt.

Ez egy olyan modul, amelyet a loader.conf(5) állományba kell felvenni, hogy a rendszerindítása közben be tudjon töltõdni.

Ezt a házirendet többségében a setpmac(8) segédprogrammal tudjuk állítgatni, ahogy az majd lentebb látható lesz. A következõ `sysctl`-változó tartozik még a modulhoz:

- A `security.mac.partition.enabled` engedélyezi a futó programok MAC rendszeren keresztüli felosztását.

A házirend engedélyezésével a felhasználók csak a saját programjaikat láthatják, illetve mindazokat, amelyek az övékével egy partícióba tartoznak, de a rajta kívül levõ programokkal már nem dolgozhatnak. Például, ha egy felhasználó az `insecure` ("nem biztonságos") osztály tagja, akkor ne engedjük, hogy hozzáférhessen a `top` vagy bármilyen más olyan parancshoz, amely további futó programokat hoz létre.

A `setpmac` használatával tudunk címkéket készíteni a partíciókhoz és programokat rendelni

hozzájuk:

```
# setpmac partition/13 top
```

Így a `top` parancsot hozzáadjuk az `insecure` osztályban levõ felhasználókhoz rendelt címkéhez. Vegyük észre, hogy az `insecure` osztályba tartozó felhasználók által elindított összes program a `partition/13` címkét fogja használni.

16.11.1. Példák

A következõ parancs megmutatja a partíciók címkéit és a futó programok listáját:

```
# ps Zax
```

Ezzel paranccsal pedig megnézhetjük egy másik felhasználó programjainak címkéit és a felhasználó által futtatott programokat:

```
# ps -ZU trhodes
```

A felhasználók látják a `root` címkéjével futó programokat is, hacsak be nem töltjük a mac_seeotheruids(4) házirendet.

Ezt a megoldást úgy tudnánk igazán ravaszul felhasználni, ha például az /etc/rc.conf állományban letiltanánk az összes szolgáltatást és egy olyan szkripttel indítanánk el ezeket, amely futtatásuk elõtt beállítja hozzájuk a megfelelõ címkét.

A most következõ házirendek a három alapértelmezett címkeérték helyett egész számokat használnak. Ezekrõl, valamint a rájuk vonatkozó korlátozásokról a megfelelõ modulok man oldalain ismerhetünk meg többet.

16.12. A többszintû biztonsági MAC-modul

A modul neve: mac_mls.ko

A rendszermag konfigurációs beállítása: `options MAC_MLS`

Rendszerindítási beállítás: `mac_mls_load="YES"`

A mac_mls(4) (MLS, Multi-Level Security) házirend az információ szigorú áramoltatásával vezérli a rendszerben található alanyok és objektumok közti elérést.

A MLS megoldását alkalmazó környezetekben a rekeszek mellett minden alanyra és objektumra be kell még állítanunk egy adott szintû "engedélyt" is. Mivel az engedélyek avagy az érzékenység szintje akár a hatezret is meghaladhatja, egy rendszergazda számára valódi rémálommá válthat az egyes alanyok és objektumok precíz beállítása. Szerencsére a házirend erre a célra tartalmaz

három elõre definiált "instant" címkét.

Ezek az `mls/low`, `mls/equal` és `mls/high`. Mivel a man oldal elég részletesen kifejti ezeket, ezért itt csak érintõlegesen foglalkozunk velük:

- Az `mls/low` címke egy olyan alacsony szintû beállítást képvisel, amely lehetõvé teszi, hogy az összes többi objektum uralja. Tehát bárminek is adjuk az `mls/low` címkét, alacsony szintû engedéllyel fog rendelkezni és nem lesz képes elérni a magasabb szinten levõ információt. Ráadásul a címke a magasabb szintû objektumok számára se fogja engedni, hogy információt közöljön vagy adjon át az alacsonyabb szintek felé.
- Az `mls/equal` címke olyan objektumok esetében ajánlott, amelyeket ki akarunk hagyni a házirend szabályozásaiból.
- Az `mls/high` címke az elérhetõ legmagasabb szintû engedélyt ábrázolja. Az ilyen címkével ellátott objektumok a rendszer összes többi objektuma felett uralommal rendelkeznek, habár az alacsonyabb szintû objektumok felé nem képesek információt közvetíteni.

Az MLS:

- Egy hierarchikus védelmi szinteket épít fel nem hierarchikus kategóriákkal.
- Szabályai rögzítettek: a felsõbb szintek olvasása és az alsóbb szintek írása egyaránt tiltott (az alanyok csak a saját vagy az alatta levõ szinteken levõ objektumokat képesek olvasni, de a felette állókat már nem. Ehhez hasonlóan az alanyok a velük egyezõ vagy a felsõbb szinteket tudják írni, de az alattuk levõket már nem).
- Megõrzi a titkokat (megakadályozza az adatok alkalmatlan közzétételét).
- Megadja mindazt az alapot, ami szükséges ahhoz, hogy az adatokat több kényességi szinten, párhuzamosan is kezelni tudjuk (anélkül, hogy titkos és bizalmas információkat szivárogtatnánk ki).

A speciális szolgáltatások és felületek beállításához az alábbi `sysctl`-változók használhatóak:

- A `security.mac.mls.enabled` engedélyezi vagy tiltja le az MLS házirend alkalmazását.
- A `security.mac.mls.ptys_equal` hatására látja el `mls/equal` címkével az összes pty(4) eszközt létrehozásuk során.
- A `security.mac.mls.revocation_enabled` használható az alacsonyabb szintre minõsített objektumok hozzáférésének megvonására.
- A `security.mac.mls.max_compartments` segítségével adható meg az objektumok által használt rekeszek szintjének maximális száma. Lényegében a rekeszek rendszerben engedélyezett maximuma.

Az MLS címkéit a setfmac(8) paranccsal tudjuk módosítani. Egy ehhez hasonló paranccsal tudunk egy objektumhoz címkét rendelni:

```
# setfmac mls/5 próba
```

A próba állomány MLS-címkéjét az alábbi paranccsal kérhetjük le:

```
# getfmac próba
```

Ezzel össze is foglaltuk az MLS házirend lehetõségeit. Az eddigiket úgy is megoldhatjuk, hogy létrehozunk egy központi házirendet az /etc könyvtárban, amelyben megadjuk az MLS házirendhez tartozó információkat, majd átadjuk a `setfmac` parancsnak. Erre a módszerre majd a házirendek bemutatása után kerül sor.

16.12.1. A kényesség megállapítása

A többszintû biztonsági házirend használatával a rendszergazda a kényes információk áramlásának irányát tudja befolyásolni. A megoldás "felfele nem lehet olvasni, lefele nem lehet írni" jellege folytán alapból mindent a legalacsonyabb szintre helyez. Így tehát kezdetben minden elérhetõ, és a rendszergazdának lassanként ebbõl az állapotból elindulva kell behangolnia az erre alapozó védelmi rendszert az információ bizalmasságának megfelelõen.

A fentebb említett három alapvetõ címke mellett a rendszergazdának valószínûleg szüksége lesz a felhasználók csoportosítására és a csoportok közti információáramlás szabályozására. A információ bizalmasságának szintjeit minden bizonnyal könnyebb szavakkal beazonosítani, például `Confidential` (bizalmas), `Secret` (titkos) vagy `Top Secret` (szigorúan bizalmas). Bizonyos helyzetekben elég csak a futó projekteknek megfelelõen kialakítani csoportokat. Az osztályozás konkrét módszerétõl függetlenül azonban mindig elmondható, hogy elõzetes tervezés nélkül sose állítsunk össze ilyen fajsúlyú házirendet.

Ezt a biztonsági modult például webes üzletek esetén érdemes használnunk, ahol egy állományszerver tárolja a cég fontos adatait és pénzügyi információit. Viszont egy két vagy három felhasználóval üzemelõ munkaállomás esetében szinte teljesen felesleges gondolkodni rajta.

16.13. A Biba MAC-modul

A modul neve: mac_biba.ko

A rendszermag konfigurációs beállítása: `options MAC_BIBA`

Rendszerindítási beállítás: `mac_biba_load="YES"`

A mac_biba(4) modul a MAC Biba elnevezésû házirendjét tölti be. Ez leginkább az MLS házirendhez hasonlít, azzal a kivétellel, hogy az információ áramoltatására vonatkozó szabályok némileg visszafelé mûködnek. Tehát míg az MLS házirend a kényes információ áramlását felfelé nem engedi, addig ez a lefelé irányuló áramlást állítja meg. Emiatt ez a szakasz tulajdonképpen mind a két házirendre érvényesül.

A Biba alkalmazása során minden alany és objektum egy "sértetlenséget" jelképezõ címkét visel. Ezek a címkék hierarchikus osztályokból, nem peidg hiearchikus összetevõkbõl származnak. Egy objektum vagy alany sértetlensége a besorolásával növekszik.

A modul a `biba/low`, `biba/equal` és `biba/high` címkéket ismeri, vagyis bõvebben:

- A `biba/low` címke tekinthetõ az alanyok és objektumok legkisebb sértetlenségének. Ha beállítjuk

egy objektumra vagy alanyra, akkor ezzel megakadályozzuk, hogy nagyobb sértetlenségû objektumokat vagy alanyokat tudjanak írni. Ettõl függetlenül azonban még képesek olvasni ezeket.

- A `biba/equal` címke használata kizárólag olyan objektumok esetében javasolt, amelyeket ki akarunk vonni a házirend alól.
- A `biba/high` címke megengedi az alacsonyabb szinteken levõ objektumokat írását, de az olvasását viszont már nem. Ezt a címkét olyan objektumra érdemes ragasztani, amelyek hatással vannak az egész rendszer sértetlenségére.

A Biba:

- Hierarchikus sértetlenségi szinteket épít fel nem hiearchikus sértetlenségi kategóriákkal kiegészítve.
- Szabályai rögzítettek: az felsõbb szintek írása és az alsóbb szintek olvasása egyaránt tilos (pontosan az MLS ellentéte). Egy alany csak a saját vagy az alatta álló szinteken szereplõ objektumokat tudja írni. Ehhez hasonló módon egy alany csak a saját vagy az afeletti szinten található objektumokat képes olvasni.
- Az adatok sértetlenségét biztosítja (megakadályozza az alkalmatlan módosításukat)
- Sértetlenségi szinteket határoz meg (szemben az MLS kényességi szintjeivel).

Az alábbi `sysctl`-változókkal vezérlhetjük a Biba házirend mûködését:

- A `security.mac.biba.enabled` használható a célrendszeren a Biba házirend engedélyezére vagy letiltására.
- A `security.mac.biba.ptys_equal` segítségével kapcsolhatjuk ki a Biba házirend alkalmazását a pty(4) eszközökön.
- A `security.mac.biba.revocation_enabled` hatására visszavonódik az objektumok hozzáférése, ha az rájuk vonatkozó címke megváltozik.

A rendszer objektumain a Biba házirendet a `setfmac` és `getfmac` paranccsal állíthatjuk be:

```
# setfmac biba/low próba
# getfmac próba
próba: biba/low
```

16.13.1. A sértetlenség megállapítása

A sértetlenség a kényességtõl eltérõen azt igyekszik szavatolni, hogy az információt illetéktelenek nem módosítják. Ez egyaránt vonatkozik az alanyok, objektumok és a kettõ között átadott adatokra. Gondoskodik róla, hogy a felhasználók csak olyan információkat változtathathassanak meg, sõt csak olyat érhessenek el, amire ténylegesen szükségük van.

A mac_biba(4) biztonsági modul megengedi a rendszergazda számára, hogy megmondja milyen állományokat és programokat láthat vagy hívhat meg a felhasználó vagy felhasználók egy csoportja, miközben biztosítja, hogy az állományok és a programok nincsenek kitéve semmilyen fenyegetésnek, és a rendszer az adott felhasználóban vagy felhasználói csoportban megbízik.

A kezdeti tervezési fázis során a rendszergazdának fel kell készülnie arra, hogy a felhasználókat osztályokra, szintekre és területekre kell osztania. A felhasználók nem csak adatokhoz, hanem programokhoz és segédprogramokhoz sem lesznek képesek hozzáférni, mind az indításuk elõtt és után. A modul aktiválás után a rendszer alapból rögtön a legmagasabb címkét kapja meg, és teljesen a rendszergazdára hárul, hogy a felhasználókhoz beállítsa a különféle osztályokat és szinteket. A fentebb leírt engedélyszintek helyett akár témák alapján is tervezhetünk. Például kizárólag csak a fejlesztõk számára engedjük meg a forráskód módosítását, a forráskód lefordítását és a többi fejlesztõeszköz használatát. Eközben a többi felhasználót felosztjuk további csoportokba, például tesztelõkre és tervezõkre, vagy meghagyjuk ezeket átlagos felhasználóknak, akik csak olvasási joggal rendelkeznek.

A megvalósított biztonsági modell természetébõl fakadóan egy kevésbé sértetlenebb alany nem írhatja a sokkal sértetlenebb alanyokat, a sokkal sértetlenebb alanyok pedig nem érhetik el vagy olvashatják a kevésbé sértetlen objektumokat. A lehetõ legkisebb osztályú címke beállításával gyakorlatilag elérhetetlenné teszük az alanyok számára. A modult valószínûleg egy korlátozott webszerver, fejlesztõi- és tesztgépek vagy forráskód tárolására szánt környezetben érdemes bevetni. Annál esélytelenebb a használata viszont egy munkaállomás, útválasztó vagy hálózati tûzfal esetében.

16.14. A LOMAC MAC-modul

A modul neve: mac_lomac.ko

A rendszermag konfigurációs beállítása: `options MAC_LOMAC`

Rendszerindítás beállítás: `mac_lomac_load="YES"`

Eltérõen a MAC Biba házirendjétõl, a mac_lomac(4) egyedül csak azután engedi elérni az kevésbé sértetlenebb objektumokat, miután csökkentjük a sértetlenség szintjét és ezzel betartjuk a sértetlenségre vonatkozó szabályokat.

A gyenge vízjeles sértetlenségi házirend MAC alapú változatát nem szabad összetéveszteni a korábbi lomac(4) implementációval, amely majdnem ugyanúgy mûködik, mint a Biba, azzal az a kivétellel, hogy a lebegõ címkékkel támogatjuk az alanyok lefokozását egy kisegítõ osztály rekeszén keresztül. Ez a másodlagos rekesz `[kisegítõ_osztály]` alakú. Tehát amikor egy kisegítõ osztállyal adjuk meg a lomac házirendet, valahogy így néz ki: `lomac/10[2]`, ahol a kettes (2) szám ez a kisegítésre használt osztály.

A MAC LOMAC házirendje az összes rendszerszintû objektum esetében jelenlevõ sértetlenségi címkézésen alapszik, megengedve az alanyok számára, hogy az kevésbé sértetlen objektumokat olvasni tudják, majd a címke leminõsítésével az alany meg tudja akadályozni a sokkal sértetlenebbnek ítélt objektumok jövõbeni írását. Ez az a fentebb tárgyalt `[kisegítõ_osztály]` opció, ezért ez a modul a Bibáénál több kompatibilitást és kevesebb kezdeti beállítást igényel.

16.14.1. Példák

Hasonlóan a Biba és MLS házirendeknél megszokottakhoz, a `setfmac` és `setpmac` segédprogramok használhatóak a címkék hozzárendeléséhez:

```
# setfmac /usr/home/trhodes lomac/high[low]
# getfmac /usr/home/trhodes lomac/high[low]
```

Itt a kisegítõ osztály a `low`. Ezt csak a LOMAC MAC-házirendnél adhatjuk meg.

16.15. A Nagios elzárása a MAC rendszerrel

A most következõ bemutatóban a MAC moduljainak és a megfelelõen beállított házirendek használatával fogunk kialakítani egy biztonságos környezetet. Ne feledjük azonban, hogy ez csupán egy ártatlan próba és nem pedig a mindenki biztonsági aggályait kielégítõ legvégsõ megoldás. Ha egy házirendet vakon építünk fel és nem értjük meg a mûködését, az soha nem válik hasznunkra, és egy éles helyzetben katasztrofális hatással járhat.

A folyamat megkezdése elõtt be kell állítanunk a `multilabel` opciót mindegyik állományrendszerre, a fejezet elején leírtaknak megfelelõen. Ha ezt a lépést kihagyjuk, akkor hibákat kapunk. Továbbá még az elõkészület részeként ne felejtsünk el gondoskodni a net-mngt/nagios-plugins, net-mngt/nagios és www/apache13 portok telepítésérõl, beállításáról és megfelelõ mûködésérõl sem.

16.15.1. A nem megbízható felhasználók osztályának létrehozása

Az eljárást kezdjük az alábbi (insecure) felhasználói osztály hozzáadásával az /etc/login.conf állományban:

```
insecure:\
:welcome=/etc/motd:\
:setenv=MAIL=/var/mail/$,BLOCKSIZE=K:\
:path=~/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
:manpath=/usr/shared/man /usr/local/man:\
:nologin=/usr/sbin/nologin:\
:cputime=1h30m:\
:datasize=8M:\
:vmemoryuse=100M:\
:stacksize=2M:\
:memorylocked=4M:\
:memoryuse=8M:\
:filesize=8M:\
:coredumpsize=8M:\
:openfiles=24:\
:maxproc=32:\
:priority=0:\
:requirehome:\
:passwordtime=91d:\
:umask=022:\
:ignoretime@:\
:label=biba/10(10-10):
```

Valamint egészítsük ki az alapértelmezett (default) felhasználói osztályt a következõ sorral:

```
:label=biba/high:
```

Ahogy ezzel elkészültünk, az hozzá tartozó adatbázis újbóli legyártásához a következõ parancsot kell kiadnunk:

```
# cap_mkdb /etc/login.conf
```

16.15.2. A rendszerindítással kapcsolatos beállítások

Még ne indítsuk újra a számítógépet, csupán a szükséges modulok betöltéséhez bõvítsük ki a /boot/loader.conf állományt az alábbi sorokkal:

```
mac_biba_load="YES"
mac_seeotheruids_load="YES"
```

16.15.3. A felhasználók beállítása

Soroljuk be a `root` felhasználót a `default` osztályba:

```
# pw usermod root -L default
```

Az összes `root` felhasználón kívüli hozzáférésnek vagy rendszerfelhasználónak most kelleni fog egy bejelentkezési osztály. A bejelentkezési osztályra egyébként is szükség lesz, mert ennek hiányában a felhasználók még az olyan egyszerû parancsokat sem tudják kiadni, mint például a vi(1). A következõ `sh` szkript nekünk erre pontosan megfelel:

```
# for x in `awk -F: '($3 >= 1001) && ($3 != 65534) { print $1 }' \
    /etc/passwd`; do pw usermod $x -L default; done;
```

Helyezzük át a `nagios` és `www` felhasználókat az insecure osztályba:

```
# pw usermod nagios -L insecure
```

```
# pw usermod www -L insecure
```

16.15.4. A contexts állomány létrehozása

Most csinálnunk kell egy contexts állományt. Ebben példában az /etc/policy.contexts állományt használjuk.

```
# Ez a rendszer alapértelmezett BIBA házirendje.
```

```
# Rendszer:
/var/run                        biba/equal
/var/run/*                      biba/equal

/dev                            biba/equal
/dev/*                          biba/equal

/var                biba/equal
/var/spool                      biba/equal
/var/spool/*                    biba/equal

/var/log                        biba/equal
/var/log/*                      biba/equal

/tmp                biba/equal
/tmp/*              biba/equal
/var/tmp            biba/equal
/var/tmp/*          biba/equal

/var/spool/mqueue       biba/equal
/var/spool/clientmqueue     biba/equal

# Nagios:
/usr/local/etc/nagios
/usr/local/etc/nagios/*         biba/10

/var/spool/nagios               biba/10
/var/spool/nagios/*             biba/10

# Apache:
/usr/local/etc/apache           biba/10
/usr/local/etc/apache/*         biba/10
```

Ezzel a házirenddel az információ áramlását szabályozzuk. Ebben a konkrét konfigurációban a felhasználók, a `root` és társai, nem férhetnek hozzá a Nagioshoz. A Nagios beállításait tároló állományok és a neve alatt futó programok így teljesen különválnak vagyis elzáródnak a rendszer többi részétõl.

Ez az iménti állomány a következõ parancs hatására kerül be a rendszerünkbe:

```
# setfsmac -ef /etc/policy.contexts /
# setfsmac -ef /etc/policy.contexts /
```

A fenti állományrendszer felépítése a környezettõl függõen eltérhet, habár ezt minden egyes állományrendszeren le kell futtatni.

Az /etc/mac.conf állományt törzsét a következõképpen kell még átírnunk:

```
default_labels file ?biba
default_labels ifnet ?biba
default_labels process ?biba
default_labels socket ?biba
```

16.15.5. A hálózat engedélyezése

Tegyük hozzá a következõ sort az /boot/loader.conf állományhoz:

```
security.mac.biba.trust_all_interfaces=1
```

Ezt az alábbi beállítást pedig szúrjuk be az rc.conf állományba a hálózati kártya konfigurációjához. Amennyiben az internetet DHCP segítségével érjük el, ezt a beállítást manuálisan kell megtenni minden rendszerindítás alkalmával:

```
maclabel biba/equal
```

16.15.6. A konfiguráció kipróbálása

Gondoskodjunk róla, hogy a webszerver és a Nagios nem fog elindulni a rendszer indításakor, majd indítsuk újra a gépet. Ezenkívül még ellenõrizzük, hogy a `root` ne tudjon hozzáférni a Nagios beállításait tartalmazó könyvtárhoz. Ha a `root` képes kiadni egy ls(1) parancsot a /var/spool/nagios könyvtárra, akkor valamit elronthattunk. Normális esetben egy `permission denied` üzenetet kell kapnunk.

Ha minden jónak tûnik, akkor a Nagios, Apache és Sendmail most már elindítható a biztonsági házirend szabályozásai szerint. Ezt a következõ parancsokkal tehetjük meg:

```
# cd /etc/mail && make stop && \
setpmac biba/equal make start && setpmac biba/10\(10-10\) apachectl start && \
setpmac biba/10\(10-10\) /usr/local/etc/rc.d/nagios.sh forcestart
```

Kétszer is ellenõrizzük, hogy minden a megfelelõ módon viselkedik-e. Ha valamilyen furcsaságot tapasztalunk, akkor nézzük át a naplókat vagy a hibaüzeneteket. A sysctl(8) használatával tiltsuk le a mac_biba(4) biztonsági modult és próbáljunk meg mindent a szokott módon újraindítani.

A `root` felhasználó különösebb aggodalom nélkül képes megváltoztatni a biztonsági rend betartatását és átírni a konfigurációs állományokat. Egy frissen indított parancsértelmezõ számára ezzel a paranccsal tudjuk csökkenteni a biztonsági besorolást:

```
# setpmac biba/10 csh
```

> Ennek kivédésére a felhasználókat a login.conf(5) beállításaival le kell korlátozni. Ha a setpmac(8) megpróbál a rekesz határain túl futtatni egy parancsot, akkor hibát ad vissza és a parancs nem fut le. Ebben az esetben a `root` felhasználót tegyük a `biba/high(high-high)` értékek közé.

16.16. A felhasználók korlátozása

Ebben a példában egy viszonylag kicsi, nagyjából mindössze ötven felhasználós, adattárolásra használatos rendszert veszünk alapul. A felhasználók rendelkezhetnek bizonyos bejelentkezési tulajdonságokkal, és nem csak adatokat tudnak tárolni, hanem az erõforrásokhoz is hozzá tudnak férni.

Itt most a mac_bsdextended(4) és a mac_seeotheruids(4) modulokat vetjük be együttesen, és nem csak a rendszer objektumainak elérését tudjuk megakadályozni, hanem az egyes felhasználók futó programjait is elrejtjük.

A mûveletet kezdjük azzal, hogy a /boot/loader.conf állományt kibõvítjük a következõ módon:

```
mac_seeotheruids_load="YES"
```

A mac_bsdextended(4) biztonsági modul az alábbi rc.conf-változóval hozható mûködésbe:

```
ugidfw_enable="YES"
```

A hozzá tartozó alapértelmezett szabálykészlet az /etc/rc.bsdextended állományban tárolódik, amely pedig a rendszer indítása során töltõdik be. Ezeket némileg módosítanunk kell majd. Mivel a példában szereplõ számítógép csak a felhasználók kiszolgálását hivatott ellátni, az utolsó kettõ kivételével mindent hagyhatunk megjegyzésben. Így kikényszerítjük felhasználók által birtokolt rendszerobjektumok alapértelmezés szerinti betöltését.

Vegyük fel a szükséges felhasználókat a számítógépre és indítsuk újra. Tesztelési célból próbáljunk meg különbözõ felhasználókként bejelentkezni két konzolon. Futassuk le a `ps aux` parancsot, és így meg tudjuk figyelni, hogy mennyire látjuk a többi felhasználót. Amikor megpróbáljuk kiadni a ls(1) parancsot a többiek felhasználói könyvtáraira, akkor hibát kell kapnunk.

Ne próbálgassunk a `root` felhasználóval, hacsak a megfelelõ `sysctl` változókban be nem állítottuk az õ hozzáférésének blokkolását is.

> Amikor felveszük egy felhasználót a rendszerbe, a hozzá tartozó mac_bsdextended(4) szabály nem fog szerepelni a szabályrendszerben. A szabályrendszer gyors frissítését úgy tudjuk megoldani, ha a kldunload(8) használatával egyszerûen eltávolítjuk a biztonsági modult a memóriából és újratöltjük a kldload(8) paranccsal.

16.17. A hibák elhárítása a MAC rendszerben

A fejlesztés fázisában bizonyos normál konfigurációval rendelkezõ felhasználók gondokat jeleztek. Ezeket foglaljuk most itt össze:

16.17.1. A `multilabel` beállítás nem adható meg a / állományrendszerre

A `multilabel` beállítás nem marad meg a rendszerindító (/) partíciómon!

A tapasztalatok szerint körülbelül minden ötvenedik felhasználó szembesül ezzel a problémával, és mi is találkozunk vele a kezdeti konfigurációk kialakítása során. Ennek az úgynevezett "hibának" a behatóbb tanulmányozása során arra jutottunk, hogy ez többnyire vagy a hibás dokumentálásból vagy a dokumentáció félreértelmezéséből ered. Független attól, hogy ez mitõl is következett be, a következõ lépések megtételével orvosolhatjuk:

1. Nyissuk meg az /etc/fstab állományt és adjuk meg a rendszerindító partíciónak az `ro`, vagyis az írásvédett (read-only) beállítást.
2. Indítsuk újra a gépet egyfelhasználós módban.
3. A `tunefs -l enable` parancsot futtassuk le a / állományrendszeren.
4. Indítsuk újra a rendszert normál módban.
5. Adjuk ki a `mount -urw/` parancsot, majd az /etc/fstab állományban írjuk át a `ro` beállítást az `rw` értékre és megint indítsuk újra a rendszert.
6. Alaposan nézzük át a `mount` parancs kimenetét és gyõzödjünk meg róla, hogy a `multilabel` opció valóban beállítódott a rendszerindító állományrendszerre.

16.17.2. A MAC után nem lehet indítani az X11 szervert

Nem indul az X, miután MAC-kel kialakítottunk egy biztonságos környezetet!

Ezt vagy a MAC `partition` házirendje okozza, vagy az egyik címkékeket használó házirend helytelen beállítása. A következõ módon deríthetjük ki az okát:

1. Figyelmesen olvassuk el a hibaüzenetet: ha a felhasználó az `insecure` osztály tagja, akkor a `partition` házirend lesz a bûnös. Próbáljuk meg a felhasználót visszatenni a `default` osztályba és a `cap_mkdb` paranccsal újragenerálni az adatbázist. Ha ez nem segít a problémán, akkor haladjunk tovább.
2. Alaposan ellenõrizzük a címkékhez tartozó házirendeket. Vizsgáljuk meg, hogy a kérdeses felhasználó esetében a házirendet és az X11 alkalmazást, valamint a /dev eszközöket tényleg jól állítottuk be.
3. Ha az iméntiek egyik sem oldja meg gondunkat, küldjük el a hibaüzenetet és a környezetünk rövid leírását a a TrustedBSD honlapjáról elérhetõ TrustedBSD levelezési lista vagy a FreeBSD general questions levelezési lista címére.

16.17.3. Hiba: _secure_path(3) cannot stat .login_conf

Amikor a rendszerben megpróbálok a `root` felhasználóról átváltani egy másik felhasználóra, a `_secure_path: unable to state .login_conf` hibaüzenet jelenik meg.

Ez az üzenet általában akkor látható, amikor a felhasználó nagyobb értékû címkével rendelkezik annál, mint akivé válni akar. Például vegyük a `joska` nevû felhasználót a rendszerben, aki az alap `biba/low` címkével rendelkezik. A `root` felhasználó, akinek `biba/high` címkéje van, nem láthatja `joska` felhasználói könyvtárát. Ez attól függetlenül megtörténik, hogy a `root` a `su` paranccsal váltott át a `joska` nevû felhasználóra vagy sem. Egy ilyen helyzetben a Biba sértetlenségi modellje nem fogja engedni a `root` felhasználóra számára, hogy láthassa a kevésbé sértetlen objektumokat.

16.17.4. A `root` felhasználó nem megy!

A rendszer normál vagy egyfelhasználós módban sem ismeri fel a `root` felhasználót. A `whoami` parancs 0 (nullát) ad vissza és a `su` parancs pedig annyit mond: `who are you?` (`ki vagy?`). Mi történhetett?

Ez csak olyankor történhet, ha a címkézési házirendet nem engedélyezzük, vagy a sysctl(8) használatával, vagy pedig a modul eltávolításával. Ha a házirendet letiltjuk vagy ideiglenesen letiltódik, akkor a bejelentkezési tulajdonságokat tároló adatbázist a `label` beállítás eltávolításával kell újrakonfigurálni. A login.conf állományból ne felejtsük el kivenni az összes `label` beállítást és a `cap_mkdb` paranccsal újragenerálni az adatbázist.

Ilyen akkor is elõfordulhat, amikor a házirend valamilyen módon korlátozza a master.passwd állomány vagy adatbázis elérhetõségét. Ezt általában az okozza, hogy a rendszergazda az állományt olyan címke alatt módosítja, amely ütközik a rendszerben alkalmazott általános házirenddel. Ebben az esetekben a rendszer próbálja meg beolvasni a felhasználók adatait, azonban mivel közben az állomány új címkét örökölt, nem fér hozzá. Ha a sysctl(8) paranccsal letiltjuk a házirendet, minden vissza fog térni a rendes kerékvágásba.

Chapter 17. Biztonsági események vizsgálata

17.1. Áttekintés

A FreeBSD támogatja a biztonsági események aprólékos vizsgálatát. Ezzel egy megbízható, részletes és jól konfigurálható naplózási rendszert nyújtanak a rendszerben található biztonságot igénylõ események széles köréhez, beleértve a bejelentkezéseket, a konfigurációs állományokban bekövetkezõ változásokat, állomány- és hálózati hozzáféréseket. Az így létrehozott naplóbejegyzések felbecsülhetetlen értékûnek bizonyulhatnak egy élõ rendszer felügyelete során, vagy egy hálózati támadás észleléséhez, esetleg egy összeomlás okainak kielemezéséhez. A FreeBSD ehhez a Sun™ által kifejlesztett BSM technológia API-ját és állományformátumát valósítja meg, és így képes együttmûködni a Sun™ Solaris™ valamint az Apple® Mac OS® X bizonsági rendszereivel egyaránt.

Ebben a fejezetben a biztonsági események vizsgálatának telepítéséhez és beállításához szükséges ismeretek tekintjük át. Ennek keretében szó esik a vizsgálati házirendekrõl, valamint mutatunk egy példát a vizsgálatok beállítására.

A fejezet elolvasása során megismerjük:

- mit jelent az események vizsgálata és hogyan mûködik;
- hogyan kell beállítani az események vizsgálatát FreeBSD-n a különbözõ felhasználók és programok esetén;
- hogyan értelmezzük a vizsgálati nyomokat a vizsgálatot szûkítõ és -elemzõ segédprogramok segítségével.

A fejezet elolvasásához ajánlott:

- alapvetõ UNIX®-os és FreeBSD-s ismeretek (A UNIX alapjai);
- a rendszermag konfigurálásával és fordításával kapcsolatos tudnivalók alapszintû ismerete (A FreeBSD rendszermag testreszabása);
- az informatikai biztonság alapfogalmainak és annak a FreeBSD-re vonatkozó részleteinek minimális ismerete (Biztonság).

Az események vizsgálatával kapcsolatos ismert korlátozások: nem mindegyik biztonságot érintõ esemény vizsgálható, mint például az egyes bejelentkezési típusok, mivel azok nem megfelelõen hitelesítik a belépõ felhasználókat. Ilyenek például az X11-alapú felületek és az egyéb, erre a célra alkalmas, más által fejlesztett démonok.

A biztonsági események vizsgálata során a rendszer képes nagyon részletes naplókat készíteni az érintett tevékenységekrõl. Így egy kellõen forgalmas rendszeren az állománymozgások alapos nyomonkövetése bizonyos konfigurációkon akár gigabyte-okat is kitehet hetente. A rendszergazdáknak ezért mindig javasolt számolniuk a nagy forgalmú események biztonsági vizsgálatának tárigényével. Például, emiatt érdemes lehet egy egész állományrendszert szánni

erre a feladatra a /var/audit könyvtárban, és így a többi állományrendszer nem látja kárát, ha véletlenül betelne ez a terület.

17.2. A fejezet fontosabb fogalmai

A fejezet elolvasása elõtt meg kell ismernünk néhány fontos alapfogalmat:

- *esemény:* Vizsgálható eseménynek azt az eseményt nevezzük, amely egy vizsgálati alrendszerben naplózható. Biztonsági események lehetnek például: egy állomány létrehozása, egy hálózati kapcsolat felépítése, vagy egy felhasználó bejelentkezése. Egy esemény "jellegzetes", ha visszakövethetõ valamelyik hitelesített felhasználóhoz, vagy "nem jellegzetes", ha ez nem lehetséges. Nem jellegzetes esemény lehet minden olyan esemény, amely egy bejelentkezési folyamat hitelesítési lépése elõtt történik, például egy belépési kísérlet hibás jelszóval.
- *osztály:* Eseményosztálynak az összefüggõ események névvel ellátott halmazát tekintjük, és szûrési feltételekben használjuk ezeket. Általában alkalmazott osztályok: "file creation" (fc, állománylétrehozás), "exec" (ex, programindítás), és "login_logout" (lo, ki- és bejelentkezés).
- *rekord:* Rekordnak nevezzük a biztonsági eseményeket leíró biztonsági naplóbejegyzéseket. A rekordok tartalmazhatják a feljegyzett esemény típusát, az eseményt kiváltó tevékenységet (felhasználót), a dátumot és az idõt, tetszõleges objektum vagy paraméter értékét, feltételek teljesülését vagy meghiúsulását.
- *nyom:* Vizsgálati nyomnak vagy naplóállománynak nevezzük a különféle biztonsági eseményeket leíró vizsgálati rekordok sorozatát. A nyomok többnyire nagyjából az események bekövetkezése szerinti idõrendben következnek. Csak és kizárólag az erre felhatalmazott programok hozhatnak létre rekordokat a vizsgálati nyomban.
- *szûrési feltétel:* Szûrési feltételnek nevezünk egy olyan karakterláncot, amelyet események szûrésére használunk, és módosítókat valamint eseményosztályok neveit tartalmazza.
- *elõválogatás:* Elõválogatásnak nevezzük a folyamatot, amelynek során a rendszer beazonosítja azokat az eseményeket, amelyek a rendszergazda számára fontosak. Ezáltal elkerülhetjük olyan vizsgálati rekordok generálását, amelyek számunkra érdektelen eseményekrõl számolnak be. Az elõválogatás szûrési feltételek sorát használja az adott felhasználókhoz tartozó adott biztonsági események vizsgálatának beállításához, akárcsak a hitelesített és a nem hitelesített programokat értintõ globális beállítások meghatározásához.
- *leszûkítés:* Leszûkítésnek nevezzük a folyamatot, amelynek során a már meglevõ biztonsági rekordokból válogatunk le tárolásra, nyomtatásra vagy elemzésre. Hasonlóan ez a folyamat, ahol a szükségtelen rekordokat eltávolítjuk a vizsgálatai nyomból. A leszûkítés segítségével a rendszergazdák a vizsgálati adatok eltárolására alakíthatnak ki házirendet. Például a részletesebb vizsgálati nyomokat érdemes egy hónapig megtartani, ennek lejártával viszont már inkább ajánlott leszûkíteni ezeket és archiválásra csak a bejelentkezési információkat megtartani.

17.3. A vizsgálat támogatásának telepítése

A eseményvizsgálathoz szükséges felhasználói programok a FreeBSD alaprendszer részét képezik. Az eseményvizsgálat támogatása alapértelmezés szerint megtalálható a rendszermagban, azonban

egy saját rendszermag esetén már külön be kell kapcsolnunk a megfelelõ támogatást, mégpedig a rendszermag konfigurációs állományában az alábbi sor hozzáadásával:

```
options AUDIT
```

Fordítsuk és telepítsük újra a rendszermagot az A FreeBSD rendszermag testreszabásaben ismertetett folyamat szerint.

Ahogy a rendszermagot a bekapcsolt eseményvizsgálati támogatással sikerült lefordítanunk és telepítenünk, valamint a rendszerünk is újraindult, indítsuk el a vizsgáló démont a következõ sor hozzáadásával az rc.conf(5) állományban:

```
auditd_enable="YES"
```

A vizsgálatot innentõl ténylegesen egy ismételt újraindítással vagy pedig az elõbb említett démon manuális elindításával aktiválhatjuk:

```
/etc/rc.d/auditd start
```

17.4. A vizsgálat beállítása

A vizsgálatok beállításához szükséges összes konfigurációs állomány a /etc/security könyvtárban található. A következõ állományok vannak itt a démon indítása elõtt:

- audit_class - a vizsgálati osztályok definícióit tartalmazza.
- audit_control - a vizsgálati alrendszer különbözõ területeit vezérli, többek közt az alapértelmezett vizsgálati osztályokat, az vizsgálati adatok tárhelyén fenntartandó minimális lemezterületet, a vizsgálati nyom maximális méretét, stb.
- audit_event - a rendszerben jelenlevõ vizsgálati események szöveges megnevezése és leírása, valamint a lista, hogy melyikük mely osztályban található.
- audit_user - felhasználónként változó vizsgálati elvárások, kombinálva a bejelentkezéskor érvényes globálisan alapértelmezett beállításokkal.
- audit_warn - az auditd által használt testreszabható shell szkript, aminek segítségével a szélsõséges helyzetekben figyelmeztetõ üzeneteket tudunk generálni, mint például amikor a rekordok számára fenntartott hely hamarosan elfogy, vagy amikor a nyomokat tartalmazó állományt archiváltuk.

Az eseményvizsgálat konfigurációs állományait alapos körültekintés mellett szabad szerkeszteni és karbantartani, mivel a bennük keletkezõ hibák az események helytelen naplózását eredményezhetik.

17.4.1. Eseményszûrési feltételek

Az eseményvizsgálati beállítások során számtalan helyen felbukkanak a vizsgálni kívánt eseményeket meghatározó szûrési feltételek. Ezen feltételek eseményosztályok felsorolását tartalmazzák, mindegyiküket egy módosító vezeti be, ezzel jelezve, hogy az adott eseményosztályba tartozó rekordokat tartsuk meg vagy vessük el. Esetleg utalhatnak arra is, hogy vagy csak a sikerességet jelzõ rekordokat, vagy csak a sikertelenséget jelzõ rekordokat szûrjük ki. A szûrési feltételek balról jobbra értékelõdnek ki, és két kifejezés összefûzéssel kombinálható.

A most következõ lista tartalmazza a audit_class állományban található alapértelmezett eseményvizsgálati osztályokat:

- `all` - *all (mind)* - Minden eseményosztályra vonatkozik.
- `ad` - *administrive (adminisztrációs)* - olyan adminisztrációs tevékenységek, amelyek egyben az egész rendszeren végrehajtódnak.
- `ap` - *application (alkalmazás)* - az alkalmazások által meghatározott tevékenység.
- `cl` - *file close (állomány lezárása)* - a `close` rendszerhívás meghívásának vizsgálata.
- `ex` - *exec (programindítás)* - egy program indításának vizsgálata. A parancssorban átadott paraméterek és a környezeti változók vizsgálatát az audit_control(5) vezérli a `policy` beállításhoz tartozó `argv` és `envv` paraméterek segítségével.
- `fa` - *file attribute access (állományjellemzõk hozzáférése)* - a rendszerbeli objektumok jellemzõinek hozzáférésnek vizsgálata, mint például a stat(1), pathconf(2) és ehhez hasonló események.
- `fc` - *file create (állomány létrehozása)* - állományt eredményezõ események vizsgálata.
- `fd` - *file delete (állomány törlése)* - állományt törlõ események vizsgálata.
- `fm` - *file attribute modify (állományjellemzõk módosítása)* - állományok jellemzõit megváltoztató események vizsgálata, mint például a chown(8), chflags(1), flock(2), stb.
- `fr` - *file read (állományolvasás)* - állományok megnyitásával olvasásra, olvasásával, stb. kapcsolatos események vizsgálata.
- `fw` - *file write (állományírás)* - állományok megnyitásával írásra, írásával, módosításával, stb. kapcsolatos események vizsgálata.
- `io` - *ioctl* - az ioctl(2) rendszerhívást használó események vizsgálata.
- `ip` - *ipc* - a folyamatok közti kommunikáció különféle formáinak, beleértve a POSIX csövek és System V IPC mûveleteinek vizsgálata.
- `lo` - *login_logout (ki- és bejelentkezés)* - a rendszerben megjelenõ login(1) és logout(1) események vizsgálata.
- `na` - *non attributable (nem jellegzetes)* - a nem jellegzetes események vizsgálata.
- `no` - *invalid class (érvénytelen osztály)* - egyetlen biztonsági eseményt sem tartalmaz.
- `nt` - *network (hálózat)* - a hálózathoz tartozó események vizsgálata, mint például a connect(2) és az accept(2).
- `ot` - *other (egyéb)* - más egyéb események vizsgálata.

- `pc` - *process (folyamat)* - a folyamatokkal kapcsolatos mûveletek, mint például az exec(3) és az exit(3) vizsgálata.

Az imént felsorolt eseményosztályok az audit_class és az audit_event állományok módosításával igény szerint testreszabhatóak.

A listában szereplõ minden egyes eseményosztályhoz tartozik még egy módosító is, amely jelzi, hogy a sikeres vagy a sikertelen mûveleteket kell-e szûrnünk, valamint hogy a bejegyzés az adott típust vagy osztályt hozzáadja vagy elveszi az adott szûrésbõl.

- (üres) az adott típusból mind a sikereseket és mind a sikerteleneket feljegyzi.
- `+` az eseményosztályba tartozó sikeres eseményeket vizsgálja csak.
- `-` az eseményosztályba tartozó sikertelen eseményeket vizsgálja csak.
- `^` az eseményosztályból sem a sikereseket, sem pedig a sikerteleneket nem vizsgálja.
- `^+` az eseményosztályból nem vizsgálja a sikeres eseményeket.
- `^-` az eseményosztályból nem vizsgálja a sikertelen eseményeket.

Az alábbi példa egy olyan szûrési feltételt mutat be, amely a ki- és bejelentkezések közül megadja a sikereset és a sikerteleneket, viszont a programindítások közül csak a sikereseket:

```
lo,+ex
```

17.4.2. A konfigurációs állományok

A vizsgálati rendszer beállításához az esetek túlnyomó részében a rendszergazdáknak csupán két állományt kell módosítaniuk: ezek az audit_control és az audit_user. Az elõbbi felelõs a rendszerszintû vizsgálati jellemzõkért és házirendekért, míg az utóbbi az igények felhasználókénti finomhangolásához használható.

Az audit_control állomány

Az audit_control állomány határozza meg a vizsgálati alrendszer alapértelmezéseit. Ezt az állományt megnyitva a következõket láthatjuk:

```
dir:/var/audit
flags:lo
minfree:20
naflags:lo
policy:cnt
filesz:0
```

A `dir` opciót használjuk a vizsgálati naplók tárolására szolgáló egy vagy több könyvtár megadására. Ha egynél több könyvtárra vonatkozó bejegyzés található az állományban, akkor azok a megadás sorrendjében kerülnek feltöltésre. Nagyon gyakori az a beállítás, ahol a vizsgálati naplókat egy erre a célra külön kialakított állományrendszeren tárolják, megelõzve ezzel az állományrendszer

betelésekor keletkezõ problémákat a többi alrendszerben.

A `flags` mezõ egy rendszerszintû alapértelmezett elõválogatási maszkot határoz meg a jellegzetes események számára. A fenti példában a sikeres és sikertelen ki- és bejelentkezéseket mindegyik felhasználó esetén vizsgáljuk.

A `minfree` opció megszabja a vizsgálati nyom tárolására szánt állományrendszeren a minimális szabad helyet, a teljes kapacitás százalékában. Amint ezt a küszöböt túllépjük, egy figyelmeztetés fog generálódni. A fenti példa a minimálisan szükséges rendelkezésre álló helyet húsz százalékra állítja.

A `naflags` opció megadja azokat az eseményosztályokat, amelyeket vizsgálni kell a nem jellegzetes események, mind például a bejelentkezési folyamatok vagy rendszerdémonok esetén.

A `policy` opció a vizsgálat különbözõ szempontjait irányító házirendbeli beállítások vesszõvel elválasztott listáját tartalmazza. Az alapértelmezett `cnt` beállítás azt adja meg, hogy a rendszer a felmerülõ vizsgálati hibák ellenére is folytassa tovább a mûködését (erõsen javasolt a használata). A másik gyakorta alkalmazott beállítás az `argv`, amellyel a rendszer a parancsvégrehajtás részeként az execve(2) rendszerhívás parancssori paramétereit is megvizsgálja.

A `filesz` opció határozza meg a vizsgálati nyom automatikus szétvágása és archiválása elõtti maximális méretét, byte-ban. Az alapértelmezett értéke a 0, amely kikapcsolja ezt az archiválást. Ha az itt megadott állományméret nem nulla és a minimálisan elvárt 512 KB alatt van, akkor a rendszer figyelmen kívül hagyja és errõl egy figyelmeztetést ad.

Az audit_user állomány

Az audit_user állomány lehetõvé teszi a rendszergazda számára, hogy az egyes felhasználók számára további vizsgálati szigorításokat határozzon meg. Minden sor egy-egy felhasználó vizsgálatának pontosítását adja meg két mezõ segítségével: az elsõ közülük az `alwaysaudit` mezõ, mely felsorolja azokat az eseményeket, amelyeket minden esetben vizsgáni kell az adott felhasználó esetén, valamint a második a `neveraudit` mezõ, mely az adott felhasználó esetén a nem vizsgálandó eseményeket adja meg.

A most következõ audit_user példában vizsgáljuk a `root` felhasználó ki- és bejelentkezéseit és sikeres programindításait, valamint a `www` felhasználó állománylétrehozásait és sikeres programindításait. Ha a korábban bemutatott audit_control példával együtt használjuk, akkor észrevehetjük, hogy a `lo` bejegyzés a `root` felhasználó esetén redundáns, illetve ilyenkor a ki/bejelentkezést a `www` felhasználó esetén is vizsgáljuk.

```
root:lo,+ex:no
www:fc,+ex:no
```

17.5. A vizsgálati alrendszer használata

17.5.1. A vizsgálati nyomok megtekintése

A vizsgálati nyomok a BSM bináris formátumban tárolódnak, ezért a tartalmának konvertálásához

és módosításához külön segédprogramokra van szükség. A praudit(1) parancs a nyomállományokat egyszerû szöveges formátumra alakítja, az auditreduce(1) parancs pedig a nyomok elemzéséhez, archiválásához vagy nyomtatásához szükséges leszûkítéseket végzi el. Az `auditreduce` a szûrési feltételek paramétereinek széles skáláját kezeli, beleértve az eseménytípusokat, -osztályokat, felhasználókat, események dátumát vagy idõpontját, állományok elérési útvonalát vagy az általuk érintett objektumokat.

Például a `praudit` segédprogram képes kilistázni szövegesen egy adott vizsgálati napló teljes tartalmát:

```
# praudit /var/audit/AUDITFILE
```

ahol az AUDITFILE a kiírandó vizsgálati napló.

A vizsgálati nyomok tokenekbõl összeállított vizsgálati rekordok, amelyeket a `praudit` egymás után soronként megjelenít. Minden token adott típusú, például a `header` egy vizsgálati rekord fejlécét tartalmazza, vagy a `path`, amely a névfeloldásból származó elérési utat tartalmaz. A következõ példa egy `execve` eseményt mutat be:

```
header,133,10,execve(2),0,Mon Sep 25 15:58:03 2006, + 384 msec
exec arg,finger,doug
path,/usr/bin/finger
attribute,555,root,wheel,90,24918,104944
subject,robert,root,wheel,root,wheel,38439,38032,42086,128.232.9.100
return,success,0
trailer,133
```

Ez a vizsgálat egy sikeres `execve` hívást rögzít, ahol a `finger doug` parancs futott le. A paramétereket tartalmazó token magában foglalja a shell által a rendszermag felé jelzett parancsot és annak paraméterét egyaránt. A `path` token tárolja a végrehajtott állomány rendszermag által feloldott elérési útját. A `attribute` token errõl a binárisról ad további információkat, különösen az állomány módjáról, amely segít megállapítani, hogy az adott alkalmazásnál be volt-e állítva a setuid bit. A `subject` token leírja az érintett folyamatot és rendre megjegyzi a vizsgált felhasználó azonosítóját, az aktuálisan érvényben levõ felhasználó és csoport azonosítóját, a valós felhasználói és csoport azonosítót, a folyamat azonosítóját, a munkamenet azonosítóját, a port azonosítóját és a bejelentkezéshez használt hálózati címet. Vegyük észre, hogy a vizsgált felhasználó azonosítója és a valódi azonosítója eltér egymástól: a `robert` nevû felhasználó a `root` accountjára váltott a parancs futattása elõtt, de az eredetileg hitelesített felhasználójaként lett vizsgálva. Végezetül a `return` token jelzi a sikeres végrehajtást, és a `trailer` pedig zárja a rekordot.

17.5.2. A vizsgálati nyomok leszûkítése

Mivel a vizsgálatokhoz tartozó naplók akár egészen nagyok is lehetnek, ezért a rendszergazdának minden bizonnyal szüksége lehet a számára fontos, például egy adott felhasználóhoz tartozó rekordok kiválogatására:

```
# auditreduce -u trhodes /var/audit/AUDITFILE | praudit
```

Ezzel ki tudjuk szûrni a `trhodes` nevû felhasználóhoz tartozó összes vizsgálati rekordot az AUDITFILE állományból.

17.5.3. A naplók megtekintéséhez szükséges jogok továbbadása

Az `audit` csoport tagjai olvashatják a /var/audit könyvtárban található vizsgálati nyomokat. Alapértelmezés szerint ez a csoport üres, ezért csak a `root` képes ekkor vizsgálni a nyomokat. A többi felhasználó számára úgy tudunk olvasási jogot biztosítani, ha felvesszük õket az `audit` csoportba. Mivel a vizsgálati naplók tartalmának figyelése jelentõs rálátást adhat a rendszerben jelenlevõ felhasználók és folyamatok viselkedésére, ajánlott körültekintõen kiosztani az olvasási jogokat.

17.5.4. Élõ rendszerfelügyelet a vizsgálati csövekkel

A vizsgálati csövek az eszközök állományabsztrakcióit klónozzák le, és ezzel teszik lehetõvé az alkalmazások számára, hogy menet közben megcsapolhassák a megfigyelt eszközök adatait. Ez az elsõdleges célja a különbözõ betörésfigyelõ és rendszerfelügyeleti eszközök készítõinek. A rendszergazda számára azonban a vizsgálati csövek megkönnyítik az élõ megfigyelést, mert itt nem merülnek fel a nyomok jogosultságaiból vagy az archiválás miatt megszakadó eseményfolyamokból adódó problémák. Az élõ eseményfolyamra az alábbi parancs kiadásával lehet rácsatlakozni:

```
# praudit /dev/auditpipe
```

Alapértelmezés szerint a vizsgálati csõhöz tartozó csomópontok kizárólag csak a `root` felhasználó részére érhetõek el. Az `audit` csoport tagjai úgy tudnak majd hozzáférni, ha felvesszük a következõ `devfs` szabályt a devfs.rules állományba:

```
add path 'auditpipe*' mode 0440 group audit
```

A devfs állományrendszer beállításáról bõvebben lásd a devfs.rules(5) oldalt.

Könnyen gerjedést lehet elõidézni a vizsgált események megfigyelésével, amikor is az egyes események megtekintése újabb vizsgálandó események sorozatát indítják el. Például, ha az összes hálózati forgalmat egyszerre vizsgáljuk és a praudit(1) egy SSH-munkameneten keresztül fut, akkor a vizsgálati események töméntelen áradata indul meg, mivel minden kiírandó esemény egy újabb eseményt indukál. Ennek elkerülése érdekében ajánlott a `praudit` parancsot részletes forgalmat nem figyelõ vizsgálati csõvel ellátott munkameneten keresztül elindítani.

17.5.5. A vizsgálati nyomok archiválása

A vizsgálati nyomokat egyedül a rendszermag képes írni, illetve csak a vizsgálati démon, az auditd képes felügyelni. A rendszergazdáknak ebben az esetben tehát nem szabad használniuk a

newsyslog.conf(5) vagy a hozzá hasonló eszközök használatát a vizsgálati naplók archiválásához. Helyettük a `audit` segédprogramot javasolt használni a vizsgálatok leállítására, a vizsgálati rendszer újrakonfigurálására vagy a napló archiválásának elvégzésére. Az alábbi parancs utasítja a vizsgálati démont, hogy hozzon létre egy új vizsgálati naplót és jelzi a rendszermagnak, hogy váltson erre az új naplóra. Az eddig használt naplót lezárja és átnevezi, ami ezután a rendszergazda által tetszõlegesen feldolgozható.

```
# audit -n
```

Ha az auditd démon a parancs kiadásának pillanatában nem futna, akkor hiba történik és errõl hibaüzenetet kapunk.

A cron(8) segítségével tizenként óránként kikényszeríthetjük a naplók váltását, ha felvesszük a /etc/crontab állományba az alábbi sort:

```
0     */12      *       *       *       root    /usr/sbin/audit -n
```

Ez a változtatás akkor fog érvénybe lépni, ha elmentjük az új /etc/crontab állományt.

A vizsgálati nyomok mérete szerinti automatikus váltás is megvalósítható az audit_control(5) állományban szereplõ `filesz` opció beállításával, amit meg is találhatunk ebben a fejezetben, a konfigurációs állományok beállításánál.

17.5.6. A vizsgálati nyomok tömörítése

Mivel a vizsgálati nyomok óriásira is megnõhetnek, sokszor felmerül az igény, hogy lehessen õket tömöríteni vagy más egyéb módon archiválni a vizsgálati démon által lezárt nyomokat. Az audit_warn szkript használható a különbözõ vizsgálatokhoz kapcsolódó események esetén elvégzendõ mûveletek megadásához, beleértve ebbe a vizsgálati nyomok váltásakor elvégzett szabályos lezárását. Például a következõket kell beleírnunk az audit_warn szkriptbe a nyomok lezárását követõ tömörítéséhez:

```
#
# Lezáráskor tömöríti a vizsgálati nyomot.
#
if [ "$1" = closefile ]; then
        gzip -9 $2
fi
```

Egyéb archiválási tevékenységek lehetnek még: a nyomok felmásolása egy központi szerverre, a régebbi nyomok törlése, vagy a meglevõ nyomok leszûkítése csak a fontos információkra. A szkript csak akkor fog lefutni, ha a vizsgálati nyomot sikerült szabályosan lezárni, így tehát a szabálytalan leálláskor megmaradó nyomok esetén nem.

A FreeBSD 6.3 és késõbbi verzióiban, a `praudit` XML kimeneti formátumot is támogat, amely az `-x` kapcsolóval érhetõ el.

Chapter 18. Háttértárak

18.1. Áttekintés

Ez a fejezet arról szól, hogy miként használjuk a lemezeinket a FreeBSD-vel. Itt többek közt szó esik a memória (alapú) lemezekrõl, a hálózaton keresztül csatlakoztatott meghajtókról, a szabványos SCSI/IDE tárolóeszközökrõl és az USB felületet használó eszközökrõl.

A fejezet elolvasása során megismerjük:

- a FreeBSD által alkalmazott terminológiát, amivel a fizikai lemezeken elhelyezkedõ adatokat írja le (partíciók és slice-ok);
- hogyan bõvítsük rendszerünket további merevlemezekkel;
- hogyan állítsuk be a FreeBSD-t USB tárolóeszközök használatára;
- hogyan állítsunk be virtuális állományrendszereket, például memórialemezeket;
- hogyan használjuk a kvótákat a lemezterület használatának korlátozására;
- hogyan védjüket meg lemezeinket titkosítással az illetéktelenektõl;
- FreeBSD alatt hogyan készítsünk és írjuk CD-ket, DVD-ket;
- a biztonsági mentések készítésének különbözõ lehetõségeit;
- hogyan használjuk a FreeBSD alatt rendelkezésünkre álló, biztonsági mentést készítõ programokat;
- hogyan mentsünk floppy lemezekre;
- mik az állományrendszerek pillanatképei és hogyan kell ezeket hatékonyan használni.

A fejezet elolvasásához ajánlott:

- a FreeBSD rendszermag beállításának és telepítésének ismerete (A FreeBSD rendszermag testreszabása)

18.2. Az eszközök elnevezései

A most következõ listában felsoroljuk a FreeBSD által ismert fizikai tárolóeszközöket és a hozzájuk tartozó elnevezéseket.

Táblázat 6. A fizikai lemezek elnevezésének szabályai

A meghajtó típusa	A meghajtóeszköz neve
IDE merevlemezek	`ad`
IDE CD-meghajtók	`acd`
SCSI merevlemezek és USB tárolóeszközök	`da`
SCSI CD-meghajtók	`cd`
Különbözõ nem szabványos CD-meghajtók	`mcd` (Mitsumi CD-ROM) és `scd` (Sony CD-ROM)

A meghajtó típusa	A meghajtóeszköz neve
Floppy meghajtók	`fd`
SCSI szalagos meghajtók	`sa`
IDE szalagos meghajtók	`ast`
Flash meghajtó	`fla` (DiskOnChip® Flash eszköz)
RAID meghajtók	`aacd` (Adaptec® AdvancedRAID), `mlxd` és `mlyd` (Mylex®), `amrd` (AMI MegaRAID®), `idad` (Compaq Smart RAID), `twed` (3ware® RAID).

18.3. Lemezek hozzáadása

Ebben a szakaszban arról lesz szó, hogy a jelenleg egyetlen meghajtót tartalmazó rendszerünket hogyan tudjuk bõvíteni egy új SCSI-lemez hozzáadásával. Ehhez elsõként kapcsoljuk ki a számítógépünket és szereljük be a helyére az új meghajtót a számítógép, a lemezvezérlõ és a meghajtó gyártójának utasításai alapján. Mivel ezt a mûveletet rengeteg módon lehet elvégezni, ezért ennek pontos részleteivel ez a leírás most nem foglalkozik.

Jelentkezzünk be `root` felhasználóként. Miután beszereltük a meghajtót, a /var/run/dmesg.boot állomány végignézésével bizonyosodjuk meg róla, hogy a rendszer valóban megtalálta a lemezt. A példánk szerint ez a meghajtó tehát a da1 nevet fogja viselni, amelyet a /1 könyvtárba akarunk csatlakoztatni (ha IDE-meghajtót telepítünk, akkor a hozzá tartozó eszköz neve ad1 lesz).

Mivel a FreeBSD IBM PC kompatibilis számítógépeken fut, ezért nem szabad figyelmen kívül hagynunk a PC BIOS partícióit is. Ezek eltérnek a hagyományos BSD partícióktól. Egy PC-s lemeznek négy BIOS-os partícióbejegyzése lehet. Ha egy lemezt tényleg csak a FreeBSD-nek szánunk, akkor használhatjuk az ún. *dedikált* módot. Minden más esetben a FreeBSD-nek egy PC BIOS partícióban kell elhelyezkednie. A FreeBSD a PC BIOS partícióit *slice*-nak nevezi, ezzel különbözteti ezeket a hagyományos BSD partícióktól. Dedikált esetekben is használhatjuk, de elsõsorban akkor kap fontosabb szerepet, amikor a FreeBSD-nek más operációs rendszerekkel kell megosztani a helyet. Ezzel el tudjuk kerülni, hogy a más operációs rendszerekben megtalálható, nem FreeBSD alapú `fdisk` parancs megzavarodjon.

A slice-ok használatakor a meghajtó /dev/da1s1e néven kerül hozzáadásra. Így kell olvasni: egyes SCSI lemezes egység (második SCSI lemez), elsõ slice (elsõ PC BIOS partíció) és e BSD partíció. A dedikált esetben a meghajtó neve viszont egyszerûen csak /dev/da1e.

Mivel a bsdlabel(8) 32 bites egész számokat használ a szektorok számának tárolására, ezért lemezenként csak 2^32-1 szektort tud ábrázolni, ami az esetek többségében 2 TB méretû címezhetõ területet jelent. Az fdisk(8) formátuma szerint sem a kezdõszektor, sem a hossz nem lehet 2^32-1-nél több, amivel így a partíciókat 2 TB, a lemezeket pedig 4 TB méretûre korlátozza. A sunlabel(8) formátuma partíciónként 2^32-1 szektort enged meg és összesen 8 partíciót, amely ezáltal 16 TB terület lefedését teszi lehetõvé. Nagyobb lemezekhez gpt(8) partíciók használatosak.

18.3.1. A sysinstall(8) használatával

1. Közlekedés a sysinstall programban

   A `sysinstall` könnyen használható menüinek segítségével az új lemezen pillanatok alatt létre tudunk hozni partíciókat és megcímkézni ezeket. Ehhez vagy `root` felhasználóként jelentkezzünk be a rendszerbe, vagy adjuk ki a `su` parancsot. A `sysinstall` parancs kiadása után lépjünk be a `Configure` (Beállítások) menübe. A `FreeBSD Configuration Menu` menüben ezután keressük meg és válasszuk ki az `Fdisk` menüpontot.

2. Az fdisk partíciószerkesztõ

   Miután eljutottunk az fdisk alkalmazáshoz, az `A` lenyomásával felajánlhatjuk az egész lemezt a FreeBSD számára. Amikor elõkerül a kérdés, hogy "remain cooperative with any future possible operating systems" ("mûködõképes maradjon-e a késõbbiekben telepítendõ operációs rendszerekkel"), akkor válaszoljuk rá `YES`-szel (tehát igen). A `W` gomb lenyomásával írjuk a lemezre a most elvégzett változtatásokat. Ezután már a `Q` használatával ki is léphetünk az FDISK szerkesztõbõl. A következõ lépésben a "Master Boot Record"-ról fognak minket megkérdezni. Mivel most egy már mûködõ rendszert bõvítünk, ezért a válaszunk erre `None` lesz.

3. A lemezcímkék szerkesztése

   Most lépjünk ki a sysinstall alkalmazásból és indítsuk el újra. Kövessük az iménti útmutatásokat, de ezúttal a `Label` menüpontot válasszuk ki. Ezzel a `Disk Label Editor`-ba vagyis a lemezcímkék szerkesztõjéhez jutunk. Itt fogjuk létrehozni a hagyományos BSD partíciókat. Egy lemezen nyolc ilyen partíció lehet, `a`-tól `h`-ig. Közülük néhány partíció címkéjét megkülönböztetjük. Az `a` partíció jelöli a rendszer indításához használt partíciót, a gyökérpartíciót (/). Tehát `a` partíció csak a rendszerlemezünkön szerepelhet (tehát ahonnan indul a rendszer). A `b` partíció a lapozáshoz használt partíciókat jelöli és több lemezen is szerepelhet. A `c` partíción keresztül lehet elérni az egészt lemezt dedikált módban vagy az egész FreeBSD slice-ot slice módban. A többi partíció tetszõlegesen felhasználható.

   A sysinstall címkeszerkesztõje az `e` betûvel szereti megjelölni a sem nem rendszerindító, sem nem lapozó partíciókat. A címkeszerkesztõben egyetlen állományrendszert a `C` lenyomásával lehet készíteni. Amikor erre válaszul megkérdezi a típusát (FS (állományrendszer) vagy swap (lapozóterület) legyen), akkor válasszuk az `FS` beállítást és adjuk meg a csatlakozási pontját (például /mnt). Amikor a lemezt telepítés után (post-install) adjuk hozzá, akkor a sysinstall valójában nem hoz létre hozzá bejegyzéseket az /etc/fstab állományban, ezért a csatlakozási pont megadása nem is feltétlenül fontos.

   Most már készen állunk arra, hogy rögzítsük az új címkét a lemezre és létrehozzunk vele egy állományrendszert. Ehhez nyomjuk le a `W` gombot. Ne foglalkozzunk vele, ha a sysinstall nem képes csatlakoztatni az új partíciót. Ha ezzel megvagyunk, akkor lépjünk ki a címkeszerkesztõbõl és a sysinstallból is.

4. Befejezés

   Most már csak annyi teendõnk maradt, hogy felvegyük az /etc/fstab állományba az új lemezhez tartozó bejegyzést.

18.3.2. Parancssoros eszközök használatával

Slice módban

Ezzel a beállítással a lemezünkre késõbb más operációs rendszereket is telepíthetünk, és nem okoz gondot a saját `fdisk` segédprogramjaik mûködésében. Az új lemezek telepítésénél ezt a módszer ajánlatos követni. A dedikált módot viszont csak abban az esetben használjuk, ha erre nyomós okunk van!

```
# dd if=/dev/zero of=/dev/da1 bs=1k count=1
# fdisk -BI da1 # inicializáljuk az új lemezt
# bsdlabel -B -w da1s1 auto # címkézzük meg
# bsdlabel -e da1s1 # szerkeszzük át a frissen létrehozott címkét és vegyünk fel egy
új partíciót
# mkdir -p /1
# newfs /dev/da1s1e # ismételjük meg minden létrehozott partícióhoz
# mount /dev/da1s1e /1 # csatlakoztassuk a partíció(ka)t
# vi /etc/fstab # vegyük fel a megfelelő bejegyzés(eke)t az /etc/fstab állományba
```

IDE-lemezek esetén azad eszközt a da eszközzel helyettesítsük.

Dedikált módban

Amennyiben az új meghajtót nem akarjuk megosztani egyetlen más operációs rendszerrel sem, használhatjuk a `dedicated` (dedikált) módot. Ne felejtsük el azonban, hogy ez képes összezavarni a Microsoft operációs rendszereit, habár ebbõl semmilyen kárunk nem fog származni. Az IBM OS/2® operációs rendszere azonban "kisajátít" minden olyan partíciót, amelyet nem tud olvasni.

```
# dd if=/dev/zero of=/dev/da1 bs=1k count=1
# bsdlabel -Bw da1 auto
# bsdlabel -e da1               # létrehozzuk az `e' partíciót
# newfs /dev/da1e
# mkdir -p /1
# vi /etc/fstab             # felvesszük a /dev/da1e partíciót
# mount /1
```

Egy másik megoldás:

```
# dd if=/dev/zero of=/dev/da1 count=2
# bsdlabel /dev/da1 | bsdlabel -BR da1 /dev/stdin
# newfs /dev/da1e
# mkdir -p /1
# vi /etc/fstab                 # felvesszük a /dev/da1e partíciót
# mount /1
```

18.4. RAID

18.4.1. Szoftveres RAID

Összefûzött lemezek beállítása

A nagyobb méretû háttértárolók kiválasztásánál a legfontosabb tényezõk a sebesség, megbízhatóság és a költség. Nagyon ritkán lehet csak ezt a hármat egyensúlyba hozni: általában a gyors és megbízható tárolóeszközök sok pénzbe kerülnek, valamint a költségek megtakarításához vagy a sebességet vagy pedig a megbízhatóságot kell feláldoznunk.

A továbbiakban egy olyan rendszert mutatunk be, ahol a elsõsorban a költségek, majd csak ezután a sebesség és megbízhatóság kerültek elõtérben. A rendszer adatátviteli sebességét a hálózat korlátozza. Habár emellett a megbízhatóság is nagyon fontos, a tárgyalt összefûzött meghajtó (Concenated Disk, CCD) csak adatokat szolgáltat és a teljes tartalma bármikor visszaállítható, mivel rendelkezésre áll CD-n.

A feladat elvégzésére alkalmas háttértároló kiválasztásában elsõként a saját elvárásainkat kell tudnunk megfogalmazni. Ha nekünk jobban számít az árnál a sebesség vagy a megbízhatóság, akkor a mostaniaktól némileg eltérõ konfigurációt kell majd építenünk.

A hardver telepítése

A rendszert tartalmazó IDE-lemez mellett három darab, egyenként 30 GB-os 5400-as percenkénti fordulatszámú Western Digital gyártmányú merevlemez alkotja majd a létrehozni kívánt, kb. 90 GB összméretû összefûzött lemezt. Ideális esetben minden IDE-lemez saját külön vezérlõn és kábelen van, de a költségek csökkentése miatt nem használtunk további IDE-vezérlõket. Ehelyett inkább jumperekkel úgy állítottuk be a lemezeket, hogy minden vezérlõre egy mester (master) és egy szolga (slave) módú merevlemez kapcsolódjon.

A beszerelés után beállítottuk a rendszer BIOS-át, hogy automatikusan felismerje a csatlakoztatott lemezeket. De ami még fontosabb, hogy a FreeBSD is észlelte ezeket az indítás során:

```
ad0: 19574MB <WDC WD205BA> [39770/16/63] at ata0-master UDMA33
ad1: 29333MB <WDC WD307AA> [59598/16/63] at ata0-slave UDMA33
ad2: 29333MB <WDC WD307AA> [59598/16/63] at ata1-master UDMA33
ad3: 29333MB <WDC WD307AA> [59598/16/63] at ata1-slave UDMA33
```

Ha a FreeBSD nem látná az összes lemezt, akkor ellenõrizzük a jumperek helyes beállítását. Napjainkban a legtöbb IDE-meghajtón találunk egy "Cable Select" jumpert is. Ezzel *nem* a mester/szolga módot állítjuk be! A megfelelõ jumper beazonosításához olvassuk el a meghajtóhoz tartozó dokumentációt.

A következõ lépésben azt vesszük nagyító alá, hogyan lehet ezeket az állományrendszer részévé tenni. Ezzel kapcsolatban a vinum(8) (A Vinum kötetkezelő) és a ccd(4) elolvasása ajánlatos. Erre a célra itt most a ccd(4) használatát választottuk.

A CCD beállítása

A ccd(4) meghajtó segítségével több ugyanolyan lemezt tudunk összefûzni egyetlen logikai állományrendszerré. A ccd(4) használatához arra is szükségünk van, hogy a ccd(4) támogatása jelen legyen a rendszermagban. A következõ sor tegyük bele a rendszermag konfigurációs állományába, fordítsuk újra és telepítsük a rendszermagot:

```
device   ccd
```

A ccd(4) támogatása modulként is betölthetõ.

A ccd(4) beállításához elõször a bsdlabel(8) programmal meg fel kell címkéznünk a lemezeket:

```
bsdlabel -w ad1 auto
bsdlabel -w ad2 auto
bsdlabel -w ad3 auto
```

Így létrejön egy-egy BSD típusú címke a ad1c, ad2c és ad3c eszközökre, amely így lefedi a lemez egész területét.

Most pedig változtassuk meg a lemezcímke típusát. Ehhez használjuk ismét a bsdlabel(8) programot:

```
bsdlabel -e ad1
bsdlabel -e ad2
bsdlabel -e ad3
```

Az EDITOR környezeti változóban megadott szövegszerkesztõvel (ez általában a vi(1)) megnyílik minden egyes lemezhez a jelenlegi lemezcímke.

Egy módosítatlan lemezcímke valahogy így néz ki:

```
8 partitions:
#        size   offset    fstype   [fsize bsize bps/cpg]
  c: 60074784        0    unused        0     0     0   # (Cyl.    0 - 59597)
```

A ccd(4) számára hozzunk létre egy új e partíciót. Ezt lényegében a c partíció lemásolásával keletkezik, de nála az fstype (az állományrendszer típusa) oszlopban mindenképpen 4.2BSD szerepeljen! A lemezcímke most már valahogy így fog kinézni:

```
8 partitions:
#        size   offset    fstype   [fsize bsize bps/cpg]
  c: 60074784        0    unused        0     0     0   # (Cyl.    0 - 59597)
  e: 60074784        0    4.2BSD        0     0     0   # (Cyl.    0 - 59597)
```

Az állományrendszer kiépítése

Most, miután felcímkéztük az összes lemezünket, lássunk neki a ccd(4) kiépítésének. Ezt a ccdconfig(8) meghívásával és az alábbihoz hasonló paraméterek átadásával tehetjük meg:

```
ccdconfig ccd0 32 0 /dev/ad1e /dev/ad2e /dev/ad3e
```

A paraméterek rövid leírása és használata: * Az elsõ paraméter a létrehozandó eszköz, ami jelen esetünkben a /dev/ccd0c. A /dev/ részt nem kötelezõ megadni. * A kihagyás nagysága az állományrendszerben. A kihagyás határozza meg a lemezblokkban alkalmazott csíkozás (striping) vastagságát, ami általában 512 byte. Ennek megfelelõen a 32-es kihagyás 16 384 byte-os csíkokat ad meg. * A ccdconfig(8) beállításai. Ha engedélyezni akarjuk a lemezek tükrözését, akkor itt megadhatjuk. Mivel ez a konfiguráció most nem nyújt tükrözést a ccd(4) számára, ezért állítsuk nullára (0). * A ccdconfig(8) parancsnak utolsóként azokat az eszközöket kell felsorolni, amelyeket tömbbe akarunk fûzni. Minden eszközt teljes elérési úttal adjuk meg.

A ccdconfig(8) futtatása után a ccd(4) beállítódik. Most már állományrendszert is rakhatunk rá. A newfs(8) man oldalról szedjük össze a szükséges paraméterezést, vagy egyszerûen csak gépeljünk be ennyit:

```
newfs /dev/ccd0c
```

Az egész önmûködõvé tétele

A ccd(4) eszközt általában minden egyes indítás után használni akarjuk. Ennek eléréséhez elõször ezt be kell állítanunk. Az alábbi parancs kiadásával írassuk be a jelenlegi beállításainkat tükrözõ /etc/ccd.conf állományt:

```
ccdconfig -g > /etc/ccd.conf
```

Az újraindítás során az `/etc/rc` parancs futtatja le a `ccdconfig -C` parancsot, ha az /etc/ccd.conf állomány létezik. Ez automatikusan beállítja a ccd(4) eszközöket, így ilyenkor tudjuk csatlakoztatni is ezeket.

Ha egyfelhasználós módban indítjuk a rendszert, mielõtt még a mount(8) paranccsal csatlakoztatni tudnánk a ccd(4) eszközt, a tömb beállításához meg kell hívnunk a következõ parancsot:

```
ccdconfig -C
```

Ha a rendszerindításkor automatikusan csatlakoztatni akarjuk a ccd(4) eszközt, akkor az /etc/fstab állományba helyezzünk el egy hozzá tartozó bejegyzést:

```
/dev/ccd0c              /media      ufs     rw      2       2
```

A Vinum kötetkezelõ

A Vinum kötetkezelõ egy blokkos eszközmeghajtó, ami virtuális lemezes meghajtókat valósít meg. Elkülöníti a lemezes hardvereszközöket a blokkos eszközmeghajtók felületétõl és a kettõ között úgy képezi le az adatokat, hogy a hagyományos lemezes tárolással szemben megnövekedett rugalmasságot, teljesítményt és megbízhatóságot kapunk. A vinum(8) ismeri a RAID-0, RAID-1 és RAID-5 modelleket egyaránt, melyeket önmagukban és együttesen kombinálva is használhatunk.

A A Vinum kötetkezelő bõvebben ismerteti a vinum(8) rendszerét.

18.4.2. Hardveres RAID

A FreeBSD rengeteg különbözõ típusú hardveres RAID-vezérlõt ismer. Ezek az eszközök a FreeBSD külön erre a célra szánt támogatása nélkül képesek vezérelni a RAID-alrendszert.

A rajta levõ BIOS segítségével a kártya a legtöbb lemezmûveletet egyedül kezeli. A következõkben egy Promise IDERAID vezérlõt alkalmazó rendszert fogunk beállítani. Miután telepítettük a kártyát és indítjuk a rendszert, bekéri a szükséges információkat. Kövessük az utasításokat és lépjünk be a kártya beállító képernyõjére. Itt tudjuk kombinálni az összes csatlakoztatott meghajtónkat. Amikor ezzel a végeztünk, a lemezek egyetlen lemezként fognak a FreeBSD számára viselkedni. A többi RAID-szint is ehhez hasonlóan állítható be.

18.4.3. Az ATA RAID-1 tömbök újraszervezése

A FreeBSD lehetõséget a tömbben levõ meghibásodott eszközök menet közben elvégezhetõ cseréjére. Ehhez arra van szükségünk, hogy még újraindítás elõtt elcsípjük a hibát.

Hiba esetén valami hasonlót fogunk látni a /var/log/messages állományban vagy a dmesg(8) kimenetében:

```
ad6 on monster1 suffered a hard error.
ad6: READ command timeout tag=0 serv=0 - resetting
ad6: trying fallback to PIO mode
ata3: resetting devices .. done
ad6: hard error reading fsbn 1116119 of 0-7 (ad6 bn 1116119; cn 1107 tn 4 sn 11)\\
status=59 error=40
ar0: WARNING - mirror lost
```

További információkat az atacontrol(8) programtól szerezhetünk:

```
# atacontrol list
ATA channel 0:
    Master:      no device present
    Slave:   acd0 <HL-DT-ST CD-ROM GCR-8520B/1.00> ATA/ATAPI rev 0

ATA channel 1:
    Master:      no device present
    Slave:       no device present
```

```
ATA channel 2:
    Master:  ad4 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
    Slave:       no device present

ATA channel 3:
    Master:  ad6 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
    Slave:       no device present

# atacontrol status ar0
ar0: ATA RAID1 subdisks: ad4 ad6 status: DEGRADED
```

1. A lemez biztonságos eltávolításához elõször válasszuk le (detach) a meghibásodott lemezhez tartozó csatornát:

   ```
   # atacontrol detach ata3
   ```

2. Cseréljük ki a lemezt.
3. Csatlakoztassuk újra (attach) az ATA csatornát:

   ```
   # atacontrol attach ata3
   Master:  ad6 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
   Slave:   no device present
   ```

4. Tartalékként (spare) adjuk hozzá az új lemezt a tömbhöz:

   ```
   # atacontrol addspare ar0 ad6
   ```

5. Szervezzük újra (rebuild) a tömböt:

   ```
   # atacontrol rebuild ar0
   ```

6. A folyamat elõrehaladását a következõ parancs begépelésével tudjuk figyelni:

   ```
   # dmesg | tail -10
   [a kimenet többi része]
   ad6: removed from configuration
   ad6: deleted from ar0 disk1
   ad6: inserted into ar0 disk1 as spare

   # atacontrol status ar0
   ar0: ATA RAID1 subdisks: ad4 ad6 status: REBUILDING 0% completed
   ```

7. Várjunk a mûvelet befejezõdéséig.

18.5. USB tárolóeszközök

Manapság már számos külsõ tárolóeszköz az USB (Universal Serial Bus) közvetítésével csatlakozik a számítógéphez: merevlemezek, pen drive-ok, CD-írók stb. A FreeBSD ezeket az eszközöket is ismeri.

18.5.1. Beállítás

A USB tárolóeszközöket kezelõ meghajtó, az umass(4) felelõs az USB alapú tárolóeszközök támogatásáért. Ha a GENERIC rendszermagot használjuk, akkor semmit sem kell változtatnunk. Ha saját rendszermagunk van, akkor gondoskodjunk róla, hogy a következõ sorokat beraktuk a rendszermag beállításait tartalmazó állományba:

```
device scbus
device da
device pass
device uhci
device ehci
device usb
device umass
```

Az umass(4) meghajtó a SCSI alrendszeren keresztül éri el az USB tárolóeszközöket, tehát az USB eszközeinket a rendszer SCSI eszközként látja. Az alaplapon található USB chipkészlet típusától függõen vagy csak a `device uhci`, vagy USB 1.*X* esetén pedig a `device ohci` bejegyzésre lesz szükségünk. De abból sem származik kárunk, ha mind a kettõt meghagyjuk. Az USB 2.0 szabványú vezérlõket a ehci(4) meghajtó (`device ehci`) támogatja. Ha módosítani kellett a konfigurációs állományt, akkor ne felejtsük el újrafordítani és telepíteni sem a rendszermagot.

Ha az USB eszközünk egy CD- vagy DVD-író, akkor a következõ sorral a SCSI CD-meghajtók meghajtóját, a cd(4) eszközt kell beépítenünk a rendszermagba:

```
device cd
```

Mivel az író is SCSI eszközként látszik, ezért az atapicam(4) nem szerepelhet a rendszermag beállításai között.

18.5.2. A beállítások kipróbálása

A beállításaink készen állnak a kipróbálásra: csatlakoztassuk a számítógéphez az USB eszközünket és a rendszerüzeneteket tároló pufferben (dmesg(8)) hamarosan meg is jelenik a hozzá tartozó meghajtó:

```
umass0: USB Solid state disk, rev 1.10/1.00, addr 2
GEOM: create disk da0 dp=0xc2d74850
da0 at umass-sim0 bus 0 target 0 lun 0
da0: <Generic Traveling Disk 1.11> Removable Direct Access SCSI-2 device
da0: 1.000MB/s transfers
```

```
da0: 126MB (258048 512 byte sectors: 64H 32S/T 126C)
```

Természetesen a gyártóra, márkára, az eszköz leírójára (da0) és egyebekre vonatkozó részletek eltérhetnek.

Mivel az USB eszköz SCSI eszközként látszik, ezért a `camcontrol` parancs használható a rendszerhez csatlakoztatott USB tárolóeszközök listázásához:

```
# camcontrol devlist
<Generic Traveling Disk 1.11>      at scbus0 target 0 lun 0 (da0,pass0)
```

Ha a meghajtón állományrendszer is található, akkor képesek vagyunk csatlakoztatni. A Lemezek hozzáadása elolvasása segíthet az USB meghajtón partíciókat kialakítani és formázni, amennyiben szükséges.

A rendszer biztonsága szempontjából nem tekinthetõ megbízhatónak, ha olyan felhasználók számára is engedélyezzük tetszõleges meghajtók csatlakoztatását (például a `vfs.usermount` engedelyézesével), amelyekben nem bízunk meg. A FreeBSD által támogatott állományrendszerek döntõ többsége nem nyújt védelmet a káros szándékkal telepített eszközök ellen.

Ha az eszközt normál felhasználókkal is csatlakoztathatóvá akarjuk tenni, akkor további lépések megtételére is szükségünk lesz. Elõször is a felhasználóknak valahogy el kell tudniuk érniük az USB tárolóeszköz csatlakoztatásakor keletkezõ eszközöket. Ezt úgy tudjuk megoldani, ha az érintett felhasználókat felvesszük az `operator` csoportba. Ebben a pw(8) lehet a segítségünkre. Másodsorban amikor ezek az eszközök létrejönnek, az `operator` csoportnak tudniuk kell ezeket olvasniuk és írniuk. Ezt úgy tudjuk megvalósítani, ha felvesszük a következõ sorokat az /etc/devfs.rules állományba:

```
[localrules=5]
add path 'da*' mode 0660 group operator
```

Ha viszont vannak SCSI lemezeink is rendszerben, akkor a helyzet egy kicsit megváltozik. Tehát például a rendszerben már eleve vannak da0, da1 és da2 néven lemezek, akkor a második sort ennek megfelelõen változtassuk meg:

```
add path 'da[3-9]*' mode 0660 group operator
```

Ezzel kizárunk minden, korábban már létezõ lemezt az `operator` csoportból.

Emellett még az /etc/rc.conf állományban engedélyeznünk kell a saját devfs.rules(5) szabályrendszerünket is:

```
devfs_system_ruleset="usb_rules"
```

Ezt követõen be kell állítanunk a rendszermagban, hogy a hagyományos felhasználók képesek legyenek állományrendszereket csatlakoztatni. Ezt a legkönnyebb úgy tudjuk megtenni, ha az /etc/sysctl.conf állományba felvesszük a következõ sort:

```
vfs.usermount=1
```

Azonban ne felejtsük el, hogy ez csak a rendszer következõ indításától él. De a sysctl(8) parancs használatával is beállíthatjuk ezt az értéket.

Az utolsó lépésben hozzunk létre egy könyvtárat az állományrendszer csatlakoztatásához. Ezt a könyvtárat az a felhasználó fogja birtokolni, aki az állományrendszert csatlakoztatnia akarja. Ez például `root` felhasználóként úgy tudjuk megtenni, ha a felhasználónak létrehozunk egy könyvtárat /mnt/felhasználó néven (ahol a *felhasználó* nevet cseréljük a tényleges felhasználó nevére, a *csoport* nevet pedig a felhasználóhoz tartozó elsõdleges csoport nevére):

```
# mkdir /mnt/felhasználó
# chown felhasználó:csoport /mnt/felhasználó
```

Most tegyük fel, hogy csatlakoztatnuk egy USB pen drive-ot és ennek megfelelõen megjelenik a /dev/da0s1 eszköz. Mivel az ilyen eszközökre általában gyárilag FAT állományrendszert tesznek, ezért így kell ezeket csatlakoztatni a mount(8) paranccsal:

```
% mount -t msdosfs -o -m=644,-M=755 /dev/da0s1 /mnt/felhasználó
```

Ha leválasztjuk az eszközt (miután kiadtuk a umount(8) parancsot), akkor a rendszerüzenetek között valami ilyesmit fogunk látni:

```
umass0: at uhub0 port 1 (addr 2) disconnected
(da0:umass-sim0:0:0:0): lost device
(da0:umass-sim0:0:0:0): removing device entry
GEOM: destroy disk da0 dp=0xc2d74850
umass0: detached
```

18.5.3. A témáról bõvebben

A Lemezek hozzáadása és az Állományrendszerek csatlakoztatása és leválasztása címû szakaszok elolvasása mellett a következõ man oldalakat is ajánljuk: umass(4), camcontrol(8) és usbconfig(8) FreeBSD 8.*X* esetében, vagy usbdevs(8) a FreeBSD korábbi változatainál.

18.6. Lézeres tárolóeszközök (CD-k) létrehozása és használata

18.6.1. Bevezetés

A CD-k számos lehetõségünkben eltérnek a hagyományos lemezektõl. Kezdetben a felhasználók nem is voltak képesek írni ezeket. Olyannak tervezték, hogy a fejek sávok közti mozgásából fakadó késleltetés nélkül lehessen folyamatosan olvasni. A szállítása a maga idejében sokkal könnyebb volt minden vele egyforma méretû eszköznél.

A CD-ken is találhatunk sávokat, azonban ez csak a folyamatosan olvasható adat egy szakaszát jelenti, nem pedig a lemez fizikai tulajdonságát. Ha FreeBSD-n akarunk CD-t készíteni, akkor ehhez elõször össze kell állítanunk a CD egyes sávjaira kerülõ adatokat és ezután rögzíteni ezeket a sávokat a CD-n.

Az ISO 9660 állományrendszert úgy tervezték, hogy megbirkózzon ezekkel az eltérésekkel. Sajnos ezzel együtt kõbe vésték az állományrendszerek akkoriban érvényes korlátozásait is. Szerencsére lehetõséget ad bõvítésre, ezáltal a helyesen megírt CD-k képesek úgy átlépni ezeket a határokat, hogy közben az általuk alkalmazott kiterjesztéseket nem ismerõ rendszerekkel is együtt tudnak mûködni.

A sysutils/cdrtools port tartalmaz egy mkisofs(8) nevû programot, amellyel létre tudunk hozni ISO 9660 típusú állományrendszert tartalmazó adatállományt. Többféle kiterjesztést is ismer, amit majd a lentebb ismertett opciókkal érhetünk el.

A CD írásához használt konkrét segédeszköz attól függ, hogy ATAPI vagy esetleg másmilyen írónk van. Az ATAPI CD-írók az alaprendszer részeként elérhetõ `burncd` programon keresztül használhatóak. A SCSI és USB CD-írók esetén pedig a sysutils/cdrtools portban megtalálható `cdrecord` programot használhatjuk. Az ATAPI/CAM modul segítségével a `cdrecord` és más SCSI-írókra készült programokat is tudunk használni ATAPI hardvereken.

Ha a CD-író szoftverünket grafikus felhasználói felületen keresztül szeretnénk használni, akkor az X-CD-Roast vagy a K3b alkalmazásokat érdemes szemügyre vennünk. Ezek az eszközök elérhetõek csomagként vagy a sysutils/xcdroast és sysutils/k3b portokból. ATAPI hardver esetén az X-CD-Roast és a K3b alkalmazások használatához szükségünk lesz az ATAPI/CAM modulra.

18.6.2. mkisofs

A sysutils/cdrtools port részeként elérhetõ mkisofs(8) program képes a UNIX® típusú állományrendszer könyvtárszerkezete alapján egy ISO 9660 típusú állományrendszert tartalmazó image-et készíteni. Legegyszerûbb módon így használhatjuk:

```
# mkisofs -o image.iso /az/elérési/út
```

Ezzel a paranccsal egy olyan *image.iso* nevû állományt hozunk létre, amely */az/elérési/út* által megadott helyen található könyvtárszerkezetet mintázza ISO 9660 állományrendszer formájában. A folyamat során minden olyan állományt leképez szabványos ISO 9660 állományrendszerbeli névre, amely megfelel a szabvány elvárásainak, és kihagy minden olyan állományt, amely nem jellemzõ az ISO állományrendszerekre.

Számos opció lehet segítségünkre az ilyenkor felbukkanó akadályok leküzdésében. Ezek közül

különösen fontos az -R, amely a UNIX® rendszerek számára megszokott Rock Ridge kiterjesztéseket, valamint a -J, amely a Microsoft rendszerekben használt Joliet kiterjesztéseit, és végül a -hfs, amely a Mac OS® alatt létrehozott HFS állományrendszerek kiterjesztéseit engedélyezi.

A kizárólag csak FreeBSD rendszereken használt CD-k esetében a -U megadásával kapcsolhatjuk ki az állománynevek mindenféle korlátozását. Az -R beállítás használatával olyan állományrendszer képét hozzuk létre, amely teljesen megegyezik a parancsban megadott könyvtárból induló fa tartalmával, habár több módon is sérti az ISO 9660 szabvány elõírásait.

Az utolsó általános jelleggel használható beállítás a -b. Ezzel lehet megadni az "El Torito" szabványnak megfelelõ rendszerindító CD készítéséhez szükséges rendszerindító image elérését. Ennél a beállításnál tehát meg kell adni a rendszerindításhoz használt lemez image-ét, amely a CD tartalmát magában foglaló könyvtárszerkezetben található valahol. A mkisofs(8) alapértelmezés szerint egy ún. "floppy emulációs" módban hozza létre az ISO image-et, ezért a rendszerindításhoz használatos lemez image-ének pontosan 1200, 1440 vagy 2880 KB méretûnek kell lennie. Egyes rendszerbetöltõk, mint amilyen például a FreeBSD terjesztéséhez használt lemezeken található, nem használják ezt az emulációt. Ilyen helyzetekben a -no-emul-boot kapcsolót kell megadni. Tehát ha a /tmp/sajátboot könyvtárban van egy indítható FreeBSD rendszerünk, amelyben a /tmp/sajátboot/boot/cdboot a rendszerindító lemez image-e, akkor egy /tmp/indítható.iso nevû ISO 9660 formátumú állományrendszert tartalmazó image-et például így tudunk elkészíteni:

```
# mkisofs -R -no-emul-boot -b boot/cdboot -o /tmp/indítható.iso /tmp/sajátboot
```

Miután ezt megtettük, és a rendszermagunkban benne van az md eszköz támogatása, csatlakoztathatjuk is az állományrendszert:

```
# mdconfig -a -t vnode -f /tmp/indítható.iso -u 0
# mount -t cd9660 /dev/md0 /mnt
```

Ezután már össze tudjuk vetni az /mnt és /tmp/sajátboot könyvtárak egyezõségét.

A mkisofs(8) viselkedését több más opcióval tudjuk finomhangolni, mint például az ISO 9660 kiosztás módosítása vagy a Joliet és HFS lemezek készítése. A mkisofs(8) man oldalon mindezekrõl bõvebben olvashatunk.

18.6.3. burncd

Ha ATAPI CD-írónk van, akkor a burncd paranccsal írhatjuk az ISO image-et a lemezre. A burncd az alaprendszer része, és /usr/sbin/burncd néven érhetõ el. A használata igen egyszerû, csupán pár paramétere van:

```
# burncd -f eszköz data image.iso fixate
```

Ezzel a paranccsal rámásoljuk az *image.iso* állományt az *eszköz* eszközre. Az alapértelmezett eszköz a /dev/acd0. A burncd(8) man oldalán találjuk meg az írási sebességgel, a CD írás utáni kiadásával és

az audio lemezek írásával kapcsolatos beállításokat.

18.6.4. cdrecord

Ha nincs ATAPI CD-írónk, akkor az íráshoz a `cdrecord` parancsot kell használnunk. A `cdrecord` nem az alaprendszer része: vagy a sysutils/cdrtools portból vagy a neki megfelelõ csomagból kell telepítenünk. Az alaprendszerben végbemenõ változások miatt a program bináris változatai hibázhatnak, aminek következtében csak "poháralátéteket" fogunk tudni gyártani. Ezért a rendszerrel együtt érdemes frissíteni ezt a portot is. Vagy ha a -STABLE verziót használjuk, akkor mindig érdemes a port elérhetõ legújabb verziójára frissíteni.

Miközben a `cdrecord` számos paraméterrel rendelkezik, az alapvetõ használata mégis egyszerûbb a `burncd` parancsénál. Egy ISO 9660 formátumú image-et ugyanis a következõ módon tudunk felírni lemezre:

```
# cdrecord dev=eszköz image.iso
```

A `cdrecord` használatának trükkös része a megfelelõ eszköz megtalálása, tehát a `dev` beállítás helyes megadása. Ehhez használjuk a `cdrecord-scanbus` paraméterét, amely az alábbihoz hasonló eredményt fog produkálni:

```
# cdrecord -scanbus
Cdrecord-Clone 2.01 (i386-unknown-freebsd7.0) Copyright (C) 1995-2004 Jörg Schilling
Using libscg version 'schily-0.1'
scsibus0:
        0,0,0     0) 'SEAGATE ' 'ST39236LW       ' '0004' Disk
        0,1,0     1) 'SEAGATE ' 'ST39173W        ' '5958' Disk
        0,2,0     2) *
        0,3,0     3) 'iomega  ' 'jaz 1GB         ' 'J.86' Removable Disk
        0,4,0     4) 'NEC     ' 'CD-ROM DRIVE:466' '1.26' Removable CD-ROM
        0,5,0     5) *
        0,6,0     6) *
        0,7,0     7) *
scsibus1:
        1,0,0   100) *
        1,1,0   101) *
        1,2,0   102) *
        1,3,0   103) *
        1,4,0   104) *
        1,5,0   105) 'YAMAHA  ' 'CRW4260        ' '1.0q' Removable CD-ROM
        1,6,0   106) 'ARTEC   ' 'AM12S          ' '1.06' Scanner
        1,7,0   107) *
```

Itt felsorolásra kerülnek a `dev` beállítás értékeként felhasználható eszközök. Keressük meg köztük a CD írónkat és a `dev` értékének a három vesszõvel elválasztott számot adjuk meg. Ebben az esetben a CD-író eszköz most az 1,5,0 lesz, tehát itt a helyes paraméterezés `dev=1,5,0`. Ezt az értékét könnyebben is meg lehet adni. Ennek részleteirõl a cdrecord(1) man oldalán olvashatunk. Abban az esetben is érdemes fellapoznunk, ha az audio sávok írásáról, az írási sebesség korlátozásáról vagy

más hasonló dolgokról akarunk olvasni.

18.6.5. Audio CD-k másolása

Audio CD-t úgy tudunk másolni, ha elõször állományok sorozatába mentjük a lemez tartalmát, majd ezeket az állományokat egy üres CD-re írjuk. Ennek konkrét folyamata azonban némileg eltér az ATAPI- és SCSI-meghajtók használata során.

Procedure: SCSI-meghajtók esetén

1. A `cdda2wav` programmal mentsük le a lemez tartalmát.

```
% cdda2wav -vall -D2,0 -B -Owav
```

2. A `cdrecord` paranccsal írjuk fel a .wav kiterjesztésû állományokat.

```
% cdrecord -v dev=2,0 -dao -useinfo  *.wav
```

Gondoskodjunk róla, hogy a *2,0* értéket a cdrecordnak megfelelõen helyesen állítottuk be.

Procedure: ATAPI-meghajtók esetén

Az ATAPI/CAM modul segítségével a `cdda2wav` parancs ATAPI meghajtókkal is használható. Ez a megoldás általában kedvezõbb (a hibák és bytesorrend ügyesebb kezelése, stb.) a legtöbb felhasználó számára, mint az itt ismertetett.

1. Az ATAPI CD meghajtója az egyes sávokat /dev/acddtnn néven teszi elérhetõvé, ahol a *d* a meghajtó sorszáma, a *nn* a sáv két számjeggyel kiírt sorszáma, amelyet szükség szerint balról nullával egészítenek ki. Így tehát az elsõ meghajtó elsõ sávja a /dev/acd0t01, a második a /dev/acd0t02, a harmadik a /dev/acd0t03 és így tovább.

   Ellenõrizzük, hogy ezek az eszközök jelen vannak a /dev könyvtárban. Amennyiben hiányoznának, kényszerítsük ki a lemez újbóli beolvasását:

```
# dd if=/dev/acd0 of=/dev/null count=1
```

2. Szedjük le az egyes sávokat a dd(1) használatával. A parancs kiadásakor meg kell adnunk egy blokkméretet is:

```
# dd if=/dev/acd0t01 of=track1.cdr bs=2352
# dd if=/dev/acd0t02 of=track2.cdr bs=2352
...
```

3. A `burncd` használatával írjuk fel a lemezre az imént lementett állományokat. Meg kell adnunk, hogy ezek audio állományok, és hogy a `burncd` a munka befejeztével zárja le (fixate) a lemezt.

```
# burncd -f /dev/acd0 audio track1.cdr track2.cdr ... fixate
```

18.6.6. Adat CD-k másolása

Az adatot tartalmazó CD-ket le tudjuk másolni egy olyan image-be, amely funkcionálisan megegyezik egy mkisofs(8) által létrehozott image-dzsel és amivel le tudunk másolni bármilyen adat CD-t. Az itt megadott példa azt feltételezi, hogy a CD-meghajtónk neve acd0. Helyére a saját CD-meghajtónk nevét kell behelyettesíteni.

```
# dd if=/dev/acd0 of=állomány.iso bs=2048
```

Most miután lementettük az image-et, írjuk fel CD-re a fentiek szerint.

18.6.7. Adat CD-k használata

Most, hogy már készítettünk egy szabványos adat CD-t, valószínûleg szeretnénk is valamilyen csatlakoztatni és elérni a rajta levõ adatokat. Alapértelmezés szerint a mount(8) mindig azt feltételezi, hogy az állományrendszerek `ufs` típusúak. Ezért ha valami ilyesmivel próbálkozunk:

```
# mount /dev/cd0 /mnt
```

akkor egy `Incorrect super block` szövegû hibaüzenetet lesz a jutalmunk, és természetesen nem tudjuk csatlakoztatni a CD-t. Mivel a CD nem `UFS` állományrendszert tartalmaz, ezért az ilyen jellegû kísérleteink mind kudarcba fognak fulladni. Valahogy fel kell világosítanunk a mount(8) parancsot arról, hogy itt most egy `ISO9660` típusú állományrendszert akarunk csatlakoztatni, és akkor minden a helyére kerül. Ezt úgy tudjuk megtenni, ha a mount(8) parancsnak megadjuk a `-t cd9660` paramétert. Például, ha a /dev/acd0 néven elérhetõ CD-meghajtóban levõ lemezt akarjuk a /mnt könyvtárba csatlakoztatni, akkor ezt kell begépelnünk:

```
# mount -t cd9660 /dev/cd0 /mnt
```

Vegyük észre, hogy az eszköz neve (ez ebben a példában most /dev/cd0) lehet más is attól függõen, hogy milyen csatolófelületet használ a CD-meghajtónk. Sõt, a `-t cd9660` valójában csak a mount_cd9660(8) parancsot indítja el. Ennek tükrében tehát az elõbbi példát így rövidíthetjük le:

```
# mount_cd9660 /dev/cd0 /mnt
```

Ezen a módon bármilyen gyártmányú adat CD-t képesek vagyunk csatlakoztatni. Egyes ISO 9660 kiterjesztéseket használó lemezek azonban esetleg furcsán mûködhetnek. Például Joliet lemezek az

összes állomány nevét kétbyte-os Unicode karakterben tárolják. A FreeBSD rendszermagja ugyan nem beszéli a Unicode-ot, de a FreeBSD CD9660 meghajtója képes menetközben átkonvertálni a Unicode karaktereket. Ha bizonyos nem angol karakterek kérdõjelekként jelennének meg, akkor a `-C` beállítás használatával még egy helyi kódlapot is meg kell adnunk. Ezzel kapcsolatban bõvebb tájékoztatásért forduljunk a mount_cd9660(8) man oldalhoz.

A `-C` beállítás segítségével csak akkor lesz képes a rendszermag elvégezni ezt az átalakítást, ha elõtte betöltjük a cd9660_iconv.ko modult. Ezt megtehetjük úgy, hogy ha felvesszük a következõ sort a loader.conf állományba:

```
cd9660_iconv_load="YES"
```

Indítsuk újra a számítógépünket, vagy közvetlenül töltsük be a modult a kldload(8) használatával.

Estenként elõfordulhat, hogy kapunk egy `Device not configured` hibaüzenetet a CD-k csatlakoztatásakor. Ez általában arra utal, hogy a CD-meghajtó nem érzékeli a berakott lemezt, vagy éppen a meghajtó nem látható a buszon. A CD-meghajtók esetében pár másodpercig eltarthat, amíg felismeri a berakott lemezt, ilyenkor mindig legyünk türelemmel.

Néha a SCSI CD-meghajtó nem látható, mert nem volt elég ideje válaszolni busz újraindítása elõtt. Ha SCSI CD-meghajtónk van, akkor a következõ beállítást tegyük hozzá a rendszermagunk konfigurációjához és fordítsuk újra a rendszermagukat.

```
options SCSI_DELAY=15000
```

Ezzel utasítjuk a SCSI buszunkat egy 15 másodperces várakozásra a rendszer indítása során, és így ezzel elég esélyt adunk arra, hogy a CD-meghajtó válaszolni tudjon a busz újraindítása elõtt.

18.6.8. Nyers adat CD-k írása

Írhatunk közvetlenül is állományokat a CD-re, ISO 9660 formátumú állományrendszer használata nélkül. Sokan így oldják meg a mentést. Ezt sokkal gyorsabban lebonyolítható egy szabványos CD esetében:

```
# burncd -f /dev/acd1 -s 12 data archive.tar.gz fixate
```

Az ezen a módon megírt CD-ket szintén nyers módon kell olvasnunk:

```
# tar xzvf /dev/acd1
```

Az ilyen lemezeket nem tudjuk a normális CD-khez hasonlóan csatlakoztatni. Sõt, az ilyen CD-ket csak FreeBSD alatt tudjuk olvasni. Ha csatlakoztathatóvá akarjuk tenni a lemezt, vagy más operációs rendszerek alól is szeretnénk olvasni, akkor erre a célra a fentebb bemutatott mkisofs(8) parancsot kell használnunk.

18.6.9. Az ATAPI/CAM meghajtó használata

Ez a meghajtó lehetõvé teszi az ATAPI eszközök (CD-ROM, CD-RW, DVD meghajtók stb...) számára, hogy a SCSI alrendszeren keresztül legyenek elérhetõek, így esetünkben is használhatóvá válnak olyan alkalmazások, mint például sysutils/cdrdao vagy a cdrecord(1).

A meghajtó használatához a következõ sort kell a /boot/loader.conf állományba illeszteni:

```
atapicam_load="YES"
```

Indítsuk újra a számítógépet.

Amennyiben a rendszermagban az atapicam(4) statikus támogatását szeretnénk használni, úgy a következõ sort kell a rendszermag konfigurációs állományába felvenni:

```
device atapicam
```

Továbbá a következõ sorokra lesz még szükségünk:

```
device ata
device scbus
device cd
device pass
```

Ezeknek már eleve ott kell szerepelnie. Ezután fordítsuk újra és telepítsük a rendszermagot, majd indítsuk újra a számítógépet.

A rendszer indulásakor az írónak ehhez hasonló módon kell megjelennie:

```
acd0: CD-RW <MATSHITA CD-RW/DVD-ROM UJDA740> at ata1-master PIO4
cd0 at ata1 bus 0 target 0 lun 0
cd0: <MATSHITA CDRW/DVD UJDA740 1.00> Removable CD-ROM SCSI-0 device
cd0: 16.000MB/s transfers
cd0: Attempt to query device size failed: NOT READY, Medium not present - tray closed
```

A meghajtó most már elérhetõ a /dev/cd0 eszközön keresztül, és például ennyi begépelésével csatlakoztatni tudunk róla egy CD-t a /mnt könyvtárba:

```
# mount -t cd9660 /dev/cd0 /mnt
```

`root` felhasználóként a következõ paranccsal tudjuk lekérdezi az író SCSI címét:

```
# camcontrol devlist
```

```
<MATSHITA CDRW/DVD UJDA740 1.00>   at scbus1 target 0 lun 0 (pass0,cd0)
```

Eszerint a `1,0,0` lesz az eszköz SCSI címe, amelyet a cdrecord(1) és más SCSI alkalmazások esetén adunk meg.

Az ATAPI/CAM és SCSI rendszerek tekintetében olvassuk el az atapicam(4) és cam(4) man oldalakat.

18.7. Lézeres tárolóeszközök (DVD-k) létrehozása és használata

18.7.1. Bevezetés

A DVD a CD-hez képest a lézeres tárolóeszközök technológiájának újabb generációját képviseli. A DVD bármelyik CD-nél több adatot képes tárolni és napjaink ez a videók kiadásának szabványa.

Öt fizikailag írható formátummal határozhatjuk meg az írható DVD fogalmát:

- DVD-R: Ez volt az elsõ elérhetõ írható DVD formátum. A DVD-R szabványát a DVD Fórum fektette le. Ez a formátum csak egyszer írható.
- DVD-RW: Ez a DVD-R szabvány újraírható változata. A DVD-RW körülbelül 1000 alkalommal írható újra.
- DVD-RAM: Ez is a DVD Fórum által támogatott újraírható formátum. A DVD-RAM cserélhetõ merevlemeznek látzsik. Azonban ez típusú adathordozó nem kompatibilis legtöbb DVD-ROM hajtóval és DVD-Video lejátszóval. Csupán csak néhány DVD-író ismeri a DVD-RAM formátumot. A DVD-RAM használatáról a A DVD-RAM használataban találunk bõvebben információkat.
- DVD+RW: Ezt az újraírható formátumot a DVD+RW szövetség alkotta meg. A DVD+RW lemezek nagyjából 1000 alkalommal írhatóak újra.
- DVD+R: Ez a formátum a DVD+RW formátum egyszer írható változata.

Az egyrétegû írható DVD-k összesen 4 700 000 000 byte-ot képesek rögzíteni, ami 4,38 GB vagy 4 485 MB (1 kilobyte itt 1024 byte).

Meg kell különböztetnünk fizikai tárolóeszközt és az alkalmazást. Például a DVD-Video állományok olyan jellegû elrendezését írja elõ, ami bármelyik írható fizikai DVD eszközön megjelenhet: DVD-R, DVD+R, DVD-RW stb. Mielõtt kiválasztanánk az eszköz típusát, biztosnak kell lennünk benne, hogy az író és a DVD-Video lejátszó (ez lehet egy önálló lejátszó vagy egy számítógép DVD-ROM meghajtója) kompatibilis a szóbanforgó lemezzel.

18.7.2. Beállítás

A growisofs(1) programot fogjuk a DVD rögzítésére használni. Ez a program a dvd+rw-tools segédprogramok (sysutils/dvd+rw-tools) gyûjteményének része. A dvd+rw-tools az összes DVD médium típusát ismeri.

Ezek a segédprogramok a SCSI alrendszeren keresztül érik az eszközöket, ezért a használhatukhoz

a rendszermagban szükségünk lesz az ATAPI/CAM támogatásra. Ha az írónk USB felületen csatlakozik, akkor mindez szükségtelen, és ehelyett a USB tárolóeszközökt kell elolvasnunk az USB eszközök beállításához.

Engedélyeznünk kell az ATAPI eszközök DMA hozzáférését is, amit a /boot/loader.conf állományban a következõ sor hozzáadásával tudunk megtenni:

```
hw.ata.atapi_dma="1"
```

A dvd+rw-tools használatának megkezdése elõtt a DVD-írónkkal kapcsolatban érdemes átolvasnunk a dvd+rw-tools hardverkompatibilitási jegyzeteit (angolul).

Ha grafikus felületet szeretnénk használni, akkor érdemes egy pillanatást vetnünk a K3bre (sysutils/k3b), amely egy felhasználóbarát felületet ad a growisofs(1) és sok más íróprogram felé.

18.7.3. Adat DVD-k írása

A growisofs(1) a mkisofs parancs elõlapja, tehát az állományrendszer létrehozásához a mkisofs(8) programot fogja meghívni és ezt írja fel a DVD-re. Ez azt jelenti, hogy az írási folyamat megkezdése elõtt nem kell semmilyen image-et létrehoznunk.

A /az/elérési/út könyvtárból a következõ paranccsal tudjuk kiírni az adatokat DVD+R vagy DVD-R lemezre:

```
# growisofs -dvd-compat -Z /dev/cd0 -J -R /az/elérési/út
```

A `-J -R` beállítások a mkisofs(8) programhoz kerülnek át az állományrendszer létrehozásakor (itt most egy ISO 9660 állományrendszert hozunk létre, Joliet és Rock Ridge kiterjesztésekkel), használatának részleteit lásd mkisofs(8).

A `-Z` beállítást a kezdõmenetek létrehozásakor használjuk: több menetben akarjuk írni a lemezt vagy sem. A DVD eszközt, amely itt most a */dev/cd0*, a saját konfigurációnknak megfelelõen kell megadni. A `-dvd-compat` paraméterrel lezárjuk a lemezt, így ezután további írás már nem lehetséges. Ezért cserébe jobb kompatibilitást kapunk a DVD-ROM meghajtókkal.

Elõre legyártott image-dzsel is dolgozhatunk, tehát például, ha az *image.iso* állományt akarjuk kiírni, akkor ezt kell lefuttatnunk:

```
# growisofs -dvd-compat -Z /dev/cd0=image.iso
```

Az írási sebességet magától beállítja a lemez és meghajtó képességeinek megfelelõen. Az írási sebesség felülbírálásához használjuk a `-speed=` paramétert. A paraméterek lehetõségeirõl a growisofs(1) man oldaláról tudhatunk meg többet.

4,38 GB-nál több adat írásához egy hibrid UDF/ISO-9660 típusú állományrendszert

kell létrehoznunk. Ezt úgy tudjuk elérni, ha mkisofs(8) és a többi hasonló program (például growisofs(1)) hívásakor még hozzátesszük az `-udf -iso-level 3` paramétereket. Ezekre csak lemezképek készítésekor vagy az állományok közvetlen lemezre írásakor van szükségünk. Az így létrehozott lemezeket a mount_udf(8) segédprogram segítségével UDF állományrendszerként tudjuk csatlakoztatni. Ezért csak olyan operációs rendszereken használható, amelyek ismerik ezt a formátumot, ellenkezõ esetben csak hibás állományokat fogunk látni a lemezen.

Példa ilyen lemezkép létrehozására:

```
# growisofs -dvd-compat -udf -iso-level 3 -Z /dev/cd0 -J -R
/az/új/adat/helye
```

Ha a lemezkép már eleve nagyobb méretû állományokat tartalmaz, a lemez írásakor a growisofs(1) programnak már nem kell további paramétereket átadnunk.

Lehetõleg mindig a sysutils/cdrtools legfrissebb verzióját használjuk (amely a mkisofs(8) programot is tartalmazza), mivel a régebbi verziók nem támogatják a nagyobb méretû állományokat. Ha problémák adódnak a programok használata során, akkor próbálkozzunk a fejlesztõi változattal (sysutils/cdrtools-devel) és olvassuk el a mkisofs(8) man oldalát.

18.7.4. DVD-Video írása

A DVD-Video az állományok speciális szervezésére utal, amely az ISO 9660 és az mikró UDF (M-UDF) specifikációkon alapszik. A DVD-Video emellett egy adott adatszerkezeti hierarchiát is takar, ezért kell egy külön programmal, például a multimedia/dvdauthor segítségével összeállítani egy DVD-t.

Ha már a birtokunkban van egy DVD-Video állományrendszer képe, akkor az eddigiek szerint egyszerûen csak írjuk fel egy lemezre, ahogy azt az elõzõ szakaszban is láthattuk. Ha összeállítottuk a DVD anyagát és például a /a/videó/elérési/útja könyvtárba raktuk, akkor a következõ paranccsal írathatjuk ki a DVD-Video formátumú lemezt:

```
# growisofs -Z /dev/cd0 -dvd-video /a/videó/elérési/útja
```

A `-dvd-video` paramétert kell átadni a mkisofs(8) programnak, amelynek hatására létrehoz egy DVD-Video formátumú állományrendszert. Emellett a `-dvd-video` beállítás maga után vonja a growisofs(1) `-dvd-compat` beállítását is.

18.7.5. A DVD+RW használata

Eltérõen a CD-RW-tõl, egy érintetlen DVD+RW-t az elsõ használat elõtt meg kell formázni. A growisofs(1) program errõl az elsõ adandó alkalommal gondoskodik, és ez az *ajánlott*. Azonban a DVD+RW formázására használhatjuk a `dvd+rw-format` parancsot is:

```
# dvd+rw-format /dev/cd0
```

Ezt a mûveletet csak egyszer kell elvégezni, hiszen ne feledjük, hogy csak a szûz DVD+RW lemezeket kell megformázni. Ezután a DVD+RW-t a korábbi szakaszoknak megfelelõen tudjuk írni.

Ha a DVD+RW-re új adatot akarunk írni (egy teljesen új állományrendszert, nem pedig adatokat hozzáfûzni), akkor nem kell üressé tenni a lemezt, egyszerûen csak elegendõ felülírni az elõzõeket (egy új kezdõmenet létrehozásával) valahogy így:

```
# growisofs -Z /dev/cd0 -J -R /az/új/adat/helye
```

A DVD+RW formátum felajánlja annak lehetõségét is, hogy könnyedén hozzá lehessen fûzni adatokat az elõzõ íráshoz. A mûvelet során az új menetet összefûzi a meglévõvel, tehát ez nem egy többmenetes írás, hanem a growisofs(1) *megnöveli* a lemezen található ISO 9660 állományrendszert.

Például, ha egy korábban megírt DVD+RW lemezen levõ adatokhoz akarunk hozzáírni, akkor a következõ parancsot kell kiadnunk:

```
# growisofs -M /dev/cd0 -J -R /az/új/adat/helye
```

A mkisofs(8) beállításainál a kezõmenetnél megadottakat érdemes ismét megadni.

Ha kompatibilisek akarunk maradni a többi DVD-meghajtóval, akkor adjuk meg `-dvd-compat` paramétert. Ez a DVD+RW esetében annyit jelent, hogy nem tudunk további adatokat hozzáfûzni.

Ha valamilyen okból mégis üressé szeretnénk tenni a lemez, akkor ír járhatunk el:

```
# growisofs -Z /dev/cd0=/dev/zero
```

18.7.6. A DVD-RW használata

A DVD-RW két lemezformátumot fogad el: a inkrementális soros hozzáférést és a korlátozott felülírást. Alapértelmezés szerint a DVD-RW lemezek soros elérésûek.

A még fel nem használt DVD-RW lemezek közvetlenül írhatóak külön formázás nélkül, habár a korábban már soros formátumban használt DVD-RW lemezeket egy új kezdõmenet létrehozása elõtt üressé kell tenni.

Soros módban így kell letörölni egy DVD-RW lemezt:

```
# dvd+rw-format -blank=full /dev/cd0
```

A teljes törlés (`-blank=full`) egy 1x média esetén körülbelül egy órát vesz igénybe. A `-blank` beállítással egy gyorsított törlés zajlik le, amennyiben a DVD-RW lemezt Disk-At-Once (DAO) módban írjuk. A DVD-RW lemezeket az alábbi paranccsal tudjuk DAO módban írni:

```
# growisofs -use-the-force-luke=dao -Z /dev/cd0=image.iso
```

A `-use-the-force-luke=dao` beállítást nem kötelezõ megadni, mivel a growisofs(1) igyekszik a lehetõ leggyorsabban törölni a lemezt és megkezdeni a DAO módú írást.

A DVD-RW esetében valójában a korlátozott felülírást lenne érdemes használnunk, mivel ez a formátum sokkal rugalmasabb az alapértelmezés szerint felkínált inkrementális soros elérésnél.

A soros DVD-RW lemezekre ugyanúgy tudunk adatokat rögzíteni, mint az összes többi formátum esetében:

```
# growisofs -Z /dev/cd0 -J -R /az/adat/helye
```

Ha az elõzõ íráshoz akarunk még hozzáfûzni adatokat, akkor ehhez a growisofs(1) `-M` beállítását kell használnunk. Azonban ha a DVD-RW lemezhet inkrementális soros módban adunk hozzá adatot, akkor ezzel egy új menetet hozunk létre a lemezen és így egy többmenetes lemezt kapunk.

A korlátozott felülírású DVD-RW formátum használata esetén nem kell mindegyik kezdõmenet elõtt törölni a lemezt, egyszerûen csak felül kell írni a `-Z` beállítással, hasonlóan a DVD+RW esetéhez. A DVD+RW `-M` beállításához hasonlóan lehetõségünk van a lemezen található ISO 9660 formátumú állományrendszer növelésére. Ennek az eredménye egy egymenetes DVD.

A következõ paranccsal tudjuk a DVD-RW lemezt korlátozott felülírású módba tenni:

```
# dvd+rw-format /dev/cd0
```

Így tudunk visszaváltani a soros formátum használatára:

```
# dvd+rw-format -blank=full /dev/cd0
```

18.7.7. Több menet használata

Nagyon kevés DVD-ROM meghajtó ismeri a többmenetes DVD-ket, és legtöbbször is csak általában az elsõ menetet olvassák. A DVD+R, DVD-R és DVD-RW formátumok soros formátumban képesek több mentetet is befogadni, viszont a DVD+RW és DVD-RW korlátozott felülírású formátuma esetén nem létezik több menet.

Az alábbi parancs egy újabb menetet ad hozzá egy megkezdett (le nem zárt) DVD+R, DVD-R vagy

DVD-RW soros formátumú lemezhez:

```
# growisofs -M /dev/cd0 -J -R /az/új/adat/helye
```

Ha ezt a parancsot egy korlátozott felülírású DVD+RW vagy DVD-RW lemez esetén adjuk ki, akkor az új adatokat úgy fûzi hozzá, hogy egy új menetet összefésüli a meglévõvel. Ezzel egy egymenetes lemez keletkezik. Ilyenkor így bõvítik a megkezdett lemezeket.

A menetek kezdése és befejezése általában felhasznál valamennyi helyet a lemezen. Ezért úgy tudjuk optimalizálni a lemez helykihasználtságát, hogy kevés menetben sok adatot viszünk fel rá. A DVD+R esetén 154, a DVD-R-nél körülbelül 2000, és a dupla rétegû DVD+R lemezeknél 127 menetet tudunk létrehozni.

18.7.8. További olvasnivalók

A DVD lemezrõl részletesebb információkat a `dvd+rw-mediainfo /dev/cd0` parancs kiadásával tudunk lekérdezni.

A dvd+rw-tools használatáról a growisofs(1) man oldalon találunk információt, valamint a dvd+rw-tools honlapján (angolul) és a cdwrite levelezési lista archívumaiban (angolul).

Futassuk `dvd+rw-mediainfo` parancsot minden olyan esetben, amikor gondunk akad valamilyen lemez írásával. A kimenete nélkül szinte lehetetlen segítenünk bárkinek is.

18.7.9. A DVD-RAM használata

Beállítás

A DVD-RAM írók SCSI vagy ATAPI csatolófelülettel rendelkeznek. Az ATAPI eszközök esetén engedélyezni kell a DMA elérését, amit a /boot/loader.conf állományban az alábbi sor hozzáadásával tudunk megtenni:

```
hw.ata.atapi_dma="1"
```

A lemez elõkészítése

Ahogy arra már korábban utaltunk a fejezet bevezetésében, a DVD-RAM úgy látható, mint egy cserélhetõ merevlemez. A hagyományos merevlemezekhez hasonlóan a DVD-RAM-ot is "elõ kell készíteni" az elsõ használatához. Ebben a példában a lemez teljes területét egy szabványos UFS2 állományrendszerrel töltjük fel:

```
# dd if=/dev/zero of=/dev/acd0 bs=2k count=1
# bsdlabel -Bw acd0
# newfs /dev/acd0
```

A DVD eszköz nevét, vagyis az acd0 eszközt a saját rendszerünknek megfelelõen kell módosítani.

A lemez használata

Miután az elõbbi mûveletet elvégeztük a DVD-RAM lemezen, már tudjuk is normális merevlemezként csatlakoztatni:

```
# mount /dev/acd0 /mnt
```

Ezt követõen a DVD-RAM egyaránt olvasható és írható.

18.8. Hajlékonylemezek létrehozása és használata

Néha hasznos lehet, ha az adatokat floppy lemezeken tároljuk, például olyankor, amikor más cserélhetõ tárolóeszköz már nem jöhet számításba, vagy amikor kis mennyiségû adatot kell átvinnünk az egyik számítógéprõl a másikra.

Ebben a szakaszban bemutatjuk hogyan kell FreeBSD alatt floppy lemezeket használni. Elsõsorban a 3,5 colos DOS lemezek formázásával és használatával foglalkozik, de ezek fogalmak a többi hajlékonylemezes formátum esetében is hasonlóak.

18.8.1. A hajlékonylemezek formázása

Az eszköz

A floppy lemezek a többi eszközhöz hasonlóan a /dev könyvtárban érhetõek el. A nyers floppy lemezek eléréséhez egyszerûen csak használjuk a /dev/fdN hivatkozást.

A formázás

Használat elõtt a floppy lemezeket alacsony szinten meg kell formázni. Ezt általában maga a gyártó végzi el, de a formázás gyakran hasznos lehet a lemez sértetlenségének ellenõrzésére. A legtöbb floppy lemez hivatalos kapacitása 1440 KB, de használhatjuk nagyobb (és kisebb) méretekben is.

A floppy lemezek alacsony szintû formázására az fdformat(1) parancsot használhatjuk. Ez a segédprogram paraméterként az eszköz nevét várja.

Figyeljünk a menetközben megjelenõ hibaüzenetekre, mivel ezek segítik eldönteni, hogy a lemez használható vagy sem.

A hajlékonylemezek formázása

A /dev/fdN eszközök segítségével tudunk megformázni egy floppy lemezt. Tegyünk be egy 3,5 colos floppy lemezt a meghajtóba, majd adjuk ki a következõ parancsot:

```
# /usr/sbin/fdformat -f 1440 /dev/fd0
```

18.8.2. A lemez címkézése

Miután alacsony szinten formáztuk a lemezt, tennünk kell rá egy lemezcímkét is. Ez a lemezcímke késõbb meg fog semmisülni, de a rendszernek szüksége van rá, hogy pontosan meg tudja állapítani a lemez méretét és geometriáját.

Az új lemezcímke lefedi az egész lemezt, és tartalmazni fogja az összes információt a floppy geometriájáról. A lemezcímkék geometriaértékeit az /etc/disktab állományban találjuk meg felsorolva.

Most már futtathatjuk is a bsdlabel(8) parancsot:

```
# /sbin/bsdlabel -B -w /dev/fd0 fd1440
```

18.8.3. Az állományrendszer

A hajlékonylemez most már készen áll a magas szintû formázásra. Ennek során egy új állományrendszert teszünk rá, amelyet a FreeBSD képes írni és olvasni. Miután létrejött ez az új állományrendszer, a lemezcímke megsemmisül, így tehát ha újra meg akarjuk formázni a lemezt, akkor újra létre kell majd hoznunk a lemezcímkét.

A floppy állományrendszere lehet UFS vagy FAT. A FAT általánosságban véve jobb választás a floppy lemezek számára.

Az alábbi módon tudunk új állományrendszert tenni a floppyra:

```
# /sbin/newfs_msdos /dev/fd0
```

A lemez most már készen áll a használatra.

18.8.4. A hajlékonylemezek használata

A floppy lemezt használatához a mount_msdosfs(8) paranccsal kell csatlakoztatnunk. Ugyanerre a célra használhatjuk a Portgyûjteményböl elérhetõ emulators/mtools portot is.

18.9. Szalagok létrehozása és használata

A legfontosabb szalagos adathordozók a 4 mm-es, 8 mm-es, QIC, a minikazettás és a DLT.

18.9.1. 4 mm-es (Digitális adattároló, avagy DDS: Digital Data Storage)

A 4 mm-es szalagok a QIC-szalagokat váltják fel a munkaállomások biztonsági mentésének eszközeként. Ez a tendencia csak tovább növekedett, ahogy a Conner felvásárolta az Archive-ot, a QIC típusú meghajtók legnagyobb gyártóját, majd leállított a QIC-meghajtók gyártását. A 4 mm-es meghajtók mérete kicsi és csendben is dolgoznak, de a megbízhatóság terén nem tudhatják maguknak mindazt a sikert, amit a 8 mm-es társaiknál könyvelhettünk el. A kazetták is sokkal olcsóbbak és kisebbek (3 x 2 x 0,5 col, ami 76 x 51 x 12 mm) a 8 mm-es kiadásénál. A 4 mm-es feje,

hasonlóan a 8 mm-eséhez, valamilyen okból szintén viszonylag rövid ideig bírja, és mind a kettõ spirális pásztázást használ.

Ezeknél a meghajtóknál az adatátvitel nagyjából 150 KB/mp-nél kezdõdik és 500 KB/mp-nél végzõdik. Az adattárolási képességük 1,3 GB-tól indul és 2,0 GB-ig tart. A hardveres tömörítés, ami a legtöbb ilyen típusú meghajtónál elérhetõ, közel megduplázza a kapacitást. A többmeghajtós szalagos könyvtár egységek egyetlen szekrényben 6 meghajtót képes befogadni, a szalagok automatikus cserélgetésével. Az ilyen könyvtárak kapacitása a 240 GB-ot is elérheti.

A DDS-3 szabvány most már akár 12 GB (vagy tömörítve 24 GB) kapacitást is elérhetõvé tesz.

A 4 mm-es meghajtók, hasonlóan a 8 mm-es meghajtókhoz, spirális pásztázást alkalmaznak. A spirális pásztázás összes elõnye és hátránya ezért egyaránt él a 4 mm-es és 8 mm-es meghajtók esetén.

A szalagok 2 000 menet vagy 100 teljes mentes után kopnak el.

18.9.2. 8 mm-es (Exabyte)

A 8 mm-es szalagok a legelterjedtebb szalagos SCSI-meghajtók. A szalagok használatára ez a legjobb választás. Szinte mindegyik rendszerben egy 2 GB-os 8 mm-es Exabyte szalagos meghajtót használnak. A 8 mm-es meghajtók megbízhatóak, kényelmesek és csendesek. A kazetták olcsók és kicsik (4,8 x 3,3 x 0,6 col, azaz 122 x 84 x 15 mm). A 8 mm-es szalagok feje viszonylag csak rövid ideig bírja a szalag nagy mértékû oda-vissza mozgása miatt.

Az adatátvitel sebessége 250 KB/mp-tõl 500 KB/mp-ig terjed, valamint a 300 MB-tól egészen 7 GB-os méretig találkozhatunk velük. A meghajtókban elérhetõ hardveres tömörítés képes közel megduplázni a kapacitást. Ezek a meghajtók önálló egységként is beszerezhetõek vagy egy 6 egységbõl álló és 120 szalagos szalagos könyvtár részeként. Ezek az egységek önállóan váltják a szalagokat. Az ilyen könyvtárak kapacitása eléri a közel 840 GB-ot.

Az Exabyte "Mammoth" modellje szalagonként 12 GB (tömörítéssel pedig 24 GB) adatot képes tárolni, viszont a hagyományos szalagos meghajtóknál nagyjából kétszer többe kerül.

Az adatok spirális pásztázással kerülnek a szalagra, és a fejek adott (nagyjából 6 fokos) szögben állnak a szalag felett. A szalag a fejeket tartó orsó köré tekeredik, körülbelül 270 fokban. Ennek eredményképpen nagyobb adatsûrûség és szorosan zárt sávok jönnek létre, ahogy ebben a szögben a fej eljut a szalag egyik élérõl a másikra.

18.9.3. QIC

A QIC-150 meghajtók és szalagok talán a legelterjedtebb szalagos egységek és adathordozók. A QIC szalagos meghajtók a legolcsóbb "komolynak tekinthetõ" biztonsági mentésre alkalmas meghajtók. Az olcsóság azonban megköveteli a maga árát. A QIC-szalagok a 4 és 8 mm-es szalagokkal szemben akár ötször is drágábbak lehetnek gigabyte-onként. De ha megelégszünk csupán féltucat szalaggal is, akkor a QIC jó vásárnak tûnhet. A QIC a *leginkább* elterjedtebb szalagos meghajtó. Minden rendszerben biztonsan találunk valamilyen minõségben QIC-meghajtót. A QIC fizikailag hasonló (és gyakran azonos) felépítésû szalagokat gyárt rengeteg különbözõ adatsûrûséggel. Az ilyenkor keletkezõ súrlódások miatt a QIC-meghajtók egyáltalán nem nevezhetõek csendesnek. Az ilyen

típusú meghajtók az adatok rögzítése elõtt külön hangjelenség kíséretében keresik meg a megfelelõ pozíciót és tisztán hallható, ahogy olvasnak, írnak és keresnek. A QIC-szalagok mérete 6 x 4 x 0,7 col (avagy 152 x 102 x 17 mm).

Az adatátviteli sebesség nagyjából 150 KB/mp-tõl 500 KB/mp-ig terjedhet. A kapacitás szalagonként 40 MB és 15 GB között változhat. A legtöbb újabb QIC-meghajtó támogatja a hardveres tömörítést. QIC-meghajtókat azonban egyre kevésbé találhatunk, helyüket szépen lassan mindenhol átveszik a DAT-meghajtók.

A szalagokra sávokban rögzítik az adatokat. Ezek a sávok szalag felületének hosszanti tengelyén futnak az egyik végétõl a másikig. A sávok száma valamint a sávok vastagsága a szalagok kapacitásától függõen változnak. Ha nem is összes legújabb, de a legtöbb meghajtó legalább olvasás szintjén kompatibilis a régebbi típusokkal (de gyakran írásban is). A QIC híresen megbízható az adatbiztonság tekintetében (a mechanikája sokkal egyszerûbb és strapabíróbb a spirális pásztázással mûködõ meghajtókénál).

A szalagokat 5000 mentés után érdemes lecserélni.

18.9.4. DLT

A DLT rendelkezik a legnagyobb adatátviteli sebességgel az itt összefoglalt mezõnyben. A 1/2 colos (12,5 mm-es) szalag egy egyorsós tokban foglal helyet (mérete 4 x 4 x 1 col, azaz 100 x 100 x 25 mm). A tok egyik oldalán végig egy csúszó kapu található. A meghajtó ezt a kaput nyitja ki és ezen keresztül húzza be a szalagot. A szalag elején található egy ovális lyuk, amibe a meghajtó "bele tud akaszkodni". A feszítõ orsó a szalagos meghajtóban foglal helyet. Az összes többi szalag esetén (kivéve egyedül a 9 sávos szalagokat) mind a segéd- és feszítõ orsók magában a kazettában találhatóak.

Az adatátviteli sebessége megközelítõleg 1,5 MB/mp, tehát háromszor nagyobb bármelyik 4 mm-es, 8 mm-es vagy QIC-szalagos egységénél. Az adattároló képessége kazettánként 10 GB-tól 20 GB-ig terjedhet. A meghajtók egyaránt elérhetõek többkazettás, cserélgetõs és többkazettás, többmeghajtós könyvtárakban is, melyek 5 kazettától egészen 900 kazettáig, illetve 1 meghajtótól 20 meghajtóig képesek befogadni, így teljes tárterületük 50 GB-tól 9 TB-ig terjed.

A DLT Type V formátum tömörítéssel közel 70 GB-os kapacitást képes elérni.

A szalagra az adatok a haladási iránnyal párhuzamosan kerülnek fel (akárcsak a QIC-szalagok esetében). Egyszerre két sávot rögzít. A író/olvasó fejek élettartama viszonylag nagy. Ahogy a szalag megáll, a fej és a szalag között nincs szükség további relatív mozgásra.

18.9.5. AIT

Az AIT a Sony új formátuma, ami egészen 50 GB mennyiségû adatot képes tárolni (tömörítéssel) egyetlen szalagon. A szalagokat memóriachipekkel látják el, melyek a szalag tartalmát indexelik. Az indexek felhasználásával aztán a szalagos meghajtó villámgyorsan képes meghatározni a szalagon található állományok helyét, szemben az ilyenkor megszokott többperces mûvelettel. A SAMS:Alexandria és a hozzá hasonló szoftverek negyven vagy több AIT-szalagos könyvtárral is képesek egyszerre dolgozni, és közvetlenül a szalagok memóriájával veszik fel a kapcsolatot a tartalmuk megjelenítéséhez, a mentett állományok rendszerezéséhez, a helyes szalag

megkereséséhez, betöltéséhez és visszatöltéséhez.

Az ilyen könyvtárak a 20 000 dolláros (kb. 3,5 millió forintos) árkategóriába tartoznak, ami miatt csak egy kicsivel csúsznak ki a hobbi kategóriából.

18.9.6. Az új szalagok elsõ használata

Amikor az elsõ alkalommal akarunk beolvasni vagy írni egy új, teljesen üres szalagot, hibára fogunk futni. Egy ehhez hasonló konzolüzenet fog megjelenni:

```
sa0(ncr1:4:0): NOT READY asc:4,1
sa0(ncr1:4:0):  Logical unit is in process of becoming ready
```

A szalag nem tartalmaz azonosító blokkot (Identifier Block) a nulladik blokkban. A QIC-525 szabvány átvétele óta mindegyik QIC szalagos meghajtó létrehozza ezt az azonosító blokkot. Tehát két megoldás létezik:

- Az `mt fsf 1` paranccsal felírunk egy ilyen azonosító blokkot a szalagra.
- A meghajtó elõlapján található gomb segítségével dobassuk ki a szalagot.

  Rakjuk vissza a szalagot és hajtsunk végre rajta egy `dump` parancsot.

  A `dump` parancs erre egy `DUMP: End of tape detected` ("szalag vége") hibaüzenetet ad, majd a következõ jelenik meg a konzolon: `HARDWARE FAILURE info:280 asc:80,96`.

  Tekertessük vissza a szalagot az `mt rewind` paranccsal.

  A szalag következõ mûvelete most már sikeres lesz.

18.10. Biztonsági mentés hajlékonylemezekre

18.10.1. Hajlékonylemezre is lehet biztonsági mentést készíteni?

A floppy lemezek nem igazán felelnek meg biztonsági mentés készítésére, mivel:

- Nem megbízható adathordozók, különösen hosszabb idõre.
- Esetükben a mentés és visszaállítás nagyon lassú.
- Kapacitásuk erõsen korlátozott (annak már régen elmúlt az ideje, amikor egész merevlemezeket tudtunk lementeni egy tucat floppyra).

Habár ha máshogy nem tudunk biztonsági mentést készíteni, akkor a floppy lemezekkel még mindig jobban járunk, mint nélkülük.

Ha már mindenképpen floppy lemezeket kell használnunk, akkor igyekezzünk minél jobb minõségûeket beszerezni. Tehát az olyan floppyk, amik már évek óta kavarognak az irodában, erre a célra nem éppen bizonyulnak a legjobb választásnak. Ideális esetben egy megbízható gyártótól származó új floppykat használunk.

18.10.2. Tehát akkor hogyan mentsük az adatokat hajlékonylemezre?

Legegyszerûbban a tar(1) `-M` (többkötetes) opciójával tudunk floppy lemezre menteni, aminek használatával több floppyra kiterjedõ mentéseket is készíthetünk.

Az aktuális könyvtár és a benne levõ alkönyvtárak tartalmát (`root`) felhasználóként a következõ paranccsal tudjuk lementeni:

```
# tar Mcvf /dev/fd0 *
```

Amikor az elsõ floppy megtelik, a tar(1) kérni fogja a következõ kötetet (volume) (mivel a tar(1) adathordozótól független módon hivatkozik a kötetekre, tehát ebben a környezetben a kötet egy floppy lemezt jelent):

```
Prepare volume #2 for /dev/fd0 and hit return:
```

Az üzenet fordítása:

```
Készítse elő a 2. kötetet a /dev/fd0 eszközön és nyomja le a
return billentyűt
```

A folyamat egészen addig ismétlõdik (a kötetek számának növekedésével), amíg az összes állomány lementésre nem kerül.

18.10.3. Lehet tömöríteni a mentéseket?

Sajnos a tar(1) többkötetes mentések esetén nem engedi a `-z` beállítás használatát. Természetesen ettõl függetlenül a gzip(1) segítségével még be tudjuk tömöríteni az összes állományt, a tar(1) paranccsal floppyra menteni ezeket, majd a gunzip(1) paranccsal kitömöríteni.

18.10.4. Hogyan állítsuk vissza a biztonsági mentéseket?

Az egész mentés visszaállításához adjuk ki a következõ parancsot:

```
# tar Mxvf /dev/fd0
```

Két módon tudunk csak bizonyos állományokat visszaállítani. Elõször is, tegyük be a mentés elsõ lemezét és adjuk ki a következõ parancsot:

```
# tar Mxvf /dev/fd0 állomány
```

A tar(1) segédprogram ezután sorban kérni fogja a többi lemezt egészen addig, amíg meg nem találja a keresett állományt.

Vagy ha pontosan tudjuk, hogy melyik lemezen található a keresett állomány, akkor az iménti parancs használatát azzal a lemezzel kezdjük. Vigyázzunk, mert ha a lemezen található elsõ állomány az elõzõ lemezen kezdõdik, akkor a tar(1) figyelmeztetni fog minket, hogy nem állítja vissza még akkor sem, ha erre nem is kértük!

18.11. Mentési stratégiák

Egy biztonsági mentés kidolgozása során az elsõ követelmény gondoskodnunk az alábbi problémákról:

- Lemezhiba
- Az állományok véletlen törlése
- Az állományok véletlenszerû károsodása
- Számítógépek teljes megsemmisülése (például tûz által), belértve a közelében tárolt összes biztonsági mentést

Tökéletesen megoldható, hogy egyes rendszerek a fentebb felsorolt problémák mindegyikét teljesen eltérõ technikával oldják meg. A nagyon személyes rendszerektõl és a nagyon értéktelen adatoktól eltekintve szinte egyértelmûen kizárt, hogy egyetlen technika képes lefedni az összes problémát.

Kelléktárunk néhány alapvetõ eszköze:

- Az egész rendszer mentése, amit egy megbízható helyre elzárt, tartós adattárolóra készítünk. Ez tulajdonképpen védelmet biztosít a fentebb megemlített összes probléma esetében, de lassú és kényelmetlen róla visszaállítani az adatokat. A közelben és/vagy neten is tarthatunk errõl másolatokat, de még így is kényelmetlen az állományok visszaállítása, különösen az egyszerû felhasználók számára.
- Pillanatképek készítése az állományrendszerrõl. Ez valójában csak olyan esetekben lehet a segítségünkre, amikor véletlenül töröltünk állományokat, ám ilyenkor *határozottan* jól jön, mivel igen gyorsan és könnyen lehet vele dolgozni.
- Az egész állományrendszer és/vagy az összes lemez másolata (például az rsync(1) idõszakos alkalmazása a komplett gépre). Az általában az egyedi igényekkel bíró hálózatok esetében eshet a kezünkre. A lemezhiba ellen védelemben ez a megoldás általában a RAID alatt áll. A véletlenül törölt állományok visszaállításának tekintetében az UFS pillanatképeivel mérhetõ össze, de ez leginkább a saját igényeinktõl függ.
- RAID alkalmazása. A lemezek meghibásodása esetén segíti minimalizálni vagy elkerülni a kiesést, ugyan gyakori lemezhibák árán (mivel ilyenkor több lemezt használunk) de kisebb sürgõsséggel.
- Az állományok ujjlenyomatának ellenõrzése. Az mtree(8) segédprogram nagyon hasznos tud lenni ebben az esetben. Habár ez nem egy mentési technika, mégis segít megállapítani, hogy mikor kell nyugdíjba küldenünk a biztonsági mentéseinket. Ez különösen az aktív nem használt mentésekre vonatkozik, ezeket bizonyos idõ elteltével mindig érdemes ellenõrizni.

Nagyon könnyû lenne további technikákat is felsorolni, melyek legtöbbje az iméntiek valamilyen kombinációja lenne. A speciális igények általában speciális technikákat eredményeznek (például egy éles adatbázis biztonsági mentése általában az adott adatbáziskezelõ rendszer közremûködését

is elvárja). Mindig fontos tudni, hogy milyen veszélyek ellen védekezünk és hogyan kezeljük le ezeket.

18.12. Alapvetõ tudnivalók a biztonsági mentésrõl

A dump(8), tar(1) és cpio(1) a három legfontosabb biztonsági mentésekkel kapcsolatos program.

18.12.1. Mentés és helyreállítás

A UNIX® típusú rendszerekben a biztonsági mentést hagyományosan a `dump` és `restore` programok végzik. A meghajtókat lemezblokkok összeségeként kezelik, az állományrendszerek által létrehozott állományok, linkek és könyvtárak szintje alatt. Eltérõen más, biztonsági mentést végzõ szoftverektõl, a `dump` az adott eszközön egy egész állományrendszert képes lementeni. Nem képes csak az állományrendszer vagy egy több állományrendszerre kiterjedõ könyvtárszerkezet egy részét lementeni. A `dump` nem állományokat és könyvtárakat ír a szalagra, hanem nyers adatblokkokat, amelyek állományokat és könyvtárakat formáznak. A `restore` parancs az adatokat alapértelmezés szerint a /tmp könyvtárba tömöríti ki. Ha nem lenne elegendõ helyünk a /tmp könyvtárban, akkor a `TMPDIR` környezeti változó átállításával ehelyett megadhatunk egy olyat, ahol már kellõ mennyiségû terület áll rendelkezésre a `restore` akadálytalan lefutásához.

Ha a `dump` parancsot a gyökér könyvtárban adjuk ki, akkor nem fogja lementeni a /home vagy /usr vagy bármilyen más könyvtárat, mivel ezek jellemzõ módon más állományrendszerek csatlakozási pontja vagy más állományrendszerekre mutató szimbolikus linkek.

A `dump` parancsnak vannak olyan rigolyái, amelyek még az AT&T UNIX 6. verziójából (1975 környékérõl) maradtak vissza. Az alapértelmezett paraméterezése 9 sávos szalagokat feltételezi (6250 bpi), nem pedig a napjainkban elterjedt nagy írássûrûségû (egészen 62 182 ftpi-s) adathordozókat. Ezek az alapértelmezések természetesen paranccsorból felülbírálhatóak, és így a manapság alkalmazott szalagos meghajtók teljes kapacitása is kihasználható vele.

Emellett az `rdump` és `rrestore` programok segítségével hálózaton keresztül is le tudjuk menteni az adatainkat egy másik számítógépre csatlakoztatott szalagos egységre. Mind a két program az rcmd(3) és a ruserok(3) parancsokat használja a távoli szalagos meghajtó eléréséhez. Az `rdump` és `rrestore` paramétereinek a távoli számítógép használatához kell illeszkedniük. Amikor egy FreeBSD rendszerû számítógépet az `rdump` paranccsal egy Sun rendszerû, `komodo` nevû számítógépre mentünk, amelyhez egy Exabyte szalagos meghajtó csatlakozik, akkor ezt a írjuk be:

```
# /sbin/rdump 0dsbfu 54000 13000 126 komodo:/dev/nsa8 /dev/da0a 2>&1
```

Figyelem: az .rhosts állományon keresztül hitelesítésnek megvannak a maga biztonsági kockázatai. Ne felejtsük el felmérni ezt a saját környezetünkben sem.

A `dump` és `restore` parancsokat az `ssh` használatával még biztonságosabbá tehetjük.

Példa 21. A dump használata az ssh alkalmazással

```
# /sbin/dump -0uan -f - /usr | gzip -2 | ssh -c blowfish \
          célfelhasználó@cél.gép.hu dd of=/nagyállományok/dump-usr-l0.gz
```

Vagy az RSH környezeti változó megfelelõ beállításával használhatjuk a dump beépített módszerét:

Példa 22. A dump használata az ssh alkalmazással, az RSH környezeti változó beállításával

```
# RSH=/usr/bin/ssh /sbin/dump -0uan -f célfelhasználó@cél.gép.hu:/dev/sa0 /usr
```

18.12.2. tar

A tar(1) is az AT&T UNIX 6. verziójáig nyúlik vissza (tehát nagyjából 1975-ig). A tar az állományrendszerrel szoros együttmûködésben dolgozik, állományokat és könyvtárakat ír a szalagra. A tar ugyan nem ismeri a cpio(1) által felkínált összes lehetõséget, de nincs is szüksége olyan szokatlan paranccsoros összekapcsolásokra, mint a cpio parancsnak.

A FreeBSD 5.3 vagy késõbbi változataiban a GNU tar és az alapértelmezés szerinti bsdtar egyaránt elérhetõ. A GNU változat a gtar paranccsal hívható meg. Az rdump parancshoz hasonló felírásban képes kezelni a távoli eszközöket. Tehát így tudjuk használni a tar parancsot a komodo nevû Sun számítógép Exabíte szalagos meghajtójának elérésére:

```
# /usr/bin/gtar cf komodo:/dev/nsa8 . 2>&1
```

Ugyanez eltérhetõ a bsdtar használatával is, amikor az rsh programmal összekapcsolva küldünk át a távoli szalagos egységre.

```
# tar cf - . | rsh hálózati-név dd of=szalagos-eszköz obs=20b
```

Ha a hálózaton keresztül mentés során fontos számunkra a biztonság, akkor az rsh parancs helyett az ssh parancsot használjuk.

18.12.3. cpio

A cpio(1) eredetileg a UNIX® szalagos programjai és szalagos egységei között közvetített. A cpio parancs (többek közt) képes a byte-ok sorrendjének felcserélésére, több különbözõ archívum formátuma szerint írni és adatokat közvetíteni más programok felé. Ez utóbbi lehetõsége miatt a cpio kíválóan alkalmas a telepítõeszközök számára. A cpio nem képes bejárni a könyvtárszerkezetet, és az állományok listáját a szabványos bemeneten keresztül kell megadni neki.

A cpio nem támogatja a biztonsági mentés átküldését a hálózaton. Programok összekapcsolásával és

az rsh használatával tudunk adatokat küldeni távoli szalagos meghajtókra.

```
# for f in könyvtár_lista; do
find $f >> mentési.lista
done
# cpio -v -o --format=newc < backup.list | ssh felhasználó@gép "cat > mentőeszköz"
```

Ahol a *könyvtár_lista* a menteni kívánt könyvtárak listája, a *felhasználó@gép* a mentést végzõ gép felhasználójának és hálózati nevének együttese, valamint a *mentõeszköz*, ahova a mentés kerül (például /dev/nsa0).

18.12.4. pax

A pax(1) az IEEE/POSIX® válasza a tar és cpio programokra. Az évek során a tar és a cpio különbözõ változatai egy kissé inkompatibilissé váltak. Ezért a szabványosításuk kiharcolása helyett inkább a POSIX® létrehozott egy új archiváló segédprogramot. A pax megpróbálja írni és olvasni a cpio és tar formátumok legtöbb változatát, valamint emellett további saját formátumokat is kezel. A parancskészlete inkább a cpio parancséra emlékeztet, mintsem a tar parancséra.

18.12.5. Amanda

Az Amanda (Advanced Maryland Network Disk Archiver) egy kliens-szerver alapú mentési rendszer, nem pedig egy önálló program. Az Amanda szerver menti tetszõleges számú számítógép adatát egyetlen szalagra, melyek az Amanda klienst futtatják és hálózaton keresztül hozzá csatlakoznak. A nagy mennyiségû és nagy kapacitású lemezekkel rendelkezõ rendszerekben közvetlenül a mentéshez szükséges idõ nem áll rendelkezésre a feladat elvégzéséhez. Az Amanda viszont képes megoldani ezt a problémát. Az Amanda képes egy "saját lemez" használatával egyszerre több állományrendszerrõl is biztonsági mentést készíteni. Az Amanda"archívumkészleteket" hoz létre: az Amanda konfigurációs állományában megadott állományrendszerekrõl készít teljes mentést egy adott idõ alatt egy adott mennyiségû szalagra. Az "archívumkészlet" ezenkívül még tartalmaz egy napi inkrementális (vagy különbözeti) mentést is minden egyes állományrendszerrõl. A sérült állományrendszerek visszaállításához mindig a legújabb teljes biztonsági mentésre és a hozzá tartozó inkrementális mentésekre van szükségünk.

A konfigurációs állomány segítségével precíz irányítást gyakorolhatunk a létrehozott mentések és az Amanda által keltett hálózati forgalom felett. Az Amanda a fentiek közül bármelyik programmal képes az adatokat szalagra rögzíteni. Az Amanda portként vagy csomagként is elérhetõ, alapértelmezés szerint nem települ.

18.12.6. Ne csináljunk semmit

A "Ne csináljunk semmit" nem egy újabb számítógépes program, hanem egy igen gyakran alkalmazott mentési stratégia. Nem kell beruházni. Nem kell semmilyen biztonsági mentési rendet követni. Egyszerûen semmit se csinálunk. Ha véletlenül valami történne az adatainkkal, akkor csak mosolyogjunk és törõdjünk bele!

Amennyiben az idõnk és adataink keveset vagy éppen semmit se érnek, akkor a "Ne csináljunk semmit" az elérhetõ legjobb biztonsági mentési megoldás számítógépünk számára. De legyünk

óvatosak, mert a UNIX® egy igen hasznos eszköz, és fél éven belül könnyen úgy találhatjuk magunkat, hogy mégis csak vannak értékes adataink.

A "Ne csináljunk semmit" tökéletesen megfelelõ mentési módszer a /usr/obj és a hozzá hasonló módon a számítógépen automatikusan generált könyvtárak és állományok esetében. Ugyanilyen példa lehetne a kézikönyv HTML vagy PostScript® változata. Ezek a formátumok ugyanis az SGML források alapján keletkeznek, így a HTML vagy PostScript® állományok mentése nem életbevágó. Az SGML állományokat viszont már annál inkább mentsük!

18.12.7. Melyik a legjobb?

dump(8) *Pont.* Elizabeth D. Zwicky komolyan letesztelte az itt felsorolt összes programot. A UNIX® állományrendszerek jellegzetességeinek és rajtuk az összes adatunk megõrzésének egyértelmûen a `dump` felel meg a legjobban. Elizabeth a minden egyes program tesztjéhez olyan állományrendszereket hozott létre, amelyek rengeteg különféle szokatlan helyzetet tartalmaztak (valamint néhány nem annyira szokatlant). Az érintett jellegzetességek: lyukas állományok, lyukas állományok és egy halom nulla, állományok érdekes karakterekkel a nevükben, olvashatatlan és írhatatlan állományok, eszközök, a mentés közben méretüket változtató állományok, a mentés közben keletkezõ és megszûnõ állományok és még sok minden más. Az eredményeit a LISA V-ben jelentette meg 1991 októberében. Lásd A biztonsági mentéshez és archiváláshoz használt programok tesztje (angolul).

18.12.8. Az adatok helyreállítása vészhelyzetben

A katasztrófa elõtt

Csupán négy lépést kell megtennünk az esetleges katasztrófák bekövetkezésének esetére.

Elõször is két példányban nyomtassuk ki az egyes lemezek lemezcímkéjét (például a `bsdlabel da0 | lpr` paranccsal) valamint az állományrendszerek táblázatát (az /etc/fstab állományt) és az összes rendszerindításkor megjelenõ üzenetet.

A második lépésben készítenünk kell egy "élõ" rendszerrel rendelkezõ CD-lemezt. Ezen a lemezen megtalálható minden, ami el tudunk indítani egy helyreállításhoz elegendõ rendszert. Ekkor a felhasználó futtatni tudja például a dump(8), restore(8), fdisk(8), bsdlabel(8), newfs(8), mount(8) és a többi segédprogramot. Ez az image a FreeBSD/i386 12.0-RELEASE kiadáshoz az ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/ISO-IMAGES/12.0/FreeBSD-12.0-RELEASE-i386-livefs.iso címrõl tölthetõ le.

A harmadik lépésben igyekezzünk minél gyakrabban szalagra menteni. Mindig gondoljuk arra, hogy a legutolsó mentés óta létrehozott változatásaink teljesen el fognak veszni. A mentéseket tartalmazó szalagokat tegyük írásvédetté.

A negyedik lépésben ellenõrizzük a a második lépésben készített helyreállító lemezünket és a biztonsági mentéseket tartalmazó szalagokat. Jegyezzük le az eljárást. Ezeket a jegyzeteket is rakjuk el rendszerindító lemezzel, a kinyomtatott adatokkal és a mentéseket tartalmazó szalagokkal együtt. Ezek a jegyzetek megvédenek minket attól, hogy a helyreállítás közbeni kétségbeesésünkben nehogy véletlenül tönkretegyük a biztonsági mentéseinket. (Hogy miként is? Például ha a `tar xvf /dev/sa0` parancs helyett izgalmunkban a `tar cvf /dev/sa0` parancsot gépeljük be, akkor azzal

felülírjuk a biztonsági mentéseinket).

A fokozott biztonság kedvéért minden alkalommal készítsünk rendszerindító lemezt és legalább két mentést. Az egyiket valamilyen távoli helyen tároljuk. Ez a távoli hely NE ugyanannak az épületnek az alagsora legyen! Számos cég alaposan megtanulta ezt a szabályt a Világkereskedelmi központ tragédiája kapcsán. Ez a távoli hely számítógépeinkbõl és merevlemezes meghajtóinkól is fizikailag jól elkülöníthetõ, jelentõs távolságban legyen.

A katasztrófa után

Az alapvetõ kérdés: a hardver túlélte? Ha rendszeresen készítettünk biztonsági mentéseket, akkor a szoftverek miatt egyáltalán nem kell aggódnunk.

Ha a hardver megsérült, akkor a számítógép használatának újból megkezdése elõtt javasolt cserélni a meghibásodott alkatrészeket.

Ha a hardverrel minden rendben találtunk, akkor helyezzük be a helyreállításhoz használatos "élõ" rendszert tartalmazó lemezt a CD-meghajtóba, és indítsuk el vele a számítógépet. Ezután nemsokára a telepítési menü jelenik meg. Itt a megfelelõ ország után a Fixit — Repair mode with CDROM/DVD/floppy or start a shell ("Helyreállítás CD/DVD/floppy használatával, vagy parancssor indítása"), majd a CDROM/DVD — Use the live filesystem CDROM/DVD ("A CD/DVD-n található élõ rendszer használata") menüpontokat válasszuk. A `restore` és a többi segédprogram a /mnt2/rescue könyvtárban lesznek elérhetõek.

Egyenként állítsuk vissza az egyes állományrendszereket.

A `mount` paranccsal próbáljuk meg csatlakoztatni az elsõ lemezünk rendszerindító partícióját (például `mount /dev/da0a /mt`). Ha a lemezcímke megsérült, akkor `bsdlabel` alkalmazásával partícionáljuk újra a lemezt és címkézzük meg a korábban kinyomtatott címke adatainak megfelelõen. A `newfs` segítségével újra hozzuk létre az állományrendszereket. Írható-olvasható módban csatlakoztassuk újra a lemez rendszerinító partícióját (`mount -u -o rw /mnt`). A biztonság mentést végzõ program és a biztonsági mentést tartalmazó szalagok használatával állítsuk helyre az állományrendszer tartalmát (például `restore vrf /dev/sa0`). Válasszuk le az állományrendszert (például `umount /mnt`). Mindegyik sérült állományrendszerre ismételjük a folyamatot.

Ahogy mûködõképessé vált a rendszerünk, mentsük az adatainkat új szalagokra. Akármi is okozta a rendszer összeomlását vagy az adatvesztést, ismét lecsaphat. Ha most áldozunk erre még egy órát, akkor azzal a késõbbiekben számos kellemetlenségtõl óvhatjuk meg magunkat.

18.13. Hálózat, memória és állomány alapú állományrendszerek

A számítógépünkben létezõ fizikai lemezek, például floppyk, CD-k, merevlemezek és egyebek mellett a lemezek egy másik formáját is képes megérteni a FreeBSD - a *virtuális lemezeket*.

A virtuális lemeznek tekinthetõek többek közt az olyan hálózati állományrendszerek, mint például a Hálózati állományrendszer (Network File System, NFS) és a Coda, valamint a memóriában és állományokban létrehozott állományrendszerek.

Attól függõen, hogy a FreeBSD melyik változatát használjuk, az állomány és memória alapú állományrendszerek létrehozásához, illetve használatához különbözõ segédprogramokra lesz szükségünk.

A devfs(5) a felhasználó számára láthatatlan módon hozza létre az eszközök leíróit.

18.13.1. Állomány alapú állományrendszerek

FreeBSD alatt az mdconfig(8) segédprogram segítségével tudunk memórialemezeket (md(4)) beállítani és engedélyezni. Az mdconfig(8) használatához be kell töltenünk az md(4) modult vagy hozzá kell tennünk a rendszermagunk beállításait tartalmazó állományhoz:

```
device md
```

Az mdconfig(8) parancs háromféle memória alapú virtuális lemezt ismer: a malloc(9), állományok vagy lapozóterület használatával létrehozott memórialemezeket. Így lehet például csatlakoztatni a floppyk vagy CD-k állományokban tárolt image-eit.

Egy meglevõ állományrendszer image-ének csatlakoztatása:

Példa 23. Egy meglevõ állományrendszer image-ének csatlakoztatása az `mdconfig` paranccsal

```
# mdconfig -a -t vnode -f image -u 0
# mount /dev/md0 /mnt
```

Új állományrendszer létrehozása az mdconfig(8) használatával:

Példa 24. Új állomány alapú lemez létrehozása az `mdconfig` paranccsal

```
# dd if=/dev/zero of=új-image bs=1k count=5k
5120+0 records in
5120+0 records out
# mdconfig -a -t vnode -f új-image -u 0
# bsdlabel -w md0 auto
# newfs md0a
/dev/md0a: 5.0MB (10224 sectors) block size 16384, fragment size 2048
        using 4 cylinder groups of 1.25MB, 80 blks, 192 inodes.
super-block backups (for fsck -b #) at:
 160, 2720, 5280, 7840
# mount /dev/md0a /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity  Mounted on
/dev/md0a       4710    4  4330     0%    /mnt
```

Ha az `-u` beállítással nem adjuk meg az egység számát, akkor az mdconfig(8) az md(4) automatikus

kiosztásán keresztül fog egy használatban még nem levõ eszközt kiválasztani. Az így kiosztott egység neve az md4 névhez hasonlóan jelenik meg a szabványos kimeneten. Az mdconfig(8) használatának részleteirõl olvassuk el a hozzá tartozó man oldalt.

Az mdconfig(8) egy nagyon sokoldalú segédeszköz, habár használatakor viszonylag sok parancsot kell kiadni egy állomány alapú állományrendszer létrehozásához. A FreeBSD azonban alapból tartalmaz még egy mdmfs(8) nevû segédprogramot is, ami az md(4) lemezeket az mdconfig(8) segítségével állítja be, létrehoz rajtuk egy UFS típusú állományrendszert a newfs(8) segítségével és csatlakoztatja a mount(8) paranccsal. Így például, ha az iménti állományrendszert akarjuk létrehozni és csatlakoztatni, akkor egyszerûen csak gépeljünk be ennyit:

Példa 25. Állomány alapú lemezek beállítása és csatlakoztatása az `mdmfs` paranccsal

```
# dd if=/dev/zero of=új-image bs=1k count=5k
5120+0 records in
5120+0 records out
# mdmfs -F új-image -s 5m md0 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity  Mounted on
/dev/md0        4718    4  4338     0%    /mnt
```

Ha az `md` paramétert az egység száma nélkül adjuk meg, akkor mdmfs(8) az md(4) automatikus kiosztására támaszkodva fog egy addig még nem használt eszközt kiválasztani. A mdmfs(8) használatának pontos részleteivel kapcsolatban lásd a hozzá tartozó man oldalt.

18.13.2. Memória alapú állományrendszerek

A memória alapú állományrendszerek esetében általában a "lapozóállomány alapú" megközelítést alkalmazzák. A lapozóállomány alapúság nem arra utal, hogy a memórialemezt alapból kilapozzák lemezre, hanem inkább arra, hogy a memórialemez olyan területen jön létre, amelyet szükség esetén lemezre lehet lapozni. Memória alapú lemezeket a (rendszermag szintû) malloc(9) használatával is létre lehet hozni, de a malloc alapú memórialemezeknél, különösen a nagyon nagyok esetében, a rendszer könnyen össze tud omlani, ha kifut a rendelkezésére álló memóriából.

Példa 26. Új memória alapú lemez létrehozása az `mdconfig` paranccsal

```
# mdconfig -a -t swap -s 5m -u 1
# newfs -U md1
/dev/md1: 5.0MB (10240 sectors) block size 16384, fragment size 2048
        using 4 cylinder groups of 1.27MB, 81 blks, 192 inodes.
        with soft updates
super-block backups (for fsck -b #) at:
 160, 2752, 5344, 7936
# mount /dev/md1 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity  Mounted on
```

```
/dev/md1        4718    4  4338     0%    /mnt
```

Példa 27. Új memória alapú lemez létrehozása az `mdmfs` paranccsal

```
# mdmfs -s 5m md2 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity  Mounted on
/dev/md2        4846    2  4458     0%    /mnt
```

18.13.3. Memórialemezek leválasztása a rendszerrõl

Amikor már nem akarunk tovább használni egy memória vagy állomány alapú állományrendszert, érdemes visszaadnunk az általuk felhasznált erõforrásokat a rendszernek. Elsõként válasszuk le magát az állományrendszert, majd az mdconfig(8) segítségével kapcsoljuk le a lemezt a rendszerrõl és szabadítsuk fel az általa felhasznált erõforrásokat.

Például az /dev/md4 eszközt így lehet lekapcsolni és felszabadítani:

```
# mdconfig -d -u 4
```

A beállított md(4) eszközökkel kapcsolatos többi információt az `mdconfig -l` paranccsal tudjuk lekérdezni.

18.14. Az állományrendszerek pillanatképei

A FreeBSD a Soft Updates mellett felkínál egy másik lehetõséget: az állományrendszerekrõl készíthetõ pillanatfelvételeket.

Ezek a pillanatképek lehetõvé teszik a felhasználók számára, hogy adott állományrendszerekrõl képeket hozzanak létre és azt állományként kezeljék. A pillanatképeket az adott állományrendszerben kell létrehozni, és a felhasználók állományrendszerenként húsznál többet nem hozhatnak belõlük létre. Az aktív pillanatképek a szuperblokkban kerülnek rögzítésre, ezért az állományrendszerek leválasztása és újracsatlakoztatása esetén is megmaradnak, még újraindítás után is. Amikor egy pillanatképre már nincs tovább szükségünk, egy szimpla rm(1) paranccsal eltávolítható. A pillanatképek tetszõleges sorrendben eltávolíthatóak, habár ilyenkor az összes általuk lefoglalt hely nem szabadul fel, mivel más pillanatképeknek még szüksége lehet bizonyos blokkjaira.

Miután az mksnap_ffs(8) paranccsal létrehoztunk egy pillanatképet tartalmazó állományt, beállítódik rá a módosíthatatlanságot jelentõ `snapshot` állományjelzõ. Egyedül az unlink(1) parancs képez ez alól kivételt, mivel segítségével a pillanatképek eltávolíthatóak.

A pillanatképek a mount(8) paranccsal hozhatóak létre. A következõ módon tudjuk a /var egy pillanatképét elkészíteni a /var/snapshot/snap állományban:

```
# mount -u -o snapshot /var/snapshot/snap /var
```

Vagy a mksnap_ffs(8) meghívásával is készíthetünk pillanatképeket:

```
# mksnap_ffs /var /var/snapshot/snap
```

Az állományrendszeren (például /var) a pillanatképeket tartalmazó állományokat a find(1) paranccsal kereshetjük meg:

```
# find /var -flags snapshot
```

Ahogy elkészítettünk egy pillanatképet, több mindenre is felhasználhatjuk:

- Egyes rendszergazdák a pillanatképeket biztonsági mentésekhez használják, mivel ezek gond nélkül áttehetõek CD-re vagy szalagra.
- Az állományrendszerek sértetlenségét ellenõrzõ program, az fsck(8) is lefuttatható egy ilyen pillanatképen. Feltéve, hogy az állományrendszer csatlakoztatásakor tiszta volt, mindig egy tiszta (és változásokat nem tartalmazó) eredményt kell kapnunk. Ennek megléte elengedhetetlen a háttérben futtatható fsck(8) mûködéséhez.
- Futassuk le a dump(8) segédprogramot a pillanatképen. Az így létrehozott mentés megegyezik az állományrendszer adott pillanatban felvett állapotával. Az `-L` beállítás megadásával maga a dump(8) is képes egyetlen parancsban pillanatfelvételt készíteni, ebbõl létrehozni a mentést, majd eltávolítani.
- A pillanatképet képesek vagyunk a mount(8) paranccsal az állományrendszer befagyasztott változataként csatlakoztatni:

  ```
  # mdconfig -a -t vnode -f /var/snapshot/snap -u 4
  # mount -r /dev/md4 /mnt
  ```

Így már a /mnt könyvtárba csatlakoztatva be tudjuk járni a befagyasztott /var állományrendszert. Minden a pillanatfelvétel készítésének idõpontjának megfelelõ állapotban fog maradni. Az egyetlen kivétel talán annyi, hogy korábbi pillanatképek nulla méretû állományként fognak megjelenni. Mikor befejeztük a pillanatképek használatát, a umount(8) paranccsal le tudjuk választani:

```
# umount /mnt
# mdconfig -d -u 4
```

A `softupdates` és az állományrendszerek pillanatképeinek használatával, illetve mûszaki leírásukkal kapcsolatban látogassuk meg Marshall Kirk McKusick honlapját a http://www.mckusick.com/ címen (angolul).

18.15. Az állományrendszerek kvótái

A kvóták használata az operációs rendszerben egy olyan választható lehetõség, aminek segítségével állományrendszerenként korlátozni tudjuk az egyes felhasználók vagy csoporttagok által elhasznált lemezterület és/vagy állományok mennyiségét. Ezt leggyakrabban olyan idõosztásos rendszerekben használják ki, ahol szükség lehet az egyes felhasználókra vagy csoportokra esõ erõforrások mennyiségének szabályozására. Ezzel tudjuk megakadályozni, hogy a felhasználók vagy csoportok elfogyasszák az összes rendelkezésre álló lemezterületet.

18.15.1. A kvóták használatának beállítása

Mielõtt nekilátnánk a kvóták használatának, meg kell gyõzõdnünk róla, hogy a rendszermagunkban megvan hozzá a szükséges támogatás. A kvótákat a következõ sorral lehet engedélyezni a rendszermag beállításait tartalmazó állományban:

```
options QUOTA
```

A gyári GENERIC rendszermag ezt alapból nem engedélyezi, ezért ehhez mindenképpen be kell állítani, le kell fordítani és telepíteni egy kell saját rendszermagot. A saját rendszermag létrehozásához kövessük a A FreeBSD rendszermag testreszabása utasításait.

Ha ezzel megvagyunk, akkor a következõ sorral bõvítsük ki az /etc/rc.conf állományt:

```
enable_quotas="YES"
```

A kvótákat kezelõ rendszer indításának finomabb szabályozására létezik még egy további beállítási lehetõség is. A rendszer indítása során általában az egyes állományrendszerek kvótáját a quotacheck(8) program ellenõrzi. A quotacheck(8) gondoskodik róla, hogy a kvótákat tároló adatbázis ténylegesen az állományrendszeren található adatokat tükrözi. Ez egy nagyon idõigényes folyamat, ami rányomja bélyegét a rendszer elindulásához szükséges idõ mennyiségére is. Amennyiben szeretnénk megtakarítani ezt a lépést, tegyük bele az /etc/rc.conf állományba a direkt erre a célra kialakított beállítást:

```
check_quotas="NO"
```

Végezetül az állományrendszereken az /etc/fstab megfelelõ módosításával tudjuk egyenként engedélyezni a lemezkvóták használatát. Itt lehet bekapcsolni az állományrendszerek felhasználókra vagy csoportokra, esetleg mind a kettõjükre vonatkozó kvótáikat.

Ha felhasználói szintû kvótákat akarunk engedélyezni egy állományrendszeren, akkor az /etc/fstab állományban az állományrendszer beállításai közé vegyük fel a `userquota` opciót. Például így:

```
/dev/da1s2g   /home    ufs rw,userquota 1 2
```

Ehhez hasonlóan tudjuk engedélyezni a `userquota` helyett a `groupquota` opció használatával a

csoportszintû kvótákat is. A felhasználói- és csoportszintû kvóták együttes engedélyezéséhez így kell átírni az állományrendszer bejegyzését:

```
/dev/da1s2g    /home    ufs rw,userquota,groupquota 1 2
```

Alapértelmezés szerint az állományrendszerekhez tartozó kvóták a gyökerükben található quota.user valamint quota.group állományokban tárolódnak. Errõl részletesebben az fstab(5) man oldalon olvashatunk. Noha még az fstab(5) man oldala szerint is megadható más elérési út a kvótákat tároló állományokhoz, semmiképpen sem javasoljuk ezt, mert úgy tûnik, hogy a kvótákat kezelõ különbözõ segédprogramok ezzel nem képesek rendesen megbirkózni.

Most kell újraindítani a rendszerünket az új rendszermaggal. Az /etc/rc magától le fogja futtatni a kezdeti kvótaállományok létrehozásához szükséges parancsokat az /etc/fstab állományban megadott állományrendszereken. Ennek megfelelõen tehát nem nekünk kell kézzel létrehoznunk ezeket az állományokat.

Hétköznapi esetben egyáltalán nem kell manuális futtatnunk a quotacheck(8), quotaon(8) vagy quotaoff(8) parancsokat. Habár ha tisztában szeretnénk lenni a pontos mûködésükkel, akkor mindenképpen lapozzuk fel a hozzájuk tartozó man oldalakat.

18.15.2. A kvóták beállítása

Ahogy sikerült beállítani a kvóták használatát, egybõl ellenõrizzük is a mûködõképességüket. Ezt legegyszerûbben a következõ paranccsal tehetjük meg:

```
# quota -v
```

Itt egy sorban összefoglalva láthatjuk a jelenlegi lemezhasználatot és az egyes állományrendszereken engedélyezett kvóták korlátait.

Most már készenállunk arra, hogy az edquota(8) paranccsal végre korlátokat is beállítsunk a kvótákhoz.

Számos beállítás áll rendelkezésünkre a felhasználók vagy csoportok által lefoglalható lemezterület vagy a létrehozható állományok számának korlátozását illetõen. A helyfoglalást szabályozhatjuk lemezterület alapján (blokk kvóta) vagy az állományok száma szerint (állományleíró kvóta), esetleg a kettõ kombinációjával. A korlátok további két kategóriára bonthatóak: erõsre és gyengére.

Az erõs korlátot (hard limit) nem lehet túllépni. Ahogy a felhasználó eléri a számára kiszabott erõs korlátot, semmilyen további területet nem használhat fel a kérdéses állományrendszeren. Például, ha a felhasználónak az állományrendszeren 500 kilobyte-os erõs korlátot állítottunk be, és éppen 490 kilobyte-nál tart, akkor a felhasználó innen már csak 10 kilobyte-nyi helyet foglalhat le. 11 kilobyte lefoglalása már nem fog sikerrel járni.

Ezzel szemben a gyenge korlátok (soft limit) egy adott ideig átléphetõek. Ezt az idõt türelmi idõnek (grace period) nevezik, ami alapértelmezés szerint egy hét. Ha a felhasználó a gyenge korláton felül marad a türelmi idõ után is, akkor ezt a gyenge korlát erõssé válik és semmilyen további

helyfoglalásra nem lesz lehetõsége. Amikor a felhasználók újra a gyenge korlát alá kerül, a türelmi idõ is visszaáll a beállított értékére.

A most következõ példában az edquota(8) parancsot mutatjuk be. Amikor meghívjuk az edquota(8) parancsot, akkor elindul az `EDITOR` környezeti változónak megfelelõ szövegszerkesztõ, illetve ennek hiányában a vi, és lehetõségünk nyílik a kvóta korlátainak módosítására.

```
# edquota -u teszt
```

```
Quotas for user teszt:
/usr: kbytes in use: 65, limits (soft = 50, hard = 75)
        inodes in use: 7, limits (soft = 50, hard = 60)
/usr/var: kbytes in use: 0, limits (soft = 50, hard = 75)
        inodes in use: 0, limits (soft = 50, hard = 60)
```

Normális esetben minden kvótával rendelkezõ állományrendszerhez két sort kapunk. Közülük az egyik sorban szerepelnek a blokkok korlátai, a másikban az állományleírók korlátai. Ha valamelyiküket meg akarjuk változtatni, akkor egyszerûen csak át kell írnunk az adott korlát értékét. Például növeljük meg a felhasználók 50-es gyenge és 75-ös erõs blokk korlátját 500-as gyenge és 600-as erõs korlátra. Ehhez szerkesszük át a

```
/usr: kbytes in use: 65, limits (soft = 50, hard = 75)
```

sort erre:

```
/usr: kbytes in use: 65, limits (soft = 500, hard = 600)
```

Az új korlátok akkor fognak érvénybe lépni, miután kiléptünk a szövegszerkesztõbõl.

Néha hasznos lehet a korlátokat adott felhasználói azonosítókhoz beállítani. Ezt az edquota(8) parancs `-p` paraméterével tudjuk elvégezni. Elõször is állítsuk be egy felhasználónak a beállítani kívánt korlátokat, majd futtassuk le az `edquota -p teszt kezdõuid-véguid` parancsot. Például ha a `teszt` nevû felhasználónak állítottuk be a számunkra megfelelõ korlátokat, akkor a következõ paranccsal lehet a rá vonatkozó korlátokat kiterjeszteni a 10 000 és 19 999 közötti azonosítójú felhasználókra:

```
# edquota -p teszt 10000-19999
```

Errõl bõvebben az edquota(8) man oldalán kaphatunk felvilágosítást.

18.15.3. A kvóták korlátainak és a lemezhasználat ellenõrzése

A kvóták korlátait és a lemez jelenlegi kihasználtságát a quota(1) vagy repquota(8) parancsokkal is ellenõrizhetjük. A quota(1) parancs segítségével ellenõrizhetõ az egyes felhasználók vagy csoportok

kvótája és lemezhasználata. A felhasználók csak a saját adataikhoz férhetnek hozzá, illetve mindazon csoportokéhoz, aminek tagjai. Egyedül a rendszeradminisztrátor képes látni az összes felhasználó és csoport kvótáját. A repquota(8) paranccsal kérdezhetõ le az összes kvóta és lemezhasználat rövid kimutatása minden olyan állományrendszeren, ahol azok engedélyezettek.

A következõ kimenet a `quota -v` parancstól származik, ahol a felhasználónak két állományrendszeren is vannak kvótái:

```
Disk quotas for user teszt (uid 1002):
     Filesystem  usage    quota   limit   grace   files   quota   limit   grace
           /usr      65*     50      75   5days       7      50      60
       /usr/var       0      50      75               0      50      60
```

A fenti példában látható, hogy a felhasználó a /usr állományrendszeren pillanatnyilag 15 kilobyte-tal van az 50 kilobyte-os gyenge korlátja felett és 5 napja van hátra a türelmi idõbõl. Vegyük észre a szám mellett levõ csillagot (*), amivel a rendszer jelzi, hogy a felhasználó túllépte a korlátját.

A quota(1) parancs kimenetében általában nem jelennek meg azok az állományrendszerek, amelyeken a felhasználónak ugyan vannak kvótái, de nem foglal rajtuk lemezterületet. A `-v` beállítás megadásával ezek az állományrendszerek is láthatóvá válnak, mint ahogy azt a fenti példában is megfigyelhettük a /usr/var esetében.

18.15.4. Kvóták NFS-en keresztül

A kvóták az NFS szerver kvótákért felelõs alrendszerében is engedélyezhetõek. Az rpc.rquotad(8) démon teszi az NFS klienseken futtatott quota(1) parancsok számára elérhetõvé a kvótákkal kapcsolatos információkat, aminek köszönhetõen a felhasználók távolról is képesek lekérdezni a kvótáikat.

Az `rpc.rquotad` aktivilásához a következõt kell beállítani az /etc/inetd.conf állományban:

```
rquotad/1      dgram rpc/udp wait root /usr/libexec/rpc.rquotad rpc.rquotad
```

Majd ne felejtsük el újraindítani az `inetd` démont sem:

```
# /etc/rc.d/inetd restart
```

18.16. A lemezpartíciók titkosítása

A FreeBSD kitûnõ futásközbeni védelmet ajánl fel az adatok illetéktelen hozzáférése ellen. Az állományok engedélyei és a kötelezõ hozzáférés-vezérlés (Mandatory Access Control, MAC, lásd Kötelező hozzáférés-vezérlés (MAC)) segítenek megvédeni érzékeny adatainkat az illéktelenek ellen az operációs rendszer futása és a számítógép mûködése során. Azonban az operációs rendszerben kezelt engedélyek teljesen hatástalanok abban az esetben, ha a támadó fizikailag is képes hozzáférni a számítógépünkhöz, eltávolítani a merevlemezt és egy másik operációs rendszer segítségével kielemezni a rajta található fontos adatainkat.

Függetlenül attól, hogy a támadó valójában miként is férkõzött hozzá a merevlemezünkhöz, vagy miként kapcsolta le a számítógépünket, a FreeBSD megtalálható GEOM alapú lemeztitkosítás (gbde) és a `geli` titkosítási alrendszer egyaránt képes védelmet nyújtani a számítógépen található állományrendszerek számára az értékes adatok után kutató igen motivált betörõk ellen. A csupán egyes állományokra kiterjedõ körmönfont titkosítási módszerekkel szemben a `gbde` és a `geli` az egész állományrendszert észrevétlen módon titkosítja. Titkosítatlan adat nem is kerül a merevlemezre.

18.16.1. A lemez titkosítása a gbde használatával

1. Váljunk `root` felhasználóvá

   A gbde beállításához rendszeradminisztrátori jogosultságokra lesz szükségünk.

   ```
   % su -
   Password:
   ```

2. Adjuk hozzá a gbde(4) támogatását a rendszermag konfigurációs állományához

   Tegyük a következõ sort a rendszermag beállításait tartalmazó állományba:

   ```
   options GEOM_BDE
   ```

   Fordítsuk újra a rendszermagot a A FreeBSD rendszermag testreszabásaben leírtak szerint.

   Indítsuk el a számítógépet az új rendszermaggal.

3. A rendszermag újrafordítása helyett a `kldload` paranccsal is betölthetjük a gbde(4) modulját:

   ```
   # kldload geom_bde
   ```

A titkosított merevlemez elõkészítése

A következõ példa azt feltételezi, hogy a rendszerünkhöz egy új merevlemezt adunk hozzá, amin egyetlen titkosított partíció foglal helyet. Ezt a partíciót a /private könyvtárba fogjuk csatlakoztatni. A gbde használható a /home és a /var/mail titkosítására is, de ennek megvalósítása olyan bonyolult utasításokat igényel, amelyek meghaladják ennek a bevezetésnek a kereteit.

1. Az új merevlemez hozzáadása

   A Lemezek hozzáadásaban bemutatottak szerint adjuk hozzá a rendszerünkhöz az új merevlemezt. A példában az új lemez partícióját a /dev/ad4s1c néven fogjuk tudni elérni. A /dev/ad0s1* eszközök a példában szereplõ FreeBSD rendszer szabványos partícióit jelölik.

   ```
   # ls /dev/ad*
   ```

```
/dev/ad0        /dev/ad0s1b     /dev/ad0s1e     /dev/ad4s1
/dev/ad0s1      /dev/ad0s1c     /dev/ad0s1f     /dev/ad4s1c
/dev/ad0s1a     /dev/ad0s1d     /dev/ad4
```

2. Hozzunk létre egy könyvtárat a gbde zárolásainak tárolásához

```
# mkdir /etc/gbde
```

A gbdenek azért van szüksége a zárolásokat rögzítõ állományokra, hogy hozzá tudjon férni a titkosított partíciókhoz. Amennyiben ezt nem tudja megtenni, a gbde anélkül nem lesz képes visszafejteni a titkosított partíciókon tárolt adatokat, hogy az ezeket elérni akaró szoftvereknek ne kelljen jelentõsebb mértékben manuálisan beavatkoznia. Mindegyik titkosított partíció külön zároló állományt használ.

3. A gbde partíció inicializálása

A gbde által használt partíciókat használatuk elõtt inicializálni kell. Ezt a mûveletet azonban csak egyszer kell elvégezni:

```
# gbde init /dev/ad4s1c -i -L /etc/gbde/ad4s1c.lock
```

A gbde(8) ekkor elindít egy szövegszerkesztõt és benne egy sablon segítségével be tudjuk állítani a különbözõ konfigurációs értékeket. Az UFS1 vagy UFS2 használata esetén állítsuk a szektorméretet 2048-ra:

```
$FreeBSD: src/sbin/gbde/template.txt,v 1.1 2002/10/20 11:16:13 phk Exp $
#
# Sector size is the smallest unit of data which can be read or written.
# Making it too small decreases performance and decreases available space.
# Making it too large may prevent filesystems from working.  512 is the
# minimum and always safe.  For UFS, use the fragment size
#
sector_size     =       2048
[...]
```

A megjegyzés fordítása:

```
A szektorméret az adatok írásának és olvasásának legkisebb egysége.  Ha
túlságosan kicsire választjuk meg, akkor csökken a teljesítmény és csökken a
rendelkezésre álló hely.  Ha viszont túlságosan nagyra hagyjuk, akkor azzal
akadályozzuk az állományrendszerek munkáját.  512 a legkisebb érték, amely
mindig
megbízható.  Az UFS esetén használjuk a fragmensek méretét.
```

A gbde(8) kétszer is rá fog kérdeni az adatok titkosítására használt jelmondatra. A

jelmondatnak természetesen mind a kétszer ugyanannak kell lennie. A gbde védelmének hatékonysága teljesen mértékben az általunk választott jelmondat minõségétõl függ .

A `gbde init` parancs létrehoz egy záró állományt a gbde partícióhoz, amely ebben a példában az /etc/gbde/ad4s1c.lock néven keletkezett. A gbde záró állományainak ".lock" névre kell végzõdniük, mivel az /etc/rc.d/gbde indítószkript csak ebben az esetben észleli rendesen.

A gbde záró állományait a titkosított partíciók tartalmával együtt *kell* lementeni. Miközben a záró állomány törlése nem tudja megakadályozni, hogy az elszánt támadó visszafejtse a gbde által titkosított partíciót, addig a záró állomány nélkül a jogos tulajdonos órási mennyiségû munka befektetése nélkül képtelen lesz hozzáférni a rajta levõ adatokhoz. Ez utóbbitól egyébként a gbde(8) és a rendszer tervezõje is totálisan elhatárolja magát.

4. A titkosított partíció illesztése a rendszermaghoz

```
# gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c.lock
```

Ekkor a titkosított partíció illesztéséhez a rendszer kérni fogja az inicializálás során választott jelmondatot. Ezután az új titkosított eszköz megjelenik a /dev könyvtárban /dev/eszköznév.bde néven:

```
# ls /dev/ad*
/dev/ad0        /dev/ad0s1b     /dev/ad0s1e     /dev/ad4s1
/dev/ad0s1      /dev/ad0s1c     /dev/ad0s1f     /dev/ad4s1c
/dev/ad0s1a     /dev/ad0s1d     /dev/ad4        /dev/ad4s1c.bde
```

5. Állományrendszer kialakítása egy titkosított eszközön

Ahogy sikerült a titkosított eszközt illeszteni a rendszermaghoz, létre is tudunk hozni egy állományrendszert rajta. Erre a célra a newfs(8) remekül használható. Mivel egy új UFS2 állományrendszerek inicializálása sokkal gyorsabb a régi UFS1 állományrendszerek inicializálásánál, ezért a newfs(8) használata esetén az `-O2` beállítás megadása ajánlott.

```
# newfs -U -O2 /dev/ad4s1c.bde
```

A newfs(8) parancsot egy illesztett gbde partíción kell végrehajtani, amit onnan ismerhetünk meg, hogy az eszköz nevében szerepel a *.bde kiterjesztés.

6. A titkosított partíció csatlakoztatása

Hozzunk létre egy csatlakozási pontot a titkosított állományrendszer számára.

```
# mkdir /privát
```

Csatlakoztassuk a titkosított állományrendszert.

```
# mount /dev/ad4s1c.bde /privát
```

7. Ellenõrizzük a titkosított állományrendszer mûködõképességét

   A titkosított állományrendszert most már látja a df(1) program és készen áll a használatra.

```
% df -H
Filesystem        Size   Used  Avail Capacity  Mounted on
/dev/ad0s1a      1037M    72M   883M     8%    /
/devfs            1.0K   1.0K     0B   100%    /dev
/dev/ad0s1f       8.1G    55K   7.5G     0%    /home
/dev/ad0s1e      1037M   1.1M   953M     0%    /tmp
/dev/ad0s1d       6.1G   1.9G   3.7G    35%    /usr
/dev/ad4s1c.bde   150G   4.1K   138G     0%    /private
```

Létezõ titkosított állományrendszerek csatlakoztatása

A rendszer minden egyes indítása után az összes titkosított állományrendszert tényleges használata elõtt újra illeszteni kell a rendszermaghoz, ellenõrizni az épségét és csatlakoztatni. Az ehhez szükséges parancsokat `root` felhasználóként kell kiadni.

1. A gbde partíció illesztése a rendszermaghoz

```
# gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c.lock
```

   A gbde partíció inicializálása során megadott jelmondatot kell megadnunk a mûvelet elvégzéséhez.

2. Az állományrendszer épségének ellenõrzése

   Mivel a titkosított állományrendszerek az automatikus csatlakoztatáshoz még nem szerepeltethetõek az /etc/fstab állományban, ezért az ilyen állományrendszereket csatlakoztatásuk elõtt manuálisan ellenõriztetni kell a fsck(8) lefuttatásával.

```
# fsck -p -t ffs /dev/ad4s1c.bde
```

3. A titkosított állományrendszer csatlakoztatása

```
# mount /dev/ad4s1c.bde /privát
```

A titkosított állományrendszer most már készen áll a használatra.

A titkosított partíciók önálló csatlakoztatása

Lehet írni olyan szkriptet, amely a titkosított partíciókat magától illeszti, ellenõrzi és csatlakoztatja, de biztonsági megfontolásokból semmi esetben sem szabad tartalmaznia a gbde(8) jelszavát. Ehelyett azt javasoljuk, hogy az ilyen szkripteknek külön meg kelljen adni a jelszót konzolon vagy az ssh(1) használatán keresztül.

De használhatjuk a mellékelt rc.d szkriptet is. A szkript paramétereit az rc.conf(5) állományon keresztül adhatjuk meg, például:

```
gbde_autoattach_all="YES"
gbde_devices="ad4s1c"
gbde_lockdir="/etc/gbde"
```

Ilyenkor a gbde által használt jelmondatot a rendszer indításakor kell megadni. Miután begépeltük a megfelelõ jelmondatot, a titkosított gbde partíció magától csatlakoztatásra kerül. Ez akkor lehet hasznos, ha a gbde megoldását hordozható számítógépeken alkalmazzuk.

A gbde által alkalmazott titkosítási módszerek

A gbde(8) a szektorok tartalmát 128 bites AES használatával CBC módban titkosítja. A lemezen található minden egyes szektort eltérõ AES kulccsal kódolja. A gbde kriptográfiai felépítését, valamint mindazt, hogy az egyes szektorok kulcsai miként származtathatóak a felhasználó által megadott jelmondatból, a gbde(4) man oldalán olvashatjuk.

Kompatibilitási problémák

A sysinstall(8) nem kompatibilis a gbde által titkosított eszközökkel. A sysinstall(8) indítása elõtt minden *.bde eszközt ki kell iktatni a rendszermagból, különben az eszközök keresése során össze fog omlani. A példánkban használt titkosított eszközt a következõ paranccsal kell lekapcsolni:

```
# gbde detach /dev/ad4s1c
```

Továbbá megjegyezzük azt is, hogy a vinum(4) nem használja a geom(4) alrendszert, ezért a gbde alkalmazása során nem használhatunk Vinum-köteteket.

18.16.2. A lemezek titkosítása a `geli` használatával

A FreeBSD 6.0 változatától kezdve egy új kriptográfiai GEOM osztály is a rendelkezésünkre áll, melyet pillanatnyilag Paweł Jakub Dawidek <pjd@FreeBSD.org> fejleszt. A `geli` segédprogram némileg különbözõ a `gbde` megoldásától - más lehetõségeket kínál fel és a titkosítást is egy eltérõ séma mentén valósítja meg.

A geli(8) legfontosabb jellemzõi a következõk:

- A crypto(9) keretrendszerét használja - tehát ha rendelkezünk kriptográfiai hardverrel, akkor a `geli` automatikusan használni fogja.
- Több kriptográfiai algoritmust is ismer (melyek jelenleg az AES, Blowfish és a 3DES).
- Segítségével a rendszerindításhoz használt (gyökér) partíció is titkosítható. Ilyenkor a szükséges jelmondatot a rendszer indításakor kell megadni.
- Két független kulcsot (például egy "kulcsot" és egy "céges kulcsot") is használhatunk vele.
- A `geli` gyors - egyszerûen csak szektorról szektorra titkosít.
- Lehetõvé teszi a mesterkulcsok mentését is visszaállítását. Ha a felhasználó véletlenül megsemmisítené a kulcsát, akkor a biztonsági mentésbõl helyreállított kulcsok segítségével vissza tudjuk szerezni az adatainkat is.
- Segítségével a lemezeket véletlenszerû, egyszeri jelszavakkal is illeszthetjük - ez különösen fontos lapozóterületek és ideiglenes állományrendszerek esetében.

A `geli` által felkínált lehetõségekrõl a geli(8) man oldalán találhatunk többet.

A következõ lépések bemutatják, hogyan lehet a FreeBSD rendszermagjában engedélyezni a `geli` támogatását, és hogyan lehet létrehozni és használni egy `geli` titkosítással rendelkezõ adathordozót.

A `geli` alkalmazásához legalább a FreeBSD 6.0-RELEASE vagy késõbbi változatára van szükségünk. Mivel a rendszermagot is módosítanunk kell, ezért rendszeradminisztrátori jogosultságok kellenek a mûveletek elvégzéséhez.

1. A `geli` támogatásának hozzáadása a rendszermaghoz

   Vegyük hozzá a következõ sorokat a rendszermag beállításait tartalmazó állományhoz:

   ```
   options GEOM_ELI
   device crypto
   ```

   Fordítsuk újra a rendszermagot a A FreeBSD rendszermag testreszabásaben leírtak szerint.

   Betölthetjük a `geli` modulját is a rendszer indításakor. Ehhez a következõ sort kell betenni a /boot/loader.conf állományba:

   ```
   geom_eli_load="YES"
   ```

   A geli(8) most már használható a rendszermagban.

2. A mesterkulcs legenerálása

   A most következõ példában egy kulcsot tartalmazó állomány létrehozását illusztráljuk, amit a /privát könyvtárba csatlakoztatott titkosított adathordozó mesterkulcsához fogunk

használni. A kulcs állomány a mesterkulcs titkosításához felhasznált véletlenszerû adatot fogja tartalmazni, valamint rajta kívül még a mesterkulcsot egy jelmondattal is védjük. Az adathordozó szektormérete 4 kilobyte-os lesz. Emellett még bemutatjuk, hogyan kell illeszteni egy `geli`-adathordozót, állományrendszert létrehozni rajta, csatlakoztatni, dolgozni vele és lekapcsolni.

A nagyobb teljesítmény érdekében javasolt nagyobb szektorméretet választani (mint például 4 kilobyte).

A mesterkulcsot egy jelmondattal fogjuk védeni és a kulcsok készítéséhez használt adatforrás a /dev/random lesz. A /dev/da2.eli, amelyet mit csak adathordozónak fogunk csak hívni, szektorainak mérete 4 kilobyte lesz.

```
# dd if=/dev/random of=/root/da2.key bs=64 count=1
# geli init -s 4096 -K /root/da2.key /dev/da2
Enter new passphrase:
Reenter new passphrase:
```

Nem kötelezõ egyszerre használni a jelmondatot és a kulcs állományt. A mesterkulcs elzárásának bebiztosítására bármelyik módszer alkalmas.

Ha a kulcs állomány a "-" paraméterrel adjuk meg, akkor a szabványos bemenetrõl olvassa be a program. Ez a példa több kulcs használatát mutatja be.

```
# cat kulcs1 kulcs2 kulcs3 | geli init -K - /dev/da2
```

3. Az adathordozó illesztése a generált kulccsal

```
# geli attach -k /root/da2.key /dev/da2
Enter passphrase:
```

Az új titkosítatlan eszköz neve /dev/da2.eli lesz.

```
# ls /dev/da2*
/dev/da2  /dev/da2.eli
```

4. Az új állományrendszer kialakítása

```
# dd if=/dev/random of=/dev/da2.eli bs=1m
# newfs /dev/da2.eli
# mount /dev/da2.eli /privát
```

A titkosított állományrendszer most már df(1) számára is látszik és használható:

```
# df -H
Filesystem     Size   Used  Avail Capacity  Mounted on
/dev/ad0s1a    248M    89M   139M    38%    /
/devfs         1.0K   1.0K     0B   100%    /dev
/dev/ad0s1f    7.7G   2.3G   4.9G    32%    /usr
/dev/ad0s1d    989M   1.5M   909M     0%    /tmp
/dev/ad0s1e    3.9G   1.3G   2.3G    35%    /var
/dev/da2.eli   150G   4.1K   138G     0%    /private
```

5. Az adathordozó leválasztása és lekapcsolása

   Miután befejeztük a munkát a titkosított partíción, és a /privát partícióra már nincs tovább szükségünk, érdemes leválasztanunk és kiiktatnunk a `geli` titkosítású partíciót a rendszermagból.

   ```
   # umount /privát
   # geli detach da2.eli
   ```

A geli(8) használatáról bõvebben a saját man oldalán tájékozódhatunk.

A geli rc.d szkriptjének használata

A `geli` mellett találhatunk egy saját rc.d szkriptet, amely jelentõsen leegyszerûsíti a `geli` használatát. A `geli` például így paraméterezhetõ az rc.conf(5) állományon keresztül:

```
geli_devices="da2"
geli_da2_flags="-p -k /root/da2.key"
```

Ennek segítségével a /dev/da2 eszközt `geli` adathordozóként állítjuk be a /root/da2.key állományban található mesterkulcs felhasználásával, de az illesztéskor a `geli` nem kér jelmondatot (ezt csak akkor fogja tenni, ha a `geli init` parancs kiadásához hozzátesszük a `-P` beállítást). A rendszer leállítása elõtt pedig a `geli` adathordozó így automatikusan leválasztásra kerül.

Az rc.d beállításával kapcsolatos tudnivalókat a kézikönyv rc.d szkriptekrõl szóló szakaszában ismerhetjük meg.

18.17. A lapozóterület titkosítása

A FreeBSD-ben a lapozóterület titkosítása nagyon könnyen beállítható és már a FreeBSD 5.3-RELEASE változata óta elérhetõ. Attól függõen, hogy konkrétan a FreeBSD melyik verzióját használjuk, a konfigurációhoz kapcsolódó beállítások némileg eltérhetnek. A FreeBSD 6.0-RELEASE változatától kezdõdõen a gbde(8) és a geli(8) alrendszerek is használhatóak a lapozóterület titkosítására. A korábbi verziókban egyedül csak a gbde(8) érhetõ el. Mind a két rendszer az encswap rc.d szkriptet használja.

Az elõzõ szakaszban, vagyis a A lemezpartíciók titkosításában már röviden összefoglaltuk a különbözõ titkosítással foglalkozó alrendszereket.

18.17.1. Miért kellene titkosítanunk a lapozóterületet?

Hasonlóan a lemezpartíciók titkosításához, a lapozóterület titkosításának is az a célja, hogy védjük az érzékeny információkat. Képzeljük el, hogy egy olyan alkalmazással dolgozunk, amely jelszavakat kezel. Amíg ezek a jelszavak a memóriában maradnak, addig minden a legnagyobb rendben van. Azonban amikor az operációs rendszer nekilát a fizikai memória felszabadításához kilapozni ezeket az adatokat, a jelszavak titkosítatlanul kerülnek a lemez felületére és egy támadó számára könnyû prédává válnak. Ilyen helyzetekben csak lapozóterület titkosítása jelenthet megoldást.

18.17.2. Elõkészületek

A szakasz további részében a ad0s1b lesz a lapozásra használt partíció.

Egészen mostanáig nem titkosítottuk a lapozóterületet. Így elképzelhetõ, hogy a lemezre már titkosítatlanul kikerültek jelszavak vagy bármilyen más érzékeny adatok. A csorba kiköszörülésére a lapozóterületen található összes adatot írjuk felül véletlenszerûen generált szeméttel:

```
# dd if=/dev/random of=/dev/ad0s1b bs=1m
```

18.17.3. A lapozóterület titkosítása a gbde(8) használatával

Ha a FreeBSD 6.0-RELEASE vagy újabb változatát használjuk, akkor az /etc/fstab állományban tegyük hozzá a `.bde` utótagot az a lapozóterülethez tartozó eszköz nevéhez.

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
/dev/ad0s1b.bde         none            swap    sw              0       0
```

A FreeBSD 6.0-RELEASE elõtti kiadások esetében a következõ sort is hozzá kell tennünk az /etc/rc.conf állományhoz:

```
gbde_swap_enable="YES"
```

18.17.4. A lapozóterület titkosítása a geli(8) használatával

A gbde(8) használatához hasonlóan a geli(8) által felajánlott titkosítást is alkalmazhatjuk a lapozóterület védelmére. Ilyenkor az /etc/fstab állományban az `.eli` utótagot kell hozzátenni a lapozóterülethez tartozó eszköz névhez.

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
/dev/ad0s1b.eli         none            swap    sw              0       0
```

Az geli(8) az AES algoritmust alapértelmezés szerint 256 bites kulccsal használja.

Ezek az alapértelmezések megváltoztathatóak az /etc/rc.conf állományban a `geli_swap_flags` beállítás használatával. A következõ sor arra utasítja az encswap rc.d szkriptet, hogy a geli(8) és a Blowfish algoritmus használatával hozzon létre egy lapozópartíciót 128 bites kulccsal, 4 kilobyte-os szektorméretttel és a "detach on last close" ("lekapcsolás használat után") beállítással:

```
geli_swap_flags="-e blowfish -l 128 -s 4096 -d"
```

A FreeBSD 6.2-RELEASE verzió elõtti rendszerekben a következõ sort kell használni:

```
geli_swap_flags="-a blowfish -l 128 -s 4096 -d"
```

A többi beállításhoz a geli(8) man oldalán a `onetime` parancs leírását érdemes áttanulmányozni.

18.17.5. Ellenõrizzük a mûködését

Miután újraindítottuk a rendszert, a titkosított lapozóterület helyes mûködését a `swapinfo` paranccsal ellenõrizhetjük le.

A gbde(8) esetében:

```
% swapinfo
Device          1K-blocks     Used    Avail Capacity
/dev/ad0s1b.bde    542720        0   542720     0%
```

Valamint a geli(8) esetében:

```
% swapinfo
Device          1K-blocks     Used    Avail Capacity
/dev/ad0s1b.eli    542720        0   542720     0%
```

Chapter 19. GEOM: A moduláris lemezszervező rendszer

19.1. Áttekintés

Ez a fejezet a FreeBSD-ben található GEOM rendszert mutatja be. Ez a rendszer tömöríti az általa is alkalmazott fontosabb RAID-vezérlõ segédprogramokat. A fejezet nem részletezi, hogy a GEOM konkrétan milyen módon kezeli és vezérli az I/O-t, ahogy azt sem, hogyan mûködik az alapjául szolgáló alrendszer vagy hogy néz ki annak forráskódja. Az ilyen jellegû információk a geom(4) man oldalon, valamint az ott felsorolt helyeken találhatóak meg. Továbbá, ez a fejezet magukról a RAID-konfigurációkról sem ad pontos tájékoztatást. Kizárólag csak a GEOM által is támogatott RAID-besorolásokról esik szó.

A fejezet elolvasása során megismerjük:

- a GEOM segítségével milyen fajtájú RAID támogatást érhetünk el;
- hogyan kell használni a rendszer által nyújtott alapvetõ segédeszközöket a különféle RAID-szintek konfigurálásához, karbantartásához és kezeléséhez;
- hogyan kell a GEOM-on keresztül tükrözni, csíkozni, titkosítani és távolról összekapcsolni lemezes eszközöket;
- hogyan kell a GEOM rendszerben összekapcsolt lemezeknél felmerülõ hibákat felderíteni.

A fejezet elolvasásához ajánlott:

- megérteni, hogyan kezeli a FreeBSD a lemezes eszközöket (Háttértárak);
- ismerni, hogyan konfiguráljunk és telepítsünk egy új FreeBSD rendszermagot (A FreeBSD rendszermag testreszabása).

19.2. A GEOM bemutatása

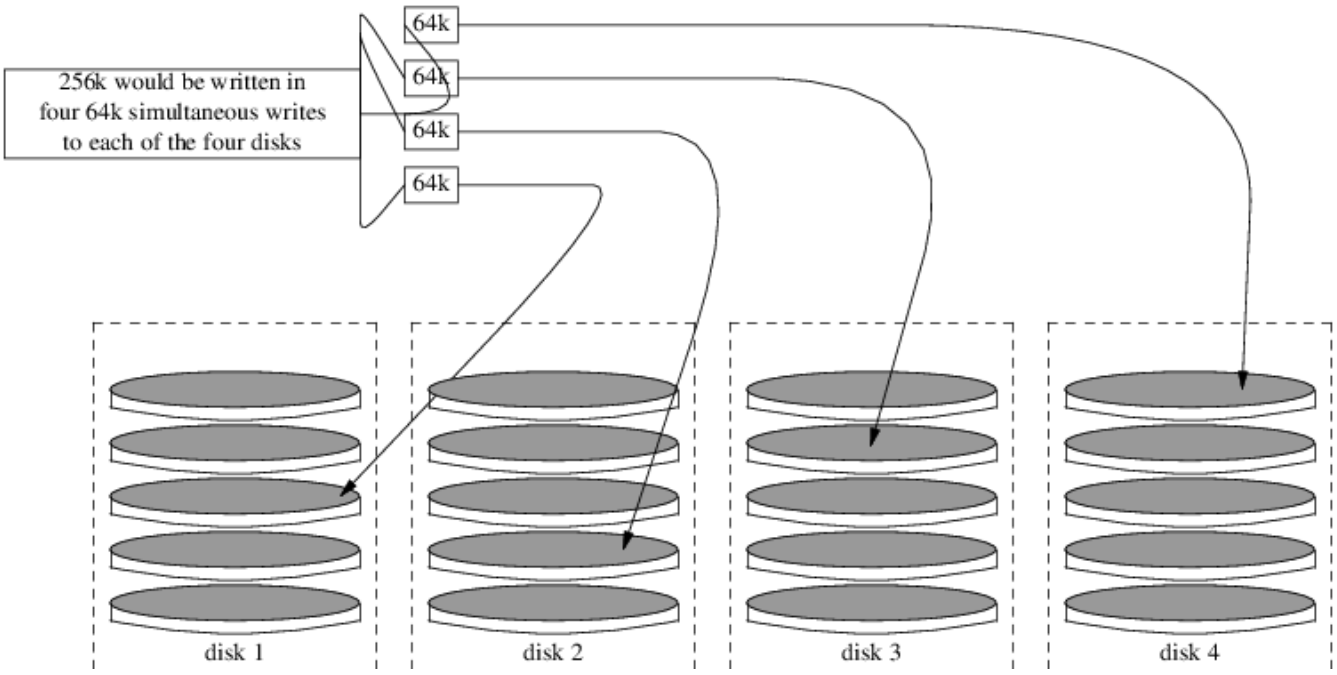
A GEOM rendszer adatszolgáltatókon vagy speciális /dev-állományokon keresztül hozzáférést és vezérlést tesz lehetõvé bizonyos osztályokhoz - Master Boot Recordokhoz, BSD-címkékhez stb. Számos szoftveres RAID konfiguráció támogatásával a GEOM transzparens elérést tesz lehetõvé mind az operációs rendszer, mind pedig az általa felkínált segédprogramok számára.

19.3. RAID0 - Csíkozás

A csíkozás módszerét használjuk abban az esetben, amikor több lemezmeghajtót akarunk egyetlen kötetté összevonni. A GEOM lemezalrendszer szoftveres támogatást nyújt a RAID0, más néven a lemezcsíkozás megvalósításához.

Egy RAID0 rendszerben az adatokat blokkokra bontva írjuk fel a tömbben található lemezek között szétosztva. Így ahelyett, hogy meg kellene várnunk 256 kb-nyi adat egyetlen lemezre írását, egy RAID0 rendszerben egyszerre íródik 64 kb-nyi adat négy különbözõ lemezre, és ezáltal gyorsabb elérést szolgáltat. Ez a gyorsaság további lemezvezérlõk használatával még jobban fokozható.

Az egy RAID0-csíkozásban résztvevõ lemezek mindegyikének azonos méretûnek kell lennie, mivel az írásra és olvasásra irányuló I/O-kérések a párhuzamos kiszolgálás érdekében összefésülõdnek.



Procedure: Csíkozás kialakítása formázatlan ATA-lemezekkel

1. Töltsük be a geom_stripe.ko modult:

```
# kldload geom_stripe
```

2. Bizonyosodjuk meg róla, hogy a rendszerünkben található egy szabad csatlakozási pont. Ha majd ezt a kötetet szánjuk rendszerünk gyökérpartíciójának, használjunk erre a célra egy másik könyvtárat, például a /mnt-ot:

```
# mkdir /mnt
```

3. Keressük meg a csíkozásra felhasználni kívánt lemezek eszközneveit, és hozzunk létre belõlük egy új csíkozott eszközt. Például, ha két használatban nem levõ, particionálatlan ATA-lemezt, név szerint a /dev/ad2 és /dev/ad3 eszközöket akarjunk csíkozni:

```
# gstripe label -v st0 /dev/ad2 /dev/ad3
Metadata value stored on /dev/ad2.
Metadata value stored on /dev/ad3.
Done.
```

4. Az így létrejött új köteten most hozzunk létre egy általános címkét, vagy más néven egy partíciós táblát, és telepítsük fel rá a rendszer alapértelmezett rendszerindító programját:

```
# bsdlabel -wB /dev/stripe/st0
```

5. Ezzel meg kellett jelennie további másik két eszköznek is a /dev/stripe könyvtárban, a st0 eszköz mellett. Ezek többek közt az st0a és az st0c. Itt már ki is tudunk alakítani egy állományrendszert az st0a eszközön a `newfs` használatával:

   ```
   # newfs -U /dev/stripe/st0a
   ```

   Sok-sok számot fogunk látni cikázni a képernyõn, majd néhány másodperc múlva befejezõdik a folyamat. Létrehoztuk a kötetet, ami most már készen áll a becsatolásra.

A kialakított lemezcsíkozást így tudjuk kézzel csatlakoztatni:

```
# mount /dev/stripe/st0a /mnt
```

A csíkozott állományrendszert a rendszerindítás folyamán automatikusan becsatlakoztathatjuk, ha elhelyezzük az alábbi kötetinformációkat az /etc/fstab állományba. Erre a célra stripe néven létrehozunk egy állandó csatlakozási pontot:

```
# mkdir /stripe
# echo "/dev/stripe/st0a /stripe ufs rw 2 2" \
      >> /etc/fstab
```

A geom_stripe.ko modult is automatikusan be kell tölteni a rendszerindítás során. Ehhez a következõ sort kell hozzáadni a /boot/loader.conf állományhoz:

```
# echo 'geom_stripe_load="YES"' >> /boot/loader.conf
```

19.4. RAID1 - Tükrözés

A tükrözés számos vállalatnál és háztartásban alkalmazott technológia, amely az adatok megszakítás nélküli lementésére használatos. Amikor tükrözést használunk, az egyszerûen csak arra utal, hogy a B lemez ugyanazokat az adatokat tartalmazza, mint az A lemez. Vagy amikor a C és D lemez tartalma egyezik meg az A és B lemezekével. Függetlenül a lemezek kiosztásától, itt az a lényeg, hogy az egyik lemez teljes területe vagy az egyik partíciója le van másolva. Késõbb az ezen a módon lementett adatok könnyen visszaállíthatóak anélkül, hogy ez a szolgáltatásban vagy az elérhetõségben bármilyen kimaradást okozna, és akár még fizikailag is biztonságosan tárolhatóak.

Elõször is szereznünk kell két egyforma méretû lemezt, valamint a példák feltételezik, hogy ezek a lemezek közvetlen elérésû (da(4)) SCSI-lemezek.

19.4.1. Az elsõdleges lemezek tükrözése

Tegyük fel, hogy a FreeBSD az elsõ, da0 nevû lemezmeghajtón található, és a gmirror(8) számára ezt szeretnénk megadni az elsõdleges adatok tárolásához.

A tükrözés létrehozásának megkezdése elõtt a `kern.geom.debugflags` sysctl(8) változó megfelelõ beállításával engedélyezzünk további nyomkövetési információkat és hozzáférést az eszközhöz:

```
# sysctl kern.geom.debugflags=17
```

Most építsük fel a tükrözést. Kezdjük az egészet a metaadatok elhelyezésével az elsõdleges lemezmeghajtón, tehát tulajdonképpen az alábbi parancs segítségével hozzuk létre a /dev/mirror/gm eszközt:

A rendszerindító meghajtóról készített tükrözés adatvesztést okozhat a lemez utolsó szektorában. Ennek kockázata csökkenthetõ, ha közvetlenül a FreeBSD friss telepítése után állítjuk be a tükrözést.

```
# gmirror label -vb round-robin gm0 /dev/da0
```

Erre a rendszernek a következõ módon kell reagálnia:

```
Metadata value stored on /dev/da0.
Done.
```

A GEOM inicializálásához szükségünk lesz a /boot/kernel/geom_mirror.ko modul betöltésére:

```
# gmirror load
```

A parancs sikeres lefutása után a /dev/mirror könyvtárban létrehoz egy gm0 eszközleírót.

A geom_mirror.ko modul betöltését így tudjuk engedélyezni a rendszer indításakor:

```
# echo 'geom_mirror_load="YES"' >> /boot/loader.conf
```

Nyissuk meg az /etc/fstab állományt, és cseréljük le benne az összes korábbi da0 hivatkozást az újonnan kialakított gm0 tükrözés eszközleírójával.

Ha vi(1) szövegszerkesztõt használjuk, akkor a következõ módon tudjuk ezt egyszerûen megtenni:

```
# vi /etc/fstab
```

A vi(1) indítása után a `:w /etc/fstab.bak` kiadásával készítsünk az fstab állomány jelenlegi tartalmáról másolatot. Ezután a `:%s/da/mirror\/gm/g` parancs használatával cseréljük ki az összes da0 hivatkozást a gm0 eszköz nevére.

Az így keletkezõ fstab állomány nagyjából következõ módon fog kinézni. Most teljesen független, hogy SCSI vagy ATA meghajtókkal dolgozunk, a RAID eszköz neve mindig gm lesz:

```
# Eszköz                Csatlakozási pont   Típus   Beállítások  Dump   Menet
/dev/mirror/gm0s1b      none                swap    sw           0      0
/dev/mirror/gm0s1a      /                   ufs     rw           1      1
/dev/mirror/gm0s1d      /usr                ufs     rw           0      0
/dev/mirror/gm0s1f      /home               ufs     rw           2      2
#/dev/mirror/gm0s2d     /store              ufs     rw           2      2
/dev/mirror/gm0s1e      /var                ufs     rw           2      2
/dev/acd0               /cdrom              cd9660  ro,noauto    0      0
```

Indítsuk újra a rendszert:

```
# shutdown -r now
```

Ennek megfelelõen a rendszer indítása közben a da0 eszköz helyett a gm0 eszközt fogjuk használni. Miután sikeresen befejezõdött a rendszerindítás, a `mount` parancs kiadásával a saját szemünkkel is meggyõzõdhetünk az eredményrõl:

```
# mount
Filesystem         1K-blocks    Used    Avail Capacity  Mounted on
/dev/mirror/gm0s1a   1012974  224604   707334    24%    /
devfs                      1       1        0   100%    /dev
/dev/mirror/gm0s1f  45970182   28596 42263972     0%    /home
/dev/mirror/gm0s1d   6090094 1348356  4254532    24%    /usr
/dev/mirror/gm0s1e   3045006 2241420   559986    80%    /var
devfs                      1       1        0   100%    /var/named/dev
```

A parancs kimenete az elvárásainknak megfelelõen remekül néz ki. Zárásképpen a szinkronizálás megkezdéséhez a következõ paranccsal illesszük be a da1 eszközt a tükrözésbe:

```
# gmirror insert gm0 /dev/da1
```

A tükrözés állapota a létrejöttét követõen az alábbi paranccsal ellenõrizhetõ:

```
# gmirror status
```

Az iménti parancs eredményének nagyjából a következõnek kell lennie miután a felépítettük a tükrözést és szinkronizáltuk az adatokat:

```
      Name    Status  Components
mirror/gm0  COMPLETE  da0
                      da1
```

Hiba esetén a tükrözés továbbra is folytatódik, azonban ilyenkor a példában szereplõ `COMPLETE` helyett a `DEGRADED` jelzést fogjuk látni.

19.4.2. Hibakeresés

A rendszer nem hajlandó elindulni

Ha a rendszerünk ehhez hasonló módon indul:

```
ffs_mountroot: can't find rootvp
Root mount failed: 6
mountroot>
```

Indítsuk újra a gépünket a kikapcsoló gomb vagy a reset segítségével. A rendszerindító menüben válasszuk a hatodik opciót (6). Ennek eredményeképpen megkapjuk a loader(8) parancssorát. Töltsük be a modult manuálisan:

```
OK? load geom_mirror
OK? boot
```

Ha ez beválik, akkor valamiért a modult nem sikerült rendesen betölteni. Ellenõrizzük, hogy a /boot/loader.conf állományban a neki szereplõ megfelelõ bejegyzés helyesen szerepel. Amennyiben a probléma továbbra is fennáll, helyezzük el a következõ sort a rendszermag konfigurációs állományába, majd fordítsuk újra és telepítsük:

```
options GEOM_MIRROR
```

Ezzel várhatóan orvosoltuk a problémát.

19.4.3. A meghibásodott lemezek cseréje

A lemezek tükrözésének egyik legcsodálatosabb elõnye, hogy a menet közben meghibásodott meghajtókat gond, és így feltehetõen adatvesztés nélkül ki tudjuk cserélni.

Vegyük az iménti RAID-1 konfigurációt, és tételezzük fel, hogy a da1 eszköz felmondta a szolgáltatot és cserére szorul. A meghajtó leváltásához keressük meg a hibás eszközt, majd állítsuk le a rendszert. Tegyük be a helyére az újat és indítsuk újra a rendszerünket. Miután elindult az operációs rendszer, a következõ parancsok kiadásával tudjuk logikailag is lecserélni a

meghibásodott lemezt:

```
# gmirror forget gm0
# gmirror insert gm0 /dev/da1
```

Innen a `gmirror status` parancsával kísérhetjük figyelemmel a tükrözés újraszervezésének menetét. Csupán ennyi az egész.

19.5. Eszközök hálózati illesztése a GEOM-ban

A GEOM távoli eszközök, például lemezek, CD-meghajtók stb. használatát is támogatja a hálózati illesztést szolgáló segédprogramjaival, hasonlóan az NFS-hez.

Kezdésként létre kell hozni a megosztást elõsegítõ állományt. Ez az állomány határozza meg, ki és milyen szinten jogosult használni a megosztott erõforrásokat. Például ha megosztjuk az elsõ SCSI-lemezen a negyedik slice-ot, az alábbi /etc/gg.exports állomány tökéletesen megfelel:

```
192.168.1.0/24 RW /dev/da0s4d
```

Ezzel a belsõ hálózaton levõ összes számítógép képes lesz elérni a da0s4d partíción található állományrendszert.

Az eszköz megosztásához elõször gondoskodnunk kell róla, hogy ne legyen csatlakoztatva, majd ezután indítsuk el a ggated(8) szerver démonját:

```
# ggated
```

Ezt követõen a `mount` felhasználásával csatoljuk az eszközt a kliensen, az alábbi parancs kiadásával:

```
# ggatec create -o rw 192.168.1.1 /dev/da0s4d
ggate0
# mount /dev/ggate0 /mnt
```

Innentõl kezdve az eszköz elérhetõ lesz a /mnt csatlakozási ponton keresztül.

Fontos kiemelnünk, hogy ez a mûvelet eredménytelen, ha az adott eszközt vagy maga a szerver, vagy pedig valamelyik másik kliens már korábban csatolta.

Amikor az eszközre már nincs tovább szükségünk, biztonságosan le tudjuk választani az umount(8) paranccsal, hasonlóan bármelyik más lemezes eszközhöz.

19.6. A lemezes eszközök címkézése

A rendszer indítása közben a FreeBSD rendszermagja a talált eszközöknek megfelelõen

mindegyiknek létrehoz egy-egy eszközleírót. Ezzel a próbálgatásos módszerrel együtt jár néhány gond, például mi történik akkor, ha az új lemezes eszközt USB-n keresztül adjuk a rendszerhez? Nagyon valószínû, hogy ez az eszköz megkapja a da0 nevet és ezzel az eredeti da0 eszköz eltolódik a da1 névhez. Ennek köszönhetõen az /etc/fstab állományban felsorolt állományrendszerek csatolása veszélybe kerül, aminek következtében akár meghiúsulhat a rendszerindulás is.

Az egyik lehetséges megoldása a problémának, ha sorbafûzzük a SCSI eszközeinket, és így a SCSI-kártyához kapcsolt újabb eszköz egy addig nem használt számot fog birtokba venni. Mi helyzet azonban az USB-s eszközökkel, amelyek kiüthetik az elsõdleges SCSI-lemezeinket? Ez egyébként azért történhet meg, mert az USB-s eszközöket általában hamarabb keresi a rendszer, mint a SCSI kártyán levõ eszközöket. Megoldhatjuk úgy ezt a gondot, hogy csak azután csatlakoztatjuk az említett eszközöket, miután a rendszer elindult. Megoldhatjuk viszont úgy is, hogy csak egyetlen ATA-meghajtót használunk és soha nem soroljuk fel a SCSI eszközöket az /etc/fstab állományban.

Ezeknél kínálkozik azonban egy jobb megoldás! A `glabel` nevû segédprogrammal a rendszergazda vagy a felhasználó úgy tudja címkézni a lemezmeghajtókat, hogy azok a /etc/fstab állományban szereplõ címkéket használják. Mivel a `glabel` a címkét az adott szolgáltató utolsó szektorában tárolja el, ez a címke megmarad az újraindítás után is. Ha ezt a címkét eszközként használjuk, az állományrendszerek mindig ugyanarról a meghajtóról fognak csatolódni, függetlenül attól, hogy milyen eszközleírón keresztül érjük el ezeket.

Egyáltalán nem állítottuk, hogy egy címke csak állandó lehet. A `glabel` segítségével egyaránt létre lehet hozni állandó és átmeneti címkéket, de csak az állandó címke képes az újraindítás után is megmaradni. A két címketípus közti különbségeket a glabel(8) man oldal tárgyalja részletesebben.

19.6.1. Címketípusok és példák

A címkéknek két típusa létezik, az általános címke és az állományrendszer-címke. A címkék lehetnek állandóak vagy ideiglenesek. Az állandó címkék a tunefs(8) vagy newfs(8) parancsokkal hozhatóak létre. Ezek a címkék az adott állományrendszer típusa alapján elnevezett alkönyvtárakban jönnek létre a /dev könyvtáron belül. Például az UFS2 állományrendszer-címkék a /dev/ufs könyvtárban keletkeznek. Állandó címkék a `glabel label` paranccsal hozhatóak létre. Az ilyen címkék nem függenek az állományrendszerek típusától, a /dev/label könyvtárban jönnek létre.

Az ideiglenes címkék a következõ induláskor elvesznek. Ezek a címkék a /dev/label könyvtárban keletkeznek, és ideálisak a kísérletezgetésre. Ideiglenes címkéket a `glabel create` paranccsal hozhatunk létre. Ezzel kapcsolatosan részletesebb felvilágosítást a glabel(8) man oldalon találhatunk.

Ha egy UFS2 állományrendszerre szeretnénk tenni egy állandó címkét az adataink megsemmisítése nélkül, adjuk ki a következõ parancsot:

```
# tunefs -L home /dev/da3
```

Ha az érintett állományrendszeren nincs üres hely, ennek a parancsnak a

használata adatvesztéshez vezethet. Ilyen esetben inkább a felesleges állományok eltávolításával kellene törõdnünk, nem pedig címkék hozzáadásával.

Ezután egy címkének kell megjelennie a /dev/ufs könyvtárban, amelyet vegyünk is fel az /etc/fstab állományba:

```
/dev/ufs/home       /home           ufs     rw              2       2
```

Az állományrendszert tilos csatolni a `tunefs` futtatása alatt!

Most már a megszokott módon csatolhatjuk az állományrendszert:

```
# mount /home
```

Ettõl a ponttól kezdve, amíg a geom_label.ko modul betöltõdik a rendszerindítás során a /boot/loader.conf állományon keresztül, vagy a `GEOM_LABEL` opció megtalálható a rendszermag konfigurációs állományában, az eszközleíró a rendszerre nézve minden komolyabb következmény nélkül megváltozhat.

Állományrendszereket létrehozhatunk alapértelmezett címkével is a `newfs -L` paraméterével. Errõl részletesebben a newfs(8) man oldalon olvashatunk.

Az alábbi paranccsal tudjuk törölni a címkét:

```
# glabel destroy home
```

A következõ példában azt láthatjuk, hogyan címkézzük fel a rendszerindító lemezünk partícióit.

Példa 28. Partíciók címkézése a rendszerindító lemezen

A rendszerindításra használt lemezen levõ partíciók felcímkézésével a rendszer képes lesz akkor is minden probléma nélkül elindulni, amikor áthelyezzük egy másik vezérlõre vagy átrakjuk egy másik számítógépbe. Például most tegyük fel, hogy van egy ATA csatolós lemezünk, amelyet a rendszer ad0 néven ismert fel. Továbbá azt is feltételezzük, hogy a FreeBSD telepítése esetén megszokott partícionálási sémát választottuk, ahol /, /var, /usr és /tmp állományrendszereink, valamint egy lapozóterületünk van.

Indítsuk újra a rendszerünket és a loader(8) menüjében a `4` billentyû lenyomásával válasszuk az egyfelhasználós módot. Ezt követõen adjuk ki a következõ parancsokat:

```
# glabel label rootfs /dev/ad0s1a
GEOM_LABEL: Label for provider /dev/ad0s1a is label/rootfs
# glabel label var /dev/ad0s1d
GEOM_LABEL: Label for provider /dev/ad0s1d is label/var
# glabel label usr /dev/ad0s1f
GEOM_LABEL: Label for provider /dev/ad0s1f is label/usr
```

```
# glabel label tmp /dev/ad0s1e
GEOM_LABEL: Label for provider /dev/ad0s1e is label/tmp
# glabel label swap /dev/ad0s1b
GEOM_LABEL: Label for provider /dev/ad0s1b is label/swap
# exit
```

A rendszer indítása ezután többfelhasználós módban folytatódik. A rendszerindítás befejezõdése után nyissuk meg az /etc/fstab állományt és írjuk át a hagyományos eszközneveket a hozzájuk tartozó címkékre. Az /etc/fstab végleges változata ennek megfelelõen körülbelül így fog kinézni:

```
# Eszköz                Csatlakozási pont  Típus   Beállítások     Dump    Menet
/dev/label/swap         none               swap    sw              0       0
/dev/label/rootfs       /                  ufs     rw              1       1
/dev/label/tmp          /tmp               ufs     rw              2       2
/dev/label/usr          /usr               ufs     rw              2       2
/dev/label/var          /var               ufs     rw              2       2
```

A rendszer most már újraindítható. Ha mindent jól csináltunk, akkor a rendszer indítása problémáktól mentesen fog zajlani és a `mount` parancs eredménye a következõ lesz:

```
# mount
/dev/label/rootfs on / (ufs, local)
devfs on /dev (devfs, local)
/dev/label/tmp on /tmp (ufs, local, soft-updates)
/dev/label/usr on /usr (ufs, local, soft-updates)
/dev/label/var on /var (ufs, local, soft-updates)
```

A FreeBSD 7.2 kiadásától kezdõdõen a glabel(8) osztály az UFS esetén támogatja az `ufsid`, az állományrendszer egyedi rendszerszintû azonosítójából származtatott új címketípus használatát. Ezek a címkék a rendszer indítása során a /dev/ufsid könyvtárban jönnek automatikusan létre. Az `ufsid` címkéken keresztül tudunk az /etc/fstab állományban állományrendszereket csatlakoztatni. A jelenleg aktív állományrendszereket és azok `ufsid` azonosítóit a `glabel status` paranccsal tudjuk lekérdezni:

```
% glabel status
                  Name  Status  Components
ufsid/486b6fc38d330916     N/A  ad4s1d
ufsid/486b6fc16926168e     N/A  ad4s1f
```

Ebben a példában az ad4s1d képviseli a /var állományrendszert, míg a ad4s1f a /usr állományrendszert. Az adott `ufsid` értékek megadásával az /etc/fstab állományban a következõképpen tudjuk csatlakoztatni ezeket az állományrendszereket:

```
/dev/ufsid/486b6fc38d330916       /var        ufs        rw        2      2
```

```
/dev/ufsid/486b6fc16926168e        /usr        ufs        rw        2      2
```

Minden `ufsid` címkével rendelkezõ partíció csatlakoztatható ezen a módon. Ekkor nem kell manuálisan létrehoznunk a számunkra állandó címkéket, így automatikusan élvethezhetjük az eszköznévtõl független csatlakoztatás elõnyeit.

19.7. Naplózó UFS GEOM-on keresztül

A FreeBSD 7.0-ás verziójának megjelenésével egy rég várt kiegészítés, a naplózás vált végre elérhetõvé vált. Maga az implementáció a GEOM alrendszeren keresztül érhetõ el, és a gjournal(8) segédprogram segítségével könnyedén beállítható.

Mit is jelent a naplózás? A naplózás támogatásával a rendszer egy naplót vezet az állományrendszert érintõ tranzakciókról - például az olyan változtatásokról, amelyek egy komplett írási mûveletet eredményeznek - mielõtt még a metaadatok és lemezírási mûveletek szabályosan befejezõdnének. Ez a könyvelés késõbb visszajátszható az állományrendszerben lezajlott tranzakciók reprodukálásához, és ezzel megelõzhetõek az állományrendszerben keletkezõ esetleges ellentmondások.

Ez egy újabb módszer az adatvesztés és az állományrendszerben elõforduló ellentmondások elkerülésére. Eltérõen a Soft Updates módszertõl, ahol a metaadatok frissítését biztosítják és követik nyomon, vagy a Snapshots módszertõl, ahol pillanatképeket tárolunk az állományrendszerrõl, itt egy konkrét naplót tárolunk a lemez erre a célra fenntartott részén, amely bizonyos esetekben akár egy teljes külön merevlemez is lehet.

Ellentétben a többi naplózó állományrendszertõl, a `gjournal` módszere blokk alapú és nem az állományrendszer részeként került implementálásra - csupán a GEOM egyik bõvítménye.

A `gjournal` támogatásához a FreeBSD rendszermag konfigurációs állományában be kell állítani a következõ opciót - amely a 7.0 és késõbbi rendszereken alapbeállítás:

```
options UFS_GJOURNAL
```

Amennyiben naplózással rendelkezõ köteteket szeretnénk a rendszerindítás során csatlakoztatni, a /boot/loader.conf állományban következõ sor hozzáadásával töltessük be a geom_journal.ko modult:

```
geom_journal_load="YES"
```

Szükség esetén ezt a funkciót akár a rendszermagba is beépíthetjük, ha felvesszük a következõ sort a rendszermag konfigurációs állományába:

```
options GEOM_JOURNAL
```

Ha ezt aktiváltuk, egy szabad állományrendszeren az alábbi lépéseken keresztül tudunk létrehozni

egy naplót, feltéve, hogy a da4 egy új SCSI-meghajtó:

```
# gjournal load
# gjournal label /dev/ad4
```

Ennél a pontnál lennie kell egy /dev/da4 és egy /dev/da4.journal eszközleírónak. Hozzunk létre egy állományrendszert ezen az eszközön:

```
# newfs -O 2 -J /dev/da4.journal
```

Ez a parancs létrehoz egy UFS2 állományrendszert a naplóval rendelkezõ eszközön.

Csatoljuk is be a `mount` segítségével az eszközt kívánt csatlakozási pontra:

```
# mount /dev/da4.journal /mnt
```

Ha több slice-unk is van, akkor a napló mindegyik slice-hoz külön létrejön. Például, ha az ad4s1 és ad4s2 egyaránt slice-ok, akkor a `gjournal` legyártja az ad4s1.journal és ad4s2.journal eszközleírókat.

A jobb teljesítmény elérése érdekében kívánatos lehet a naplót egy másik lemezen tartani. Ilyen esetekben a naplózás bekapcsolásához a naplót biztosító szolgáltatót vagy tárolóeszközt a naplózni kívánt eszköz után kell szerepeltetni. A naplózás akár az aktuálisan használt állományrendszeren is aktiválható a `tunefs` használatával. Az állományrendszer módosításakor viszont mindig érdemes biztonsági másolatot készíteni! Az esetek többségében a `gjournal` hibát fog jelezni, mivel nem tudja létrehozni a naplót, azonban ez nem védi meg az adatainkat a `tunefs` helytelen használata által okozott sérülésektõl.

A rendszerindító lemezen is lehet naplózást használni. Ennek részleit a Naplózó UFS használata asztali számítógépeken címû cikkbõl ismerhetjük meg.

Chapter 20. Támogatott állományrendszerek

20.1. Áttekintés

Az állományrendszerek szerves részét képezik napjaink operációs rendszereinek. Segítségükkel a felhasználók adatokat tölthetnek fel és tárolhatnak a számítógépen, szabályozhatják a hozzáférésüket, és természetesen mûködtethetik a merevlemezeiket. A különféle operációs rendszerekben általában azért annyi közös, hogy mindannyiukhoz tartozik egy natív, vagyis általuk alapból ismert állományrendszer. A FreeBSD esetében ezt konkrétan a Fast File System vagy röviden FFS, amely az eredeti Unix™ File System, vagy más néven UFS megoldásain alapszik. A FreeBSD tehát a merevlemezeken ebben a natív állományrendszerben tárol adatokat.

A FreeBSD természetesen ezen kívül még ismer számos egyéb állományrendszert, ezáltal képes adatokat olvasni más operációs rendszerek részérõl is kezelhetõ partíciókról, például helyi USB-eszközökrõl, flashkártyákról és merevlemezekrõl. Továbbá ismeri néhány más operációs rendszer natív állományrendszerét, mint például a Linux® Extended File System (EXT) vagy éppen a Sun™ Z File System (ZFS).

FreeBSD alatt az egyes állományrendszerek ismerete változó. Bizonyos esetekben elegendõ csupán egy megfelelõ modul betöltése, máskor viszont egy komplett eszközkészlet segítségével tudunk velük dolgozni. Ez a fejezet igyekszik a Sun™-féle Z állományrendszerrel kezdõdõen bemutatni a FreeBSD felhasználói számára más állományrendszerek használatát.

A fejezet elolvasása során megismerjük:

- a natív és támogatott állományrendszerek közti különbségeket;
- a FreeBSD által ismert állományrendszereket;
- hogyan engedélyezzünk, állítsunk be és érjünk el nem natív állományrendszereket.

A fejezet elolvasásához ajánlott:

- a UNIX® és FreeBSD alapjainak ismerete (A UNIX alapjai);
- a rendszermag konfigurációjának és fordításának alapvetõ fogásainak ismerete (A FreeBSD rendszermag testreszabása);
- a különbözõ külsõ fejlesztésû szoftverek telepítésének ismerete (Alkalmazások telepítése. csomagok és portok);
- a lemezek és egyéb tárolóeszközök, valamint a FreeBSD alatt az eszközök elnevezésének minimális ismerete (Háttértárak).

20.2. A Z állományrendszer (ZFS)

A Sun™ Z állományrendszere egy új, közös tárolási módszeren nyugvó technológia. Ez annyit jelent a gyakorlatban, hogy mindig csak annyi helyet foglal, amennyire az adatoknak közvetlenül szüksége van. Emellett úgy alakították ki, hogy az adatok épségét minél inkább védje, ezért például megtalálhatjuk benne a pillanatképek készítését, a másolatok létrehozását és az adatok sértetlenségének ellenõrzését. Továbbá egy RAID-Z néven bemutatott új replikációs modellt is

támogat. A RAID-Z alapvetõen a RAID-5 megoldásához hasonlít, azonban írás során keletkezõ hibák ellen igyekszik védelmet nyújtani.

20.2.1. A ZFS finomhangolása

A ZFS funkcióit megvalósító alrendszer alapértelmezés szerint meglehetõsen sok erõforrást kíván, ezért nem árt a legjobb hatékonyságra behangolnunk a mindennapokban felmerülõ igények mentén. Mivel ez még egy fejlesztés és tesztelés alatt álló része a FreeBSD-nek, elképzelhetõ, hogy ez a jövõben változik, viszont jelen pillanatban a következõ lépéseket javasoljuk.

Memória

Hasznos, ha a rendszerünkben legalább 1 GB memória található, de inkább 2 vagy több az ajánlott. Az itt szereplõ példákban ehelyett azonban mindenhol csupán 1 GB-ot feltételezünk.

Néhányaknak sikerült 1 GB-nál kevesebb központi memóriával is használni ezt az állományrendszert, azonban ilyenkor nagyon könnyen elõfordulhat, hogy komolyabb terhelés esetén a FreeBSD a memória elfogyása miatt egyszerûen összeomlik.

A rendszermag beállításai

A rendszermag konfigurációs állományából javasolt eltávolítani az összes nem használt meghajtót és funkciót. A legtöbb meghajtó egyébként is elérhetõ modul formájában, és a /boot/loader.conf állományon keresztül minden gond nélkül betölthetõek.

Az i386™ architektúránál szükségünk lesz az alábbi konfigurációs beállítás megadására, majd a rendszermag újrafordítására, végül a rendszer újraindítására:

```
options     KVA_PAGES=512
```

Ezzel az opcióval a rendszermag címterét növeljük meg, aminek eredményeképpen a `vm.kvm_size` változót immáron az eredetileg 1 GB-os (PAE használata esetén pedig 2 GB-os) határ felé tudjuk állítani. Az itt megadandó értéket úgy tudjuk meghatározni, ha a beállítani kívánt méret MB-okban számolt értékét elosztjuk néggyel. A példában tehát az `512` egy 2 GB nagyságú címteret ad meg.

A rendszertöltõ beállításai

A kmem címterét az összes FreeBSD által ismert architektúra esetében érdemes megnövelnünk. A teszteléshez használt rendszeren 1 GB fizikai memória állt rendelkezésre, itt a /boot/loader.conf állományban a következõ értékek megadásával minden remekül mûködött:

```
vm.kmem_size="330M"
vm.kmem_size_max="330M"
vfs.zfs.arc_max="40M"
vfs.zfs.vdev.cache.size="5M"
```

A ZFS finomhangolásával kapcsolatos további javasolatokat a http://wiki.freebsd.org/ZFSTuningGuide címen olvashatunk.

20.2.2. A ZFS használata

A Z állományrendszerhez létezik egy olyan mechanizmus, amelyen keresztül már a FreeBSD indítása során el tudjuk végezni a közös tárolók csatlakoztatását:

```
# echo 'zfs_enable="YES"' >> /etc/rc.conf
# /etc/rc.d/zfs start
```

A leírás fennmaradó részében feltételezzük, hogy három SCSI-lemezünk van, amelyeket rendre a da0, da1 és da2 eszközök formájában tudunk elérni. Az IDE lemezek tulajdonosainak értelemszerûen itt majd az ad eszközneveket kell használniuk a SCSI-eszközök hivatkozásai helyett.

Egyetlen közös tároló használata

A `zpool` kiadásával egyetlen lemezen is létre tudunk hozni egy egyszerû, nem redundáns ZFS partíciót:

```
# zpool create minta /dev/da0
```

Az új közös tárterület a `df` parancs felhasználásával rögtön láthatóvá válik:

```
# df
Filesystem  1K-blocks    Used    Avail Capacity  Mounted on
/dev/ad0s1a   2026030  235230  1628718    13%    /
devfs               1       1        0   100%    /dev
/dev/ad0s1d  54098308 1032846 48737598     2%    /usr
minta        17547136       0 17547136     0%    /minta
```

A parancs kimenetében tisztán láthatjuk, hogy a `minta` nevû tároló nem csak egyszerûen elkészült, hanem egyúttal *csatolódott*. Innentõl már a többi állományrendszerhez hasonlóan tetszõlegesen elérhetõ, az alábbi példához hasonlóan állományok hozhatóak rajta létre vagy listázható a tartalma:

```
# cd /minta
# ls
# touch proba
# ls -al
total 4
drwxr-xr-x   2 root  wheel    3 Aug 29 23:15 .
drwxr-xr-x  21 root  wheel  512 Aug 29 23:12 ..
-rw-r--r--   1 root  wheel    0 Aug 29 23:15 proba
```

Sajnos azonban ez a tároló még ki sem használja a ZFS által felkínált lehetõségeket. Ezért most hozzunk létre egy állományrendszert ezen a tárolón belül és engedélyezzük rajta a tömörítést:

```
# zfs create minta/tomoritett
# zfs set compression=gzip minta/tomoritett
```

A `minta/tomoritett` most már egy tömörített Z állományrendszer. Próbáljuk ki mit tud, és másoljunk néhány nagyobb méretû állományt a /minta/tomoritett könyvtárba.

Ezután a tömörítés akár ki is kapcsolható:

```
# zfs set compression=off minta/tomoritett
```

Az állományrendszer leválasztásához adjuk ki a lenti parancsot, majd ellenõrizzük az eredményét a `df` használatával:

```
# zfs umount minta/tomoritett
# df
Filesystem  1K-blocks    Used    Avail Capacity  Mounted on
/dev/ad0s1a   2026030  235232  1628716    13%    /
devfs               1       1        0   100%    /dev
/dev/ad0s1d  54098308 1032864 48737580     2%    /usr
minta        17547008       0 17547008     0%    /minta
```

Tegyük ismét elérhetõvé és csatlakoztassuk újra az állományrendszert, majd nézzük meg az eredményt a `df` paranccsal:

```
# zfs mount minta/tomoritett
# df
Filesystem         1K-blocks    Used    Avail Capacity  Mounted on
/dev/ad0s1a          2026030  235234  1628714    13%    /
devfs                      1       1        0   100%    /dev
/dev/ad0s1d         54098308 1032864 48737580     2%    /usr
minta               17547008       0 17547008     0%    /minta
minta/tomoritett    17547008       0 17547008     0%    /minta/tomoritett
```

A közös terület és az állományrendszer mellesleg a `mount` parancs kimenetébõl is megfigyelhetõ:

```
# mount
/dev/ad0s1a on / (ufs, local)
devfs on /dev (devfs, local)
/dev/ad0s1d on /usr (ufs, local, soft-updates)
minta on /minta (zfs, local)
minta/tomoritett on /minta/tomoritett (zfs, local)
```

Látható, hogy a létrehozásuk után a Z állományrendszerek teljesen hétköznapi módon viselkednek, de természetesen további lehetõségek is elérhetõek hozzájuk. A következõ példában `adat` néven készítünk egy új állományrendszert. Mivel ide majd nagyon fontos állományokat akarunk

elhelyezni, állítsuk be, hogy minden adatblokkból két példány legyen:

```
# zfs create minta/adat
# zfs set copies=2 minta/adat
```

A df újbóli kiadásával most már látható is ez az állományrendszer és annak tárfoglalása:

```
# df
Filesystem         1K-blocks    Used    Avail Capacity  Mounted on
/dev/ad0s1a          2026030  235234  1628714    13%    /
devfs                      1       1        0   100%    /dev
/dev/ad0s1d         54098308 1032864 48737580     2%    /usr
minta               17547008       0 17547008     0%    /minta
minta/tomoritett    17547008       0 17547008     0%    /minta/tomoritett
minta/adat          17547008       0 17547008     0%    /minta/adat
```

Vegyük észre, hogy a közös területen levõ állományrendszerek mindegyikén ugyanannyi szabad terület van. A df segítségével a késõbbiekben remekül megfigyelhetõ lesz, hogy az egyes állományrendszerek mindig csak annyi területet foglalnak el a közös területbõl, amennyire abban a pillanatban ténylegesen szükségünk van. A Z állományrendszerek esetén megszûnik a partíciók és kötetek fogalma, és több állományrendszer tárolódik egyazon közös területen. Ha már nem akarjuk használni, egyszerûen csak töröljük le az állományrendszereket és ezt a közös tárolót:

```
# zfs destroy minta/tomoritett
# zfs destroy minta/adat
# zpool destroy minta
```

Nyilván tapasztalhattunk már, hogy a lemezeink olykor menthetetlenül meghibásodnak. Amikor egy lemezes meghajtó tönkremegy, a rajta tárolt adatok általában elvesznek. Az ilyen jellegû kellemetlenségek elkerülésének egyik módja az ún. RAID-tömbök építése. A következõ szakaszban bemutatjuk, hogy a Z állományrendszerek esetén hogyan tudunk ilyen tömböket készíteni.

RAID-Z tömbök

Korábban már utaltunk rá, hogy ebben a szakaszban három SCSI-lemez, vagyis a da0, da1 és da2 eszközök használatát feltételezzük (vagy természetesen ad0 és így tovább, ha IDE-lemezeket használunk). Egy RAID-Z formátumú közös tároló készítéséhez a következõ parancsot kell kiadni:

```
# zpool create tarolo raidz da0 da1 da2
```

A Sun™ ajánlása szerint egy RAID-Z konfigurációban legalább három, legfeljebb kilenc lemezt javasolt alkalmazni. Ha egyetlen közös tárolóban esetleg tíznél több lemezt szeretnénk felhasználni, akkor érdemes inkább kisebb RAID-Z csoportokra felosztani ezeket. Ha viszont csak két lemezünk van, de továbbra is redundanciára lenne szükségünk, hozzunk helyette létre egy ZFS tükrözést. Ezzel kapcsolatban

> részletesebben a zpool(8) man oldalon keresztül tájékozódhatunk.

Ennek hatására tehát keletkezik egy `tarolo` nevû Z-tároló. Ez a korábbiakhoz hasonló módon ellenõrizhetõ is a mount(8) és df(1) parancsokon keresztül. Természetesen az iménti listába további lemezeszközök tetszõlegesen felvehetõek. Most hozzunk létre ezen a közös területen egy `felhasznalok` nevû állományrendszert, ahová majd a felhasználók adatait fogjuk tenni:

```
# zfs create tarolo/felhasznalok
```

Miután ezzel megvagyunk, az imént létrehozott állományrendszerre nyugodtan beállíthatunk tömörítést és biztonsági másolatokat. Ebben az alábbi parancsok lesznek a segítségünkre:

```
# zfs set copies=2 tarolo/felhasznalok
# zfs set compression=gzip tarolo/felhasznalok
```

Ezt követõen költöztessük át a felhasználókat, vagyis másoljuk át az adataikat ide és hozzuk létre a megfelelõ szimbolikus linkeket:

```
# cp -rp /home/* /tarolo/felhasznalok
# rm -rf /home /usr/home
# ln -s /tarolo/felhasznalok /home
# ln -s /tarolo/felhasznalok /usr/home
```

A felhasználók adatai immáron a frissen létrehozott /tarolo/felhasznalok állományrendszeren tárolódnak. Próbáljuk ki, hozzunk létre egy új felhasználót és jelentkezzünk be vele.

Készítsünk most egy pillanatképet is, amelyet aztán késõbb szükség esetén vissza tudunk állítani:

```
# zfs snapshot tarolo/felhasznalok@08-08-30
```

A `snapshot` csak valós állományrendszerekkel mûködik, könyvtárakra vagy állományokra nem. A nevében a `@` karakter választja el egymástól a hozzá tartozó címkét az állományrendszer vagy kötet nevétõl. Ha netalán a felhasználói könyvtárak valamiért megsérültek volna, a következõ paranccsal állíthatóak vissza:

```
# zfs rollback tarolo/felhasznalok@08-08-30
```

Az adott idõpontban aktív pillanatképeket az adott állományrendszer .zfs/snapshot könyvtárában találhatjuk meg. Például az elõbb készített pillanatkép az alábbi paranccsal nézhetõ meg:

```
# ls /tarolo/felhasznalok/.zfs/snapshot
```

Ha ebbõl elindulunk, akkor pillanatok alatt írható egy olyan szkript, amely a felhasználók adatairól

havonta készít egy pillanatképet. Ilyenkor azonban fontos számításba vennünk, hogy az idõvel felgyülemlõ pillanatképek rengeteg helyet el tudnak foglalni. A korábbi pillanatkép így távolítható el:

```
# zfs destroy tarolo/felhasznalok@08-08-30
```

Miután alaposan kipróbáltuk a /tarolo/felhasznalok néven létrehozott állományrendszerünket, állítsuk be véglegesen ez eddigi /home állományrendszer helyére:

```
# zfs set mountpoint=/home tarolo/felhasznalok
```

Ekkor a `df` és `mount` parancsok használatával meggyõzõdhetünk róla, hogy ezt az állományrendszert innentõl már valóban a /home könyvtárnak tekintjük:

```
# mount
/dev/ad0s1a on / (ufs, local)
devfs on /dev (devfs, local)
/dev/ad0s1d on /usr (ufs, local, soft-updates)
tarolo on /tarolo (zfs, local)
tarolo/felhasznalok on /home (zfs, local)
# df
Filesystem          1K-blocks    Used    Avail Capacity  Mounted on
/dev/ad0s1a           2026030  235240  1628708    13%    /
devfs                       1       1        0   100%    /dev
/dev/ad0s1d          54098308 1032826 48737618     2%    /usr
tarolo               26320512       0 26320512     0%    /tarolo
tarolo/felhasznalok  26320512       0 26320512     0%    /home
```

Ezzel lényegében befejeztük a RAID-Z tömb konfigurációját. Az állományrendszerek állapotára vonatkozóan a periodic(8) alkalmazásával akár naponta kérhetünk ellenõrzést:

```
# echo 'daily_status_zfs_enable="YES"' >> /etc/periodic.conf
```

A RAID-Z helyreállítása

Minden szoftveres RAID implementáció kínál valamilyen megoldást az állapotának ellenõrzésére, ez alól tulajdonképpen a ZFS sem kivétel. A RAID-Z eszközök állapota a következõ paranccsal kérdezhetõ le:

```
# zpool status -x
```

Ezt az üzenetet láthatjuk, amikor minden tároló kifogástalanul mûködik és semmilyen probléma sincs:

```
all pools are healthy
```

Ha viszont valamilyen gond lenne valamelyik lemezzel, például leállt, akkor az elõbbi parancs eredménye ehhez lesz hasonló:

```
  pool: tarolo
 state: DEGRADED
status: One or more devices has been taken offline by the administrator.
    Sufficient replicas exist for the pool to continue functioning in a
    degraded state.
action: Online the device using 'zpool online' or replace the device with
    'zpool replace'.
 scrub: none requested
config:

    NAME        STATE     READ WRITE CKSUM
    tarolo      DEGRADED     0     0     0
      raidz1    DEGRADED     0     0     0
        da0     ONLINE       0     0     0
        da1     OFFLINE      0     0     0
        da2     ONLINE       0     0     0

errors: No known data errors
```

A válasz szerint az eszközt az adminisztrátor állította le. Ez ennél a példánál valóban igaz. Lemezeket a következõ módon lehet leállítani:

```
# zpool offline tarolo da1
```

Így miután leállítottuk a rendszert, a da1 eszköz cserélhetõ. A rendszer soron következõ indításakor ezzel a paranccsal tudjuk jelezni logikailag is a lemez cseréjét:

```
# zpool replace tarolo da1
```

Nézzük meg újra a tömb állapotát, de ezúttal a `-x` kapcsoló megadása nélkül, mivel csak így fogjuk látni:

```
# zpool status tarolo
 pool: tarolo
 state: ONLINE
 scrub: resilver completed with 0 errors on Sat Aug 30 19:44:11 2008
config:

    NAME        STATE     READ WRITE CKSUM
    tarolo      ONLINE       0     0     0
```

```
      raidz1    ONLINE       0     0     0
        da0     ONLINE       0     0     0
        da1     ONLINE       0     0     0
        da2     ONLINE       0     0     0

errors: No known data errors
```

A példa szerint minden megfelelõen mûködik.

Az adatok ellenõrzése

Elõzetesen már szó esett róla, hogy a ZFS képes a tárolt adatok sértetlenségének ellenõrzésére. Az új állományrendszerek létrehozásánál ez a lehetõség automatikusan aktiválódik, de tetszés szerint letiltható:

```
# zfs set checksum=off tarolo/felhasznalok
```

Ez a lépés viszont nem feltétlenül jó döntés, mivel az adatintegritás megtartásához felhasznált ellenõrzõ összegek nagyon kevés helyet foglalnak és meglehetõsen hasznosak. Emellett semmilyen észlelhetõ lassulást nem okoznak az állományrendszer használata során. Ha engedélyezzük, a ZFS ilyen ellenõrzõ összegek segítségével folyamatosan figyelni tudja az adatok épségét. Ezt az ellenõrzést a `scrub` paranccsal válthatjuk ki. Nézzük meg például a `tarolo` esetében:

```
# zpool scrub tarolo
```

Ez a vizsgálat a tárolt adatok mennyiségétõl függõen nagyon sokáig is eltarthat, illetve rengeteg lemezmûveletet foglal magában, ezért egyszerre csak egy ilyen futtatása javasolt. Miután befejezõdött, a tároló állapota az eredményének megfelelõen frissül, amelyet közvetlenül utána le is kérdezhetünk:

```
# zpool status tarolo
 pool: tarolo
 state: ONLINE
 scrub: scrub completed with 0 errors on Sat Aug 30 19:57:37 2008
config:

    NAME        STATE     READ WRITE CKSUM
    tarolo      ONLINE       0     0     0
      raidz1    ONLINE       0     0     0
        da0     ONLINE       0     0     0
        da1     ONLINE       0     0     0
        da2     ONLINE       0     0     0

errors: No known data errors
```

A példában látható az utolsó ellenõrzés ideje. Ezen lehetõség használatával hosszú idõn keresztül

szavatolni tudjuk az adataink épségét.

A Z állományrendszerrel kapcsolatos további beállítási lehetõségekrõl a zfs(8) és zpool(8) man oldalakon olvashatunk.

Chapter 21. A Vinum kötetkezelő

21.1. Áttekintés

Nem számít, milyen lemezeink is vannak, ugyanis mindig adódnak velük kapcsolatban gondjaink:

- Kicsik.
- Lassúk.
- Nem elég megbízhatóak.

Ezekre a problémákra javasoltak és meg is valósítottak számos megoldást. A felhasználók egy része általában úgy védekezik ellenük, hogy több, gyakran redundánsan tároló lemezt használ. A különféle kártyák és hardveres RAID-vezérlõk támogatása mellett a FreeBSD alaprendszerében megtalálható egy blokkos eszközmeghajtóként a Vinum kötetkezelõ is, amellyel virtuális lemezmeghajtókat lehet létrehozni. Tehát a *Vinum* egy olyan ún. *kötetkezelõ*, vagyis virtuális lemezkezelõ, ami az említett három problémára próbál megoldást adni. A Vinum a hagyományos lemezes tárolásnál jóval nagyobb rugalmasságot, teljesítményt és megbízhatóságot biztosít, valamint ismeri a RAID-0, RAID-1 és RAID-5 modelleket külön-külön és kombinálva is.

Ebben a fejezetben összefoglaljuk a hagyományos lemezes tárolás jellegzetes problémáit és bemutatjuk a Vinum kötetkezelõt.

A FreeBSD 5-ös verziójától kezdve a Vinumot újraírták a GEOM-nak megfelelõen (GEOM. a moduláris lemezszervező rendszer), megtartva az eredeti elgondolásokat, elnevezéseket és a lemezen tárolt metaadatok formátumát. Ezt az újraírt változatot nevezik *gvinum*nak (*GEOM vinum*). A szövegben a *Vinum*ra kizárólag csak általánosságban hivatkozunk, függetlenül az implementációjától. Most már az összes parancsot a `gvinum` használatával kell kiadni, illetve a hozzá tartozó modul neve vinum.ko-ról geom_vinum.ko-ra változott és a megfelelõ eszközleírók a /dev/vinum könyvtár helyett a /dev/gvinum könyvtárban találhatóak. A FreeBSD 6. verziójától pedig a régi Vinum implementáció többé már nem is része az alaprendszernek.

21.2. Kicsik a lemezeink

A lemezek kapacitása ugyan növekszik, de velük együtt a tárigények is. Ezért gyakran érezzük úgy, hogy a rendelkezésünkre álló lemezek tárkapacitását meghaladó állományrendszerre lenne szükségünk. Kétségtelen, hogy ez a probléma messze nem akkora jelentõségû, mint például tíz évvel ezelõtt, de még mindig fennáll. Egyes rendszerek ezt úgy hidalták át, hogy létrehoztak egy olyan absztrakt eszközt, amely az adatokat több lemezen tárolja el.

21.3. A hozzáférési idõk szûk keresztmetszetei

Napjaink rendszerei szinte állandóan egyszerre több adathoz is hozzá akarnak férni. Például egy nagy forgalmú FTP vagy HTTP szerver több 100 Mbit/s sebességû kapcsolattal is csatlakozhat a világhálóhoz, amelyeken keresztül párhuzamosan többezernyi tranzakciót is folytathat, ami

jelentõsen meghaladja a legtöbb lemez átlagos átviteli sebességét.

A jelenleg kapható lemezek soros adatátviteli sebessége egészen 70 MB/s-ig is terjedhet, de ennek az értéknek kevés a jelentõsége olyan környezetekben, ahol több, egymástól függetlenül futó program próbál egyszerre hozzáférni, hiszen ilyen esetekben csak a töredékét képesek elérni. Ilyenkor sokkal érdekesebb a lemezt kezelõ alrendszer szempontjából nézni a problémát: így az egyes adatátviteli kérések terhelése lesz a meghatározó paraméter, vagyis az az idõ, amit a kérés teljesítésében érintett meghajtók eltöltenek a feldolgozással.

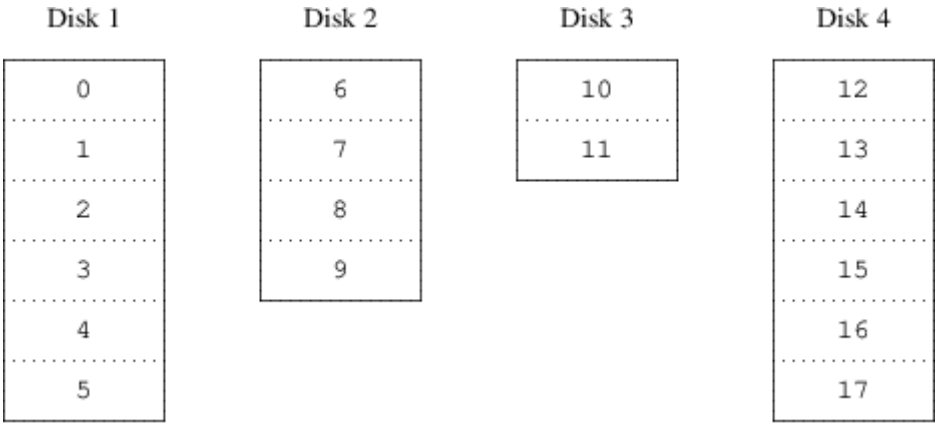
Bármelyik kérést is vesszük, a kiszolgáláshoz a meghajtónak elõször a megfelelõ helyre kell mozgatnia az író/olvasó fejeket, meg kell várni a fej alatt elhaladó elsõ szektort, majd végrehajtani a megfelelõ mûveletet. Ezek a mûveletek szétválaszthatatlanok: semmi értelme nincs megszakítani ezeket.

Tekintsünk egy átlagosnak mondható, nagyjából 10 kB méretû adatátvitelt: a legújabb nagyteljesítményû lemezek átlagosan 3,5 ms alatt képesek pozicionálni a fejeket. A leggyorsabb lemezek 15 000 fordulatot tesznek meg percenként (RPM), így az átlagos forgási késleltetés (egy fél fordulat ideje) 2 ms. 70 MB/s-os sebesség mellett az átvitel maga megközelítõleg 150 µs, ami szinte elhanyagolható a pozicionálás idejéhez képest. Ilyen esetekben a tényleges adatátviteli sebesség 1 MB/s-nél alig valamivel többre esik vissza, és tisztán látszik, hogy erõsen függ az átvitt adat mennyiségétõl.

A hagyományos és kézenfekvõ megoldása ennek a problémának "még több cséve" használata: egyetlen nagy lemez helyett alkalmazzunk több kisebb, de azonos tárkapacitású lemezt. Mindegyik lemez képes egymástól függetlenül mozgatni a fejeiket és az adatokat, aminek köszönhetõen a tényleges adatátvitel mértéke nagyjából a lemezek számával arányosan növekszik.

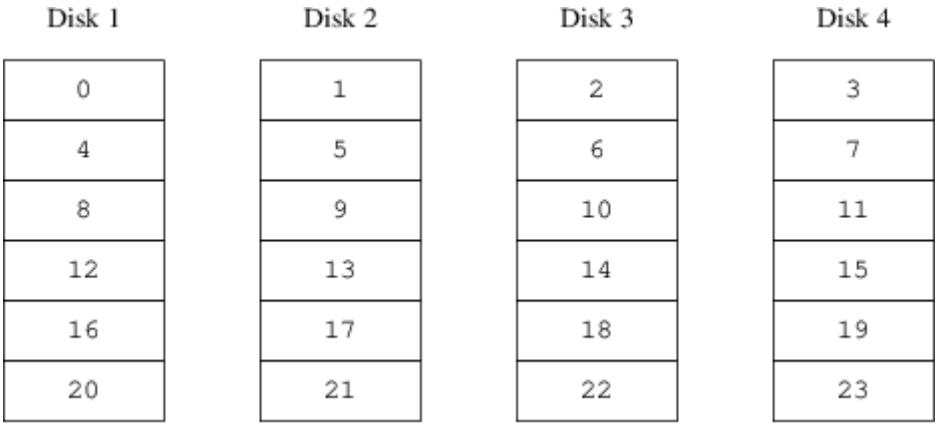
Az adatátvitelben bekövetkezõ javulás pontos aránya természetesen kisebb, mint a lemezek száma: habár az egyes meghajtók képesek párhuzamosan mozgatni az adatokat, semmilyen módon garantálhatjuk, hogy a kérések egyenletesen oszlanak el köztük. Emiatt szinte elkerülhetetlen, hogy az egyik meghajtót nagyobb terhelés érje, mint a másikat.

A lemezekre esõ terhelés egyenletessége erõsen függ attól, hogyan osztjuk el az adatokat a meghajtók között. Az itt használt példában a lemezen tárolt adatokat egy könyv oldalaiként érdemes elképzelni, vagyis rengeteg szám szerint címezhetõ adatszektorként. A virtuális lemezt ennek megfelelõen a legegyszerûbben úgy tudjuk felosztani az egymás után következõ független fizikai lemezek mérete szerint és így használni, mintha egy nagy könyvet kisebb részekre téptünk volna. Ezt a módszert nevezik *összefûzésnek*, és elõnye, hogy a résztvevõ lemezeknek nem kell azonos méretûeknek lenniük. Ez a megoldás remekül mûködik abban az esetben, amikor a virtuális lemez hozzáférései egyenletesen oszlanak el annak teljes területén. Amikor viszont az elérés csak egy kisebb területre korlátozódik, kevesebb javulás tapasztalható. A Az összefûzött szervezési mód mutatja be lemezek egy ilyen összefûzött konfigurációját.



Ábra 59. Az összefûzött szervezési mód

Feloszthatjuk a virtuális lemezünket kisebb azonos méretû darabokra is, melyeket különbözõ eszközökön sorosan tárolunk el. Például az elsõ 256 szektort eltároljuk az elsõ lemezen, majd a következõ 256 szektort a következõ lemezen és így tovább. Az utolsó lemez kitöltése után az egész folyamat ismétlõdik, egészen az összes lemez megtöltéséig. Ezt a leképezést *csíkozás*nak ("striping") vagy RAID-0-nak nevezzük . A csíkozás használata során valamivel bonyolultabbá válik az adatok megtalálása és többletmunkát is jelenthet olyan esetekben, amikor az adatátvitel több lemezt is érint, de ezzel egyidõben sokkal jobban szétosztja a terhelést a lemezek között. A A csíkozott szervezési mód mutatja be a lemezek csíkozott szervezését.



Ábra 60. A csíkozott szervezési mód

21.4. Adatintegritás

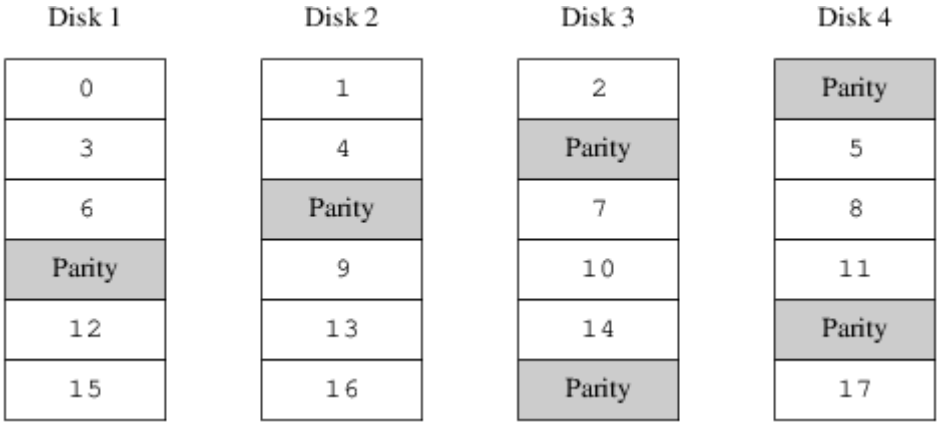
A modern lemezhajtók utolsó fontos problémája, hogy nem eléggé megbízhatóak. Annak ellenére, hogy a lemezek ezen a téren meglehetõsen sokat fejlõdtek az utóbbi pár évben, egy szervernek még mindig ezek azok a központi részei, amelyek a leginkább hajlamosak a meghibásodásra. Amikor ez bekövetkezik, a hatása akár egy katasztrófával is felérhet: a sérült lemezmeghajtók cseréje és az adatok visszaállítása napokat is igénybe vehet.

Ennek a problémának a hagyományos megközelítése lenne a *tükrözés* ("mirroring"), vagyis amikor ugyanarról az adatról tartunk két példányt két eltérõ fizikai hardveren. A RAID-szintek beköszöntével ezt a technikát RAID level 1-nak vagy RAID-1-nek is nevezik. Amikor írunk a kötetre, mindenhova írunk, az olvasás pedig bármelyik eszközrõl elvégezhetõ. Így ha az egyik meghajtó tönkremenne, egy másikon még mindig megtalálható az összes adat.

A tükrözés két problémát vet fel:

- Ár. Legalább kétszer annyiba kerül, mint a nem redundánsan tároló megoldások.
- Teljesítménycsökkenés. Mivel az írást minden meghajtón végre kell hajtani, legalább kétszer annyi sávszélességet is felémeszt, mint a nem tükrözött kötetek esetén. Az olvasás viszont nem veszít a sebességébõl: sõt, még gyorsabbnak is tûnhet.

Az adatintegritás megõrzésére egy másik megoldás a *paritás* használata, melyet a 2, 3, 4 és 5 RAID-szintek valósítanak meg. Ezek közül talán a RAID-5 a legérdekesebb. A Vinumban egy olyan csíkozott szervezési módként valósították meg, ahol minden csíkból egy blokk az összes többi paritási információját tartalmazza. A RAID-5 által megvalósított szervezés hasonlít a csíkozáshoz, azonban a RAID-5-ben mindegyik csík tartalmaz egy paritási információt is. Tehát a Vinumban, ahogy azt RAID-5 a megköveteli, a paritást tároló blokkok helye az egyik csíkról a másikra változik. Az adatblokkokban található számok relatív blokkszámokat jelölnek.



Ábra 61. A RAID-5 szervezési mód

A RAID-5-nek a tükrözéshez képest megvan az az elõnye, hogy jelentõsen kevesebb tárhelyet igényel. Az olvasás hasonló a csíkozott szervezésekéhez, azonban az írás jóval lassabb, közel 25%-a az olvasás sebességének. Az egyik meghajtó meghibásodása esetén a tömb csökkentett módban még képes folytatni a mûködést: a fennmaradó meghajtókról továbbra is a megszokott módon lehet olvasni, viszont a sérült meghajtóról olvasott adatokat folyamatosan javítani kell a többirõl származó segédinformációk szerint.

21.5. A Vinum objektumai

A tárgyalt problémák orvoslására a Vinumban egy négyszintû objektumhierarchiát alakítottak ki:

- A legjobban észlelhetõ objektum a virtuális lemez, amelyet *kötet*nek (volume) nevezünk. Ez a kötet lényegében ugyanazokkal a tulajdonságokkal rendelkezik, mint egy UNIX®-os lemezmeghajtó, habár akadnak finomabb különbségek. Mérete korlátlan lehet.
- A kötetek *erek*bõl (plex) állnak, melyek a kötet teljes területét képviselik. Ennélfogva a hierarchia ezen szintje nyújtja a redundanciát. Az ereket legegyszerûbben a tükrözött tömbben helyet foglaló lemezekként tudjuk elképzelni, melyek ugyanazt az adatot tartalmazzák.
- Mivel a Vinum a UNIX® lemezes tárolást megvalósító alrendszerében helyezkedik el, a többlemezes erek felépítéséhez használhatnánk a UNIX®-os partíciókat, azonban ehhez a feladathoz nem eléggé rugalmasak, mivel a UNIX®-os lemezek csak korlátozott számú partíciót tartalmazhatnak. A Vinum ehelyett *allemez*nek (subdisk) nevezett folytonos területekre osztja fel az egyes UNIX®-os partíciókat (a *meghajtó*kat), melyeket aztán az erek létrehozására használ

fel.

- A Vinum által létrehozott *meghajtó*kon (drive) levõ allemezek lesznek valódi UNIX®-os partíciók. A Vinum-meghajtók tetszõleges számú allemezt tartalmazhatnak. Eltekintve a meghajtó elején található apró területtõl, melyen a beállításokra és az állapotra vonatkozó információk tárolódnak, az egész meghajtó felhasználható adatok tárolására.

A most következõ szakaszokban ismertetjük, hogy ezek az objektumok milyen módon szolgáltatják a Vinum részérõl elvárt funkciókat.

21.5.1. A kötetek mérete

Az erek képesek a Vinum konfigurációjában található több különbözõ meghajtón elhelyezkedõ allemezeket is nyalábba kötni. Ennek következményeképpen az egyes meghajtók mérete nem korlátozza az erek méretét, emiatt a kötetét sem.

21.5.2. Redundáns adattárolás

A Vinum a tükrözést több ér egyetlen kötetté olvasztásával hozza létre. Az erek mindegyike a köteten található adatokat képviseli. Egy kötet legalább egy, legfeljebb nyolc eret tartalmazhat.

Habár egy ér egy kötet teljes adatát ábrázolja, elõfordulhat olyan eset, hogy bizonyos részei hiányoznak fizikai, kialakítási (nem társítottunk allemezeket hozzájuk) okokból adódóan vagy véletlenül (a hozzá tartozó lemezterületek sérültek). Amíg legalább egy ér képes a kötet teljes tartalmát szolgáltatni, addig a kötet teljesen épnek tekinthetõ.

21.5.3. Teljesítmény

A Vinum az összefûzést és a csíkozást is egyaránt megvalósítja az erek szintjén:

- Az *összefûzött ér* allemezek területeibõl építkezik.
- A *csíkozott ér* felosztja az adatokat az allemezek között. Az allemezek mindegyikének ugyanakkorának kell lennie, és legalább két allemeznek lennie kell, hogy eltérjen az összefûzött értõl.

21.5.4. Hogyan szervezzük az ereket?

A FreeBSD 12.0 verziójában két fajta erezési megoldást találhatunk:

- Az összefûzött erek a legrugalmasabbak: tetszõleges számú allemezt tartalmazhatnak, az allemezek mérete pedig eltérhet. Az ér újabb allemezek hozzáadásával tovább bõvíthetõ. Kevesebb processzoridõt igényel, mint egy csíkozott ér, habár a kettõ többletköltsége közti eltérés nem mérhetõ. Másrészrõl azonban nagyon érzékenyek a forgalmasabb pontokra, vagyis amikor az egyik lemez folyamatosan használatban van, miközben a többi üresen jár.
- A csíkozott (RAID-0) erek legnagyobb elõnye, hogy csökkentik a forgalmasabb pontok kialakulását: a megfelelõ méretû csíkszélesség (ami kb. 256 kB) választásával el tudjuk egyengetni a tömbben dolgozó meghajtók terhelését. Ennek a megközelítésnek a hátránya (részben) a sokkal összetettebb kód, valamint az allemezekre vonatkozó megszorítás, amely szerint meg kell egyezniük a méretüknek, illetve az érhez annyira bonyolult újabb allemezeket

kapcsolni, hogy a Vinum jelenleg nem is képes rá. Ezeken kívü a Vinum még támaszt egy triviális igényt is: a csíkozott érben legalább két allemeznek lennie kell, mivel másképp nem tér el egy összefûzött értõl.

A Vinum erezések foglalja össze az egyes erezések elõnyeit és hátrányait.

Táblázat 7. Vinum erezések

Erezés típusa	Legkevesebb allemez	Bõvíthetõ	Megegyezõ méret	Alkalmazás
összefûzött	1	igen	nem	Sok adat tárolása, ahol a hangsúly a rugalmasságon és a mérsékelt teljesítményen van.
csíkozott	2	nem	igen	Nagy teljesítmény, nagy mennyiségû egyidejû hozzáférés mellett

21.6. Példák

A Vinum a rendszerben ismert objektumokkal kapcsolatos információkat egy *konfigurációs adatbázis*ban tartja fenn. Kezdetben a felhasználó egy vagy több konfigurációs állomány segítségével hozza létre ezt az adatbázist a gvinum(8) segédprogrammal. A Vinum ezt a konfigurációs adatbázist bemásolja mindegyik irányítása alatt álló slice-ba (melyek a Vinum *eszköz*nek hív). Az adatbázis minden egyes állapotváltás folyamán frissül, így egy újraindítás után minden egyes Vinum-objektum állapota pontosan helyreállítódik.

21.6.1. A konfigurációs állomány

A konfigurációs állomány írja le az egyes objektumokat. Egy egyszerûbb kötet definíciója így nézhet ki:

```
drive a device /dev/da3h
volume myvol
  plex org concat
    sd length 512m drive a
```

Ez az állomány négy Vinum-objektumot definiál:

- A *drive* kezdetû sor adja meg a lemez partícióját (*meghajtó*ját) és a hardveren levõ elhelyezkedését. Az *a* szimbolikus nevet kapta. A szimbolikus és a konkrét eszköznevek szétválasztásával lehetõvé válik, hogy a lemezek félreértések nélkül átkerülhessek egyik helyrõl a másikra.
- A *volume* kezdetû sor adja meg a kötetet. Itt az egyetlen szükséges jellemzõ a név, ami ebben az

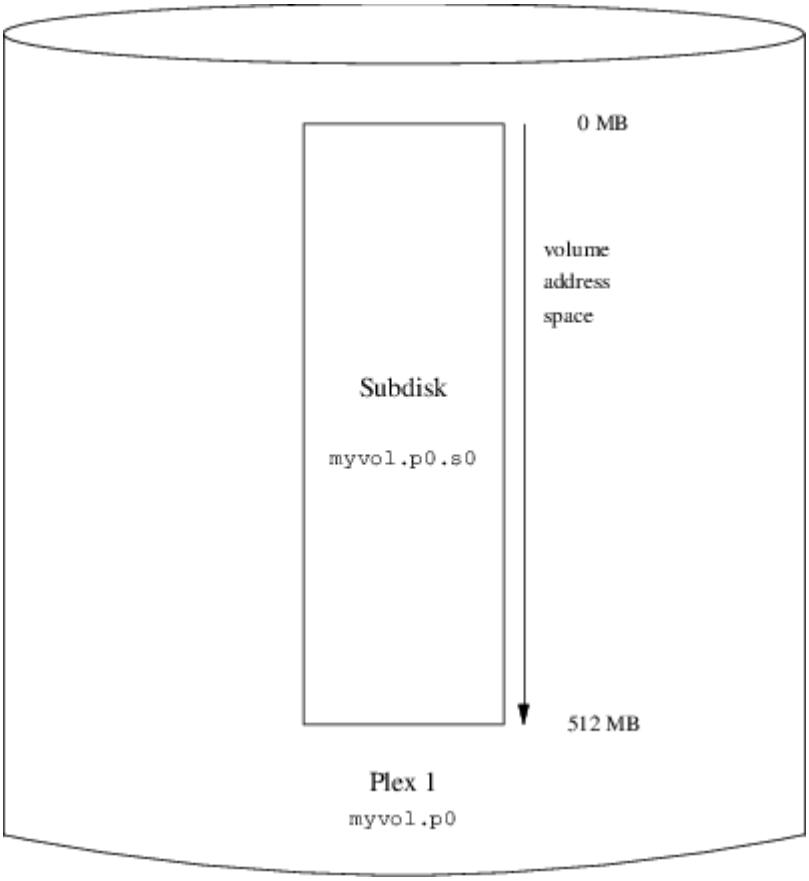
esetben a *myvol*.

- A *plex* kezdetû sor adja meg az eret. Itt az egyetlen szükséges paraméter a szervezési mód, ami ebben az esetben a *concat* (összefûzött). Nevet nem kell megadnunk, mivel a rendszer automatikusan létrehoz egy nevet a kötet nevébõl a *.px* utótag hozzáadásával, ahol az *x* az ér száma lesz a köteten belül. Emiatt a most definiált ér neve *myvol.p0* lesz.
- Az *sd* kezdetû sor adja meg az allemezt. Itt legalább meg kell adnunk a meghajtónak a nevét, ahol tárolni akarjuk, ill. a méretét. Ahogy már említettük az ereknél is, nevet nem kötelezõ megadnunk, mivel a rendszer magától rendel hozzá nevet, amit a hozzá tartozó ér nevébõl származtat, hozzáadja a *.sx* utótagot, ahol az *x* az allemez éren belüli sorszáma lesz. Ennek következtében a Vinum ennek az allemeznek a *myvol.p0.s0* nevet adja.

Miután a gvinum(8) feldolgozta ezt az állományt, az alábbi kimenetet fogja adni:

```
      #  gvinum -> create config1
      Configuration summary
      Drives:         1 (4 configured)
      Volumes:        1 (4 configured)
      Plexes:         1 (8 configured)
      Subdisks:       1 (16 configured)

    D a                   State: up       Device /dev/da3h        Avail: 2061/2573
MB (80%)

    V myvol               State: up       Plexes:       1 Size:        512 MB

    P myvol.p0          C State: up       Subdisks:     1 Size:        512 MB

    S myvol.p0.s0         State: up       PO:        0  B Size:        512 MB
```

Ez a kimenet a gvinum(8) egyszerû listázási formátumát mutatja. Grafikusan a Egyszerû Vinum-kötet mutatja be.



Ábra 62. Egyszerû Vinum-kötet

Ezen és az ezt követõ ábrán egy kötetet láthatunk, amely ereket tartalmaz, amelyek pedig allemezeket. Ebben az alapvetõ példában a kötet egyetlen eret tartalmaz, amiben pedig egyetlen allemez van.

Az itt bemutatott kötetnek nincs semmilyen elõnye a hagyományos lemezpartícionáláshoz képest. Egyetlen eret tartalmaz, tehát nem is redundáns. Az ér egyetlen allemezt tartalmaz, tehát nem tér el a megszokott lemezpartíciók helyfoglalásától sem. A következõ szakaszokban sokkal érdekesebb konfigurációs módszereket is illusztrálunk.

21.6.2. Megnövelt rugalmasság: tükrözés

A kötetek rugalmassága tükrözéssel növelhetõ. Egy tükrözött kötet kiosztása során feltétlenül gondoskodnunk kell arról, hogy az egyes erekhez tartozó allemezek eltérõ meghajtókon találhatóak, így az esetleges meghibásodások nem károsítják mind a két eret. Az alábbi konfigurációban egy kötetet tükrözünk:

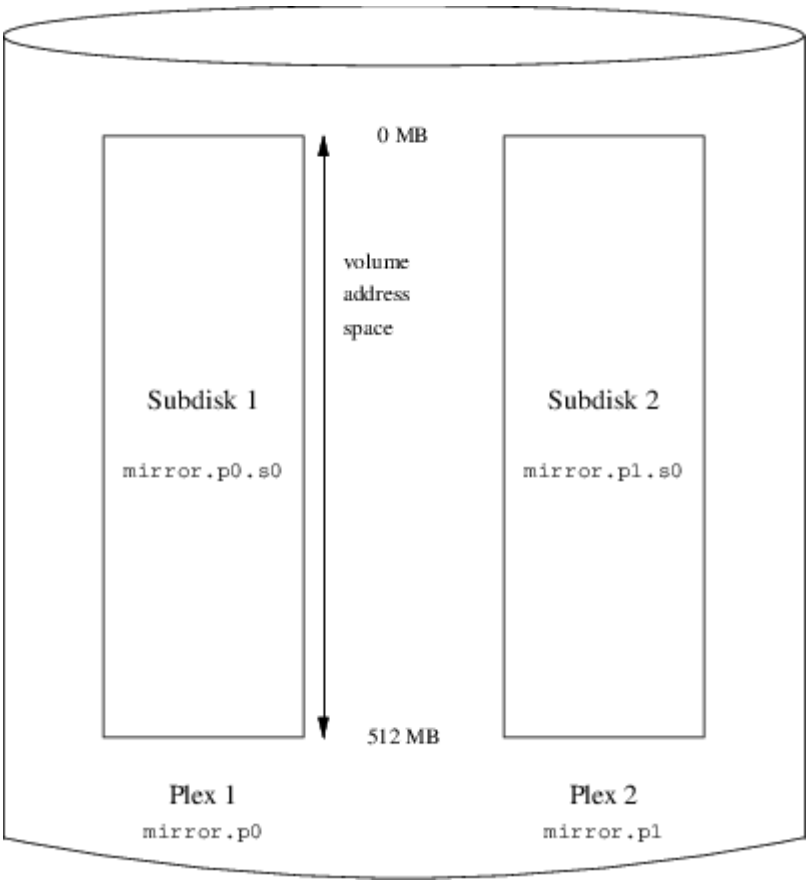
```
drive b device /dev/da4h
volume mirror
  plex org concat
    sd length 512m drive a
  plex org concat
    sd length 512m drive b
```

Ebben a példában már nem kellett újra megadnunk az *a* meghajtót, mivel a Vinum figyelemmel kíséri az összes objektumot a saját konfigurációs adatbázisában. A definíció feldolgozása után a

konfiguráció így fog kinézni:

```
    Drives:         2 (4 configured)
    Volumes:        2 (4 configured)
    Plexes:         3 (8 configured)
    Subdisks:       3 (16 configured)

    D a                     State: up       Device /dev/da3h        Avail: 1549/2573
MB (60%)
    D b                     State: up       Device /dev/da4h        Avail: 2061/2573
MB (80%)

    V myvol                 State: up       Plexes:       1 Size:        512 MB
    V mirror                State: up       Plexes:       2 Size:        512 MB

    P myvol.p0            C State: up       Subdisks:     1 Size:        512 MB
    P mirror.p0           C State: up       Subdisks:     1 Size:        512 MB
    P mirror.p1           C State: initializing     Subdisks:     1 Size:        512
MB

    S myvol.p0.s0           State: up       PO:        0  B Size:        512 MB
    S mirror.p0.s0          State: up       PO:        0  B Size:        512 MB
    S mirror.p1.s0          State: empty    PO:        0  B Size:        512 MB
```

A Tükrözött Vinum-kötet ugyanezt a szerkezetet grafikusan is.



Ábra 63. Tükrözött Vinum-kötet

Ebben a példában minden ér tartalmazza a teljes 512 MB-os területet. Ahogy a korábbi példa esetén, itt is mindegyik ér csak egyetlen allemezt tartalmaz.

21.6.3. A teljesítmény javítása

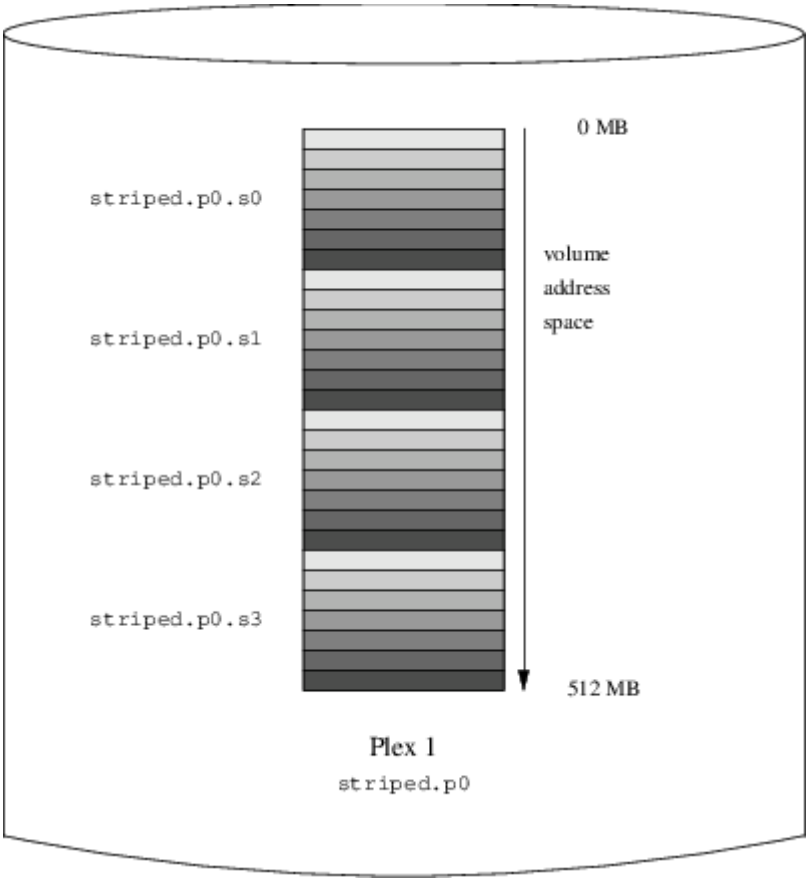
Az elõbbi példában szereplõ tükrözött kötet egy tükrözetlen kötetnél már jobban ellenáll a hibáknak, azonban a teljesítménye is kisebb. A köteten minden egyes írás mind a két meghajtóra érvényesül, ezáltal a lemezek teljes sávszélességét nagyobb arányban használja. A teljesítményre vonatkozó megfontolásaink egy másik megközelítést kívánnak meg: a tükrözés helyett inkább csíkozzuk szét az adatot a lehetõ legtöbb lemezen. Az alábbi konfiguráció egy olyan kötetet mutat be, ahol egy eret négy lemezmeghajtóan keresztül csíkozunk:

```
    drive c device /dev/da5h
    drive d device /dev/da6h
    volume stripe
    plex org striped 512k
      sd length 128m drive a
      sd length 128m drive b
      sd length 128m drive c
      sd length 128m drive d
```

Mint ahogy azt már korábban is említettük, nem szükséges még egyszer megadni azokat a meghajtókat, amiket a Vinum már ismer. A definíció feldolgozása után a konfigurációnk nagyjából így néz ki:

```
    Drives:         4 (4 configured)
    Volumes:        3 (4 configured)
    Plexes:         4 (8 configured)
    Subdisks:       7 (16 configured)

    D a                     State: up       Device /dev/da3h        Avail: 1421/2573
MB (55%)
    D b                     State: up       Device /dev/da4h        Avail: 1933/2573
MB (75%)
    D c                     State: up       Device /dev/da5h        Avail: 2445/2573
MB (95%)
    D d                     State: up       Device /dev/da6h        Avail: 2445/2573
MB (95%)

    V myvol                 State: up       Plexes:       1 Size:        512 MB
    V mirror                State: up       Plexes:       2 Size:        512 MB
    V striped               State: up       Plexes:       1 Size:        512 MB

    P myvol.p0            C State: up       Subdisks:     1 Size:        512 MB
    P mirror.p0           C State: up       Subdisks:     1 Size:        512 MB
    P mirror.p1           C State: initializing     Subdisks:     1 Size:        512
MB
    P striped.p1            State: up       Subdisks:     1 Size:        512 MB
```

```
S myvol.p0.s0           State: up       PO:        0  B Size:        512 MB
S mirror.p0.s0          State: up       PO:        0  B Size:        512 MB
S mirror.p1.s0          State: empty    PO:        0  B Size:        512 MB
S striped.p0.s0         State: up       PO:        0  B Size:        128 MB
S striped.p0.s1         State: up       PO:      512 kB Size:        128 MB
S striped.p0.s2         State: up       PO:     1024 kB Size:        128 MB
S striped.p0.s3         State: up       PO:     1536 kB Size:        128 MB
```



Ábra 64. Csíkozott Vinum-kötet

Ez a kötet a Csíkozott Vinum-kötetban látható. A csíkok sötétedése jelzi a helyüket az ér területében: a világosabbak elöl, a sötétebbek hátul szerepelnek.

21.6.4. Rugalmasság és teljesítmény

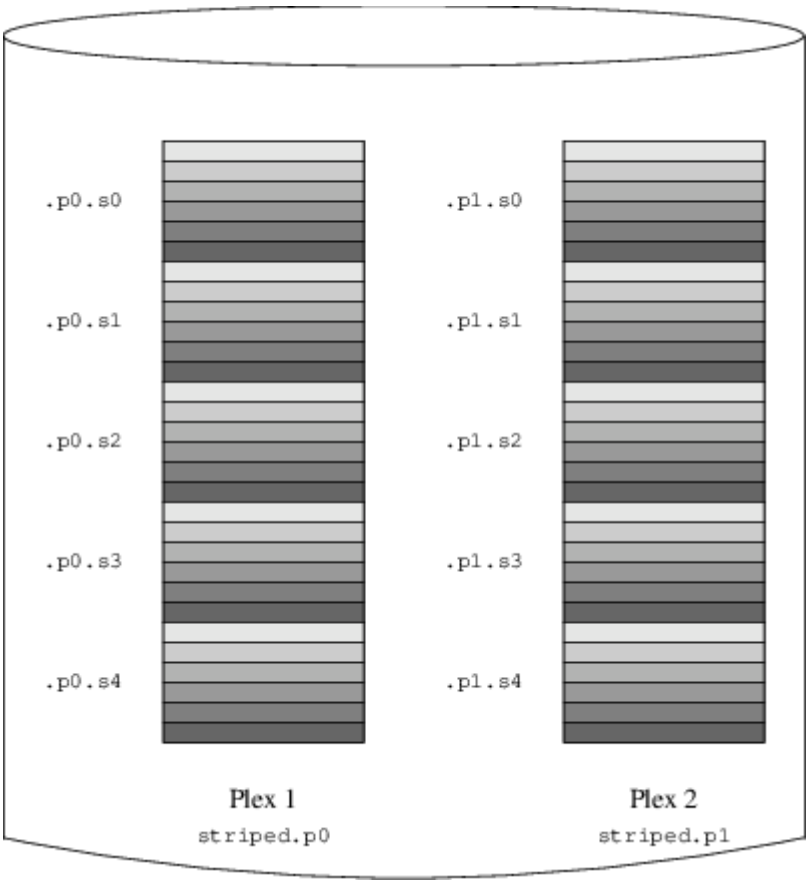
Megfelelõ hardver birtokában lehet olyan köteteket is építeni, amelyek mind megnövelt rugalmasságot, mind pedig megnövelt teljesítményt mutatnak a szabványos UNIX®-os partíciókhoz képest. Ennek a konfigurációs állománya így nézne ki:

```
volume raid10
  plex org striped 512k
    sd length 102480k drive a
    sd length 102480k drive b
    sd length 102480k drive c
    sd length 102480k drive d
    sd length 102480k drive e
```

```
      plex org striped 512k
        sd length 102480k drive c
        sd length 102480k drive d
        sd length 102480k drive e
        sd length 102480k drive a
        sd length 102480k drive b
```

A második ér allemezei el vannak tolva az elsõ ér allemezeitõl két meghajtónyival. Ez segít megelõzni, hogy az írási mûveletek ne ugyanarra az allemezre vonatkozznak, még akkor is, ha az adatátvitel két meghajtón is keresztülível.

A Tükrözött, csíkozott Vinum-kötet illusztrálja ennek a kötetnek a szerkezetét.



Ábra 65. Tükrözött, csíkozott Vinum-kötet

21.7. Az objektumok elnevezése

Korábban már megismerhettük, hogy a Vinum alapértelmezett neveket társít az erekhez és az allemezekhez, habár ezek a nevek felülbírálhatóak. Ez viszont egyáltalán nem ajánlott, mivel már a VERITAS kötetkezelõ, ahol tetszõleges neveket rendelhetünk az objektumokhoz, használata során kiderült, hogy akkora mértékû rugalmasságot nem kínál fel, mint amennyi zavart képes okozni.

A nevek tartalmazhatnak bármilyen nem üres karaktert, azonban érdemes inkább csak betûket, számjegyeket és az aláhúzást használni. A kötetek, erek és allemezek nevei akár 64 karakteresek is lehetnek, a meghajtók nevei pedig 32 karakteresek.

A Vinum objektumai a /dev/gvinum könyvtáron belüli hierarchiában helyezkednek el

eszközleírókként. Az imént említett példakonfiguráció hatására a következõ eszközleírók jönnek létre:

Ez a rész csak a Vinum korábbi, elavult implementációjára vonatkozik.

A /dev/vinum/control és /dev/vinum/controld nevû vezérlõeszközök, melyeket a gvinum(8) és a Vinum démon használ. * Mindegyik kötethez egy eszközleíró tartozik. Ezek a Vinum számára a központi eszközök, ezért az elõbbi konfiguráció révén megjelennek a /dev/gvinum/myvol, /dev/gvinum/mirror, /dev/gvinum/striped, /dev/gvinum/raid5 és /dev/gvinum/raid10 eszközök.

Ez a rész csak a Vinum korábbi, elavult implementációjára vonatkozik.

Az egyes meghajtókhoz tartozó leírók a /dev/vinum/drive könyvtárban találhatóak. Ezek valójában szimbolikus linkek a megfelelõ lemezes eszközökre. * Minden kötethez közvetlen leírók tartoznak /dev/gvinum könyvtárban. * Az egyes erek és allemezek eszközleírói a /dev/gvinum/plex és /dev/gvinum/sd könyvtárakban jelennek meg.

Például tekintsük most az alábbi konfigurációs állományt:

```
drive drive1 device /dev/sd1h
drive drive2 device /dev/sd2h
drive drive3 device /dev/sd3h
drive drive4 device /dev/sd4h
volume s64 setupstate
  plex org striped 64k
    sd length 100m drive drive1
    sd length 100m drive drive2
    sd length 100m drive drive3
    sd length 100m drive drive4
```

Az állomány feldolgozása után az eszközleírókat a gvinum(8) az alábbi módon szervezi a /dev/gvinum könyvtárban:

```
drwxr-xr-x  2 root  wheel       512 Apr 13 16:46 plex
crwxr-xr--  1 root  wheel   91,   2 Apr 13 16:46 s64
drwxr-xr-x  2 root  wheel       512 Apr 13 16:46 sd

/dev/vinum/plex:
total 0
crwxr-xr--  1 root  wheel   25, 0x10000002 Apr 13 16:46 s64.p0

/dev/vinum/sd:
total 0
crwxr-xr--  1 root  wheel   91, 0x20000002 Apr 13 16:46 s64.p0.s0
crwxr-xr--  1 root  wheel   91, 0x20100002 Apr 13 16:46 s64.p0.s1
crwxr-xr--  1 root  wheel   91, 0x20200002 Apr 13 16:46 s64.p0.s2
crwxr-xr--  1 root  wheel   91, 0x20300002 Apr 13 16:46 s64.p0.s3
```

Jóllehet, az ereket és allemezeket nem ajánlott külön-külön elnevezni, a Vinum meghajtóknak nevet kell adni. Ezzel megoldhatóvá válik, hogy az egyes meghajtók automatikusan felismerhetõek legyenek abban az esetben is, amikor fizikailag áthelyezzük ezeket. A meghajtók nevei legfeljebb 32 karakteresek lehetnek.

21.7.1. Állományrendszerek létrehozása

A kötetek egyetlen kivétellel teljesen azonosak a lemezekkel a rendszer számára. Ugyanis a UNIX®-os meghajtóktól eltérõen a Vinum nem particionálja a köteteket, és ezért nem is tárolnak partíciós táblát. Ez megkövetelte néhány lemezkezelõ segédprogram, leginkább a newfs(8) módosítását, mivel azok korábban megpróbálták a Vinum-kötetek nevének utolsó betûit egy partíció azonosítójaként értelmezni. Például egy lemezes meghajtó neve /dev/ad0a vagy /dev/da2h alakú. Az elõbbi az elsõ (0) IDE lemez elsõ (a) partícióját, míg az utóbbi a harmadik (2) SCSI lemez nyolcadik (h) partícióját jelöli. Ezzel szemben azonban a Vinum-kötetek neve /dev/gvinum/concat alakú lesz, ahol a név semmilyen kapcsolatban nem áll a partíció nevével.

Hétköznapi esetben a newfs(8) megpróbálja a lemez nevét értelmezni, és panaszkodik, ha nem sikerül. Például:

```
# newfs /dev/gvinum/concat
newfs: /dev/gvinum/concat: can't figure out file system partition
```

A köteten a newfs(8) parancs kiadásával tudunk állományrendszert létrehozni:

```
# newfs /dev/gvinum/concat
```

A FreeBSD 5.0 elõtt verzióiban a newfs(8) parancsnak a régi elnevezési séma használata mellett még át kell adni egy -v kapcsolót is:

```
# newfs -v /dev/vinum/concat
```

21.8. A Vinum beállítása

A GENERIC rendszermag nem tartalmazza a Vinumot. Habár készíteni lehet olyan rendszermagot, amelyik támogatja a Vinumot, mégsem ajánlott. A Vinumot a szabványos módon modulként (kld) indíthatjuk el. Még a kldload(8) használatára sincs szükség, mivel a gvinum(8) indulása során ellenõrzi a modul jelenlétét és betölti, ha még nem lenne jelen.

21.8.1. Indítás

A Vinum alapvetõen ugyanúgy tárolja a konfigurációkat a slice-okban, mint maguk a konfigurációs állományok. A konfigurációs adatbázis beolvasása során a Vinum felismeri azokat a kulcsszavakat, amelyeknek nem szabad elõfordulniuk az állományokban. Például a lemezek beállítása tartalmazhatja a következõ szöveget:

```
volume myvol state up
volume bigraid state down
plex name myvol.p0 state up org concat vol myvol
plex name myvol.p1 state up org concat vol myvol
plex name myvol.p2 state init org striped 512b vol myvol
plex name bigraid.p0 state initializing org raid5 512b vol bigraid
sd name myvol.p0.s0 drive a plex myvol.p0 state up len 1048576b driveoffset 265b
plexoffset 0b
sd name myvol.p0.s1 drive b plex myvol.p0 state up len 1048576b driveoffset 265b
plexoffset 1048576b
sd name myvol.p1.s0 drive c plex myvol.p1 state up len 1048576b driveoffset 265b
plexoffset 0b
sd name myvol.p1.s1 drive d plex myvol.p1 state up len 1048576b driveoffset 265b
plexoffset 1048576b
sd name myvol.p2.s0 drive a plex myvol.p2 state init len 524288b driveoffset 1048841b
plexoffset 0b
sd name myvol.p2.s1 drive b plex myvol.p2 state init len 524288b driveoffset 1048841b
plexoffset 524288b
sd name myvol.p2.s2 drive c plex myvol.p2 state init len 524288b driveoffset 1048841b
plexoffset 1048576b
sd name myvol.p2.s3 drive d plex myvol.p2 state init len 524288b driveoffset 1048841b
plexoffset 1572864b
sd name bigraid.p0.s0 drive a plex bigraid.p0 state initializing len 4194304b driveoff
set 1573129b plexoffset 0b
sd name bigraid.p0.s1 drive b plex bigraid.p0 state initializing len 4194304b driveoff
set 1573129b plexoffset 4194304b
sd name bigraid.p0.s2 drive c plex bigraid.p0 state initializing len 4194304b driveoff
set 1573129b plexoffset 8388608b
sd name bigraid.p0.s3 drive d plex bigraid.p0 state initializing len 4194304b driveoff
set 1573129b plexoffset 12582912b
sd name bigraid.p0.s4 drive e plex bigraid.p0 state initializing len 4194304b driveoff
set 1573129b plexoffset 16777216b
```

Az elõbbiektõl nyilvánvalóan eltér abban, hogy itt már megjelennek a konkrét pozíciókra és elnevezésekre vonatkozó információk (melyeket a felhasználó is megadhat, azonban ezt nem tanácsoljuk), valamint az állapotok (ezeket nem láthatja a felhasználó). A Vinum a konfigurációban nem tárolja a meghajtókat, helyette a beállított lemezes meghajtók partícióin fog Vinum-címkéket keresni. Ennek köszönhetõen a Vinum még akkor is képes pontosan megtalálni a meghajtókat, amikor megváltoznak a hozzá tartozó UNIX®-os meghajtók azonosítói.

Automatikus indítás

Ez a rész csak a Vinum elavult implementációjára vonatkozik. A loader.conf(5) közvetítésével a *Gvinum* mindig automatikusan elindul a hozzá tartozó modul betöltésével együtt. Ha a rendszerindításkor be akarjuk tölteni a *Gvinum* modult, akkor a /boot/loader.conf állományba vegyük fel a `geom_vinum_load="YES"` sort.

Az alábbi sort mindenképpen hozzá kell adnunk az /etc/rc.conf állományhoz, hogy a Vinum a rendszerindítás során automatikusan elinduljon:

```
start_vinum="YES"       # állítsuk YES-re az indításhoz
```

Hozzuk létre és írjuk bele, ha nem lenne /etc/rc.conf nevû állományunk. Ennek hatására a rendszer az indulás során betölti a Vinum kld modult, és a konfigurációban szereplõ objektumokat elindítja. Ez még az állományrendszerek csatlakoztatása elõtt történik meg, aminek révén a Vinum-köteteken található állományrendszereket a rendszer automatikusan át tudja vizsgálni az fsck(8) segítségével, majd csatlakoztatja ezeket.

Amikor a Vinumot a `vinum start` paranccsal indítjuk el, a Vinum beolvassa a konfigurációs adatbázist a Vinum-meghajtók egyikérõl. Normál körülmények között mindegyik meghajtón megtalálható a konfigurációs adatbázis egy példánya, ezért szinte teljesen mindegy, melyik meghajtót is olvassa. Egy rendszer-összeomlás után azonban a Vinumnak meg kell tudnia állapítania, melyik meghajtón található meg az adatbázis legfrissebb példánya, és ezt kell beolvasnia. Ezután a lemaradt meghajtókon található adatbázispéldányokat szinkronizálja ehhez a változathoz.

21.9. Rendszerindítás Vinum-kötetrõl

Olyan számítógépeknél, ahol a teljesen tükrözött Vinum-alapú állományrendszereket használunk, kívánatos lehet magát a rendszerindításhoz használt állományrendszert is tükrözni. Egy ilyen konfiguráció összeállítása már messze nem annyira egyszerû, mint egy tetszõleges állományrendszer esetén, mivel:

- Az indításhoz használt állományrendszernek már a folyamat nagyon korai szakaszában rendelkezésre kell állnia, ezért a Vinumnak már itt elérhetõnek kell lennie.
- A rendszerindító állományrendszert tartalmazó köteten még ott kell lennie a rendszerindító kódnak és a rendszermagnak is, melyeket a rendszer saját eszközein (például ilyen a BIOS a PC-knél) keresztül kell tudnunk beolvasni, amiket viszont nem tudunk felkészíteni a Vinumra.

A soronkövetkezõ szakaszokban "rendszerindító kötetként" (root volume) fogunk általánosságban véve hivatkozni a rendszerindításhoz használt állományrendszert tartalmazó Vinum-kötetre. Ennek megfelelõen valószínûleg jó ötlet a `"root"` névvel azonosítani ezt a kötetet, habár technikai szempontból ezt semmi nem követeli meg. Az itt felsorakozó példákban azonban ezt a nevet fogjuk használni.

21.9.1. A Vinum kellõen korai indítása

Ennek kiváltásához számos lépést kell megtennünk:

- A rendszermagnak már el kell érnie a Vinumot a rendszerindítás során. Emiatt a Automatikus indításban leírt automatikus indítási módszer nem alkalmazható erre a feladatra, és a `start_vinum` paramétert *nem* is szabad használni a most ismertetendõ konfigurációban. A Vinumot statikusan bele is építhetjük a rendszermagba és így állandóan elérhetõ, de ez általában nem kielégítõ megoldás. Megoldhatjuk úgy is, ha a /boot/loader-re (A harmadik fokozat (/boot/loader)) bízzuk a vinum modul betöltését, még a rendszermag elõtt. Ezt az alábbi sorral válthatjuk ki a /boot/loader.conf állományban:

```
geom_vinum_load="YES"
```

A *Gvinum* használata során az összes többi beállítás automatikusan végrehajtódik, amint a modul betöltõdik, ezért ilyenkor csak a fentebb leírt eljárásra van szükség. Az itt felsoroltak csak az elavult Vinum implementációra vonatkoznak, csupán a régebbi típusú rendszerek kedvéért említjük meg.

A Vinumot nagyon korán életre kell keltenünk, hiszen a rendszerindításhoz használt állományrendszert tartalmazó kötetet kell élesítenünk. Alapértelmezés szerint a Vinum rendszerszinten futó része nem keres addig semmilyen Vinum-kötetinformációval rendelkezõ meghajtót, amíg a rendszergazda (vagy valamelyik rendszerindító szkript) ki nem adja a `vinum start` parancsot.

A most következõ bekezdések mutatják be a szükséges lépéseket.

Ha hozzáadjuk a következõ sort a /boot/loader.conf állományhoz, akkor azzal utasíthatjuk a Vinumot, hogy a rendszermag indítása során vizsgálja át az összes meghajtót:

```
vinum.autostart="YES"
```

Nem szükséges megmondani a rendszermagnak, merre keresse a rendszerindításhoz használt állományrendszert. A /boot/loader megkeresi a hozzá tartozó eszközt a /etc/fstab állományban és átadja ezt az információt a rendszermagnak. Amikor a csatlakoztatására kerül sor, a rendszermag az eszköz nevébõl meg tudja állapítani, melyik eszközmeghajtót kérje meg a belsõ (fõ- és al)eszközazonosító leképzéséhez.

21.9.2. A Vinum-alapú rendszerindító kötet elérése a rendszertöltés során

Mivel a jelenlegi FreeBSD rendszertöltõ csak 7,5 KB méretû és egyébként is csak az UFS állományrendszerrõl tud állományokat beolvasni (mint például a /boot/loadert), teljesen lehetetlen még a Vinum belsõ szerkezetére is megtanítani, tehát a Vinum-konfigurációk értelmezésére és magának a rendszerindító kötet elemeinek kielemzésére. Ezért be kell vetnünk néhány trükköt ahhoz, hogy a rendszerindító kód számára a rendszerindításhoz használható szabványos `"a"` partíció képzetét keltsük.

Mindez csak akkor válik elérhetõvé, ha az alábbi követelményeket teljesíti a rendszerindító kötet:

- Nem lehet csíkozott vagy RAID-5 típusú.
- Erenként nem tartalmazhat egynél több összefûzött allemezt.

Láthatjuk, hogy hasznos és lehetséges is több eret használni, melyek mindegyike a rendszerindító állományrendszer egy-egy másolatát tartalmazza. Az indulás folyamán azonban ezen példányok közül csak az egyiken fogja keresni a rendszer a rendszertöltõt és a többi állományt egészen addig, amíg a rendszermag magát az állományrendszert nem csatlakoztatja. A látszat kedvéért az ereken belül található allemezek mindegyikének lennie kell egy saját `"a"` partíciójának, amivel lényegében alkalmassá válik a rendszerindításra. Ezeknek a hamis `"a"` partícióknak nem kell feltétlenül a

többiekkel megegyezõ pozíciókon elhelyezkedniük, azonban a tévedések elkerülése érdekében valószínûleg hasznos olyan Vinum-köteteket létrehozni, ahol a keletkezõ tükrözött eszközök szimmetrikusak.

A rendszerindító kötet egyes eszközökön található "a" partícióit az alábbiak segítségével állíthatjuk be:

1. A rendszerindító kötet részeként megjelenõ eszközön található allemez helyét (az eszköz elejétõl számított eltolását) és méretét ellenõrizni kell az alábbi parancs segítségével:

   ```
   # gvinum l -rv root
   ```

   Ne felejtsük el, hogy a Vinum az eltolásokat és méreteket byte-okban méri. Ezekbõl tehát úgy nyerünk a `bsdlabel` használatához szükséges blokkszámokat, ha ezeket elosztjuk 512-vel.

2. Futassuk le a

   ```
   # bsdlabel -e eszköznév
   ```

   parancsot minden olyan eszközön, amelyik részt vesz a rendszerindító kötet kialakításában. Az *eszköznév* legyen a slice (fdisk)-táblát nem tartalmazó lemezek esetén a lemez neve (mint például da0), vagy ellenkezõ esetben a slice neve (például ad0s1).

   Ha már lenne egy "a" partíció az eszközön (valószínûleg egy Vinum elõtti rendszeríndító állományrendszert tartalmaz), nevezzük át valami másra és így továbbra is elérhetõ marad (biztos, ami biztos), viszont többé már nem lesz a rendszer számára alapértelmezett rendszerindító eszköz. Az aktív partíciók (mint például az éppen csatlakoztatott rendszerindító állományrendszer) nem nevezhetõek át, ezért ezt a lépést csak akkor tudjuk megtenni, ha a rendszerünket egy "Fixit" (Helyreállító) eszközrõl indítjuk, vagy egy olyan kétlépéses folyamat során, ahol (tükrözés esetén) a lemezrõl még nem indítottuk el a rendszert.

   Ezt követõen az eszközön található Vinum-partíciót (amennyiben létezik) az eszközön levõ allemez eltolásához kell helyezni. Ennek eredménye lesz az új "a" partíció "offset" értéke. A partíció "size" (méret) értéke szó szerint átemelhetõ a fenti számításból. Az "fstype" legyen 4.2BSD. Az "fsize", "bsize" és "cpg" értékeket a jelenlegi állományrendszerhez mérten ajánlott megválasztani, azonban itt most egyáltalán nem bírnak jelentõséggel.

   Ezzel a módszerrel létesítettünk egy olyan új "a" partíciót, amely lefedi az eszközön található Vinum-partíciót. Jegyezzük meg, hogy a `bsdlabel` kizárolag csak abban az esetben fogja megengedi ezt az átfedést, ha a Vinum-partíciónk "vinum" típussal van megjelölve.

3. Készen is vagyunk! Most már van minden eszközön egy hamisított "a" partíciónk, amelyeken megtalálható a rendszerindító kötet egy-egy másolata. Határozottan ajánlott még egyszer ellenõrizni a munkánkat az alábbi parancs kiadásával:

```
# fsck -n /dev/eszköznéva
```

Figyeljünk arra, hogy az összes vezérlési információt tartalmazó állománynak a Vinum-köteten található rendszerindító állományrendszerre kell vonatkoznia, ami viszont egy új Vinum rendszerindító kötet beállítása után nem feltétlenül egyezik meg a jelenlegi aktív állományrendszerrel. Különösen az /etc/fstab és /boot/loader.conf állományokat kell ilyen szempontból ellenõriznünk.

A következõ indítás során a rendszertöltõ már az új Vinum-alapú rendszerindító állományrendszerrõl fogja összeszedni a mûködéséhez szükséges adatokat és ezeknek megfelelõen cselekedni. Végül, a rendszermag inicializálója után, mikor az összes eszközt felismerte, egy ehhez hasonló feltûnõ üzenet fogja jelezni a beállítás sikerességét:

```
Mounting root from ufs:/dev/gvinum/root
```

21.9.3. Egy Vinum-alapú rendszerindító állományrendszer példája

Miután sikeresen konfiguráltuk a rendszerindító Vinum-kötetet, a `gvinum l -rv root` kimenete nagyjából így fog kinézni:

```
...
Subdisk root.p0.s0:
        Size:        125829120 bytes (120 MB)
        State: up
        Plex root.p0 at offset 0 (0  B)
        Drive disk0 (/dev/da0h) at offset 135680 (132 kB)

Subdisk root.p1.s0:
        Size:        125829120 bytes (120 MB)
        State: up
        Plex root.p1 at offset 0 (0  B)
        Drive disk1 (/dev/da1h) at offset 135680 (132 kB)
```

Itt (a /dev/da0h partícióhoz képesti) `135680`-as eltoltás értékekre kell figyelnünk. Ez képzõdik le a `bsdlabel` fogalmi rendszerében aztán 265 darab 512 byte-os blokkra a lemezen. Ehhez hasonlóan a rendszerindító kötet mérete 245 760 darab 512 byte-os blokk lesz. A rendszerindító kötet másodpéldányát tartalmazó /dev/da1h ugyanilyen beállításokkal rendelkezik.

Az említett eszközök valahogy így jelennek meg a `bsdlabel` szerint:

```
...
8 partitions:
#        size   offset    fstype   [fsize bsize bps/cpg]
  a:   245760      281    4.2BSD     2048 16384     0   # (Cyl.    0*- 15*)
  c: 71771688        0    unused        0     0         # (Cyl.    0 - 4467*)
```

```
  h: 71771672       16     vinum                    # (Cyl.    0*- 4467*)
```

Megfigyelhetõ, hogy a hamis "a" partíció "size" paraméter értéke megegyezik a fentebb becsült értékkel, miközben az "offset" paraméter értéke egyenlõ lesz a "h" Vinum-partíción belüli eltolás és az eszközön (vagy slice-on) belüli eltolás összegével. Ez jellemzõen egy olyan beállítás, amivel szükségszerûen el tudjuk kerülni a Semmi sem indul, a rendszertöltõ hibákat írban leírt hibajelenséget. Látható továbbá az is, hogy az egész "a" partíció végig az eszköz összes Vinum adatát tartalmazó "h" partíciójában foglal helyet.

A példával kapcsolatban megjegyezzük, hogy itt az egész eszközt a Vinum felügyelete alá bocsátottuk, tehát nem marad hátra semmilyen Vinum elõtt használt rendszerindító partíció, hiszen ez egy olyan lemez, amelyet eleve egy Vinum-konfigurációba szántunk.

21.9.4. Hibakeresés

Fontos tudunk, hogy probléma esetén hogyan tudjuk helyreállítani a rendszerünket. A következõ felsorolásban bemutatunk néhány ismert buktatót és a megoldásaikat.

A rendszertöltõ elindul, de a rendszer viszont már nem

Ha valamilyen okból a rendszer nem indulna el, a 10 másodpercig tartó visszaszámlálás során a rendszertöltõt még meg tudjuk állítani a `szóköz` lenyomásával. Ekkor a betöltõ által használt változók (mint például a `vinum.autostart`) a `show` segítségével megvizsgálhatóak és a `set` vagy `unset` parancsokkal módosíthatóak.

Ha mindössze az volt a probléma, hogy a Vinum modulja nem szerepelt az automatikusan betöltendõ modulok között, a `load geom_vinum` parancs kiadásával betölthetjük azt.

Miután végeztünk, a rendszerindítás folyamata a `boot -as` paranccsal folytatható. A `-as` kapcsolók jelzik a rendszermag számára, hogy kérdezzen rá a rendszerindító állományrendszerre a csatlakoztatása elõtt (`-a`) és csak egyfelhasználós módban indítsa a rendszert (`-s`), ahol a rendszerindító állományrendszer írásvédett. Így, ha csak egyetlen eret csatlakoztattunk egy többeres kötetbõl, az erek még véletlenül sem tudnak egymásnak ellentmondó állapotba kerülni.

Amikor megjelenik a csatlakoztatandó rendszerindító állományrendszert bekérése, bármelyik érvényes rendszerindításra alkalmas állományrendszer megadható. Amennyiben az /etc/fstab állományt jól beállítottuk, az alapértelmezett érték egy `ufs:/dev/gvinum/root` értékhez hasonló alakú lesz. Itt általában egy `ufs:da0d` formátumú értéket láthatunk, amely feltehetõen egy Vinum használata elõtti rendszerindító állományrendszert tartalmazó partíció. Legyünk óvatosak, ha itt egy olyan "a" partíciót adunk meg, amely valójában egy rendszerindító Vinum-eszköz allemezeire hivatkozik, mivel egy tükrözött konfiguráció esetén csak az eszköz egyik részét fogjuk csatlakoztatni. Ha a késõbbiekben ezt az állományrendszert már nem csak írásvédett módban csatlakoztatjuk, mindenképpen el kell távolítanunk a rendszerindító Vinum-kötetbõl a többi eret, mivel máskülönben nagy valószínûséggel eltérõ adatokat fognak tartalmazni.

Csak az elsõdleges rendszertöltõ indul el

Amikor az elsõdleges rendszertöltõ még elindul, viszont a /boot/loader már nem tud betöltõdni (ezt rendszerindítás megkezdése után bal oldalt rögtön megjelenõ forgó vonalból vehetjük észre), a

`szóköz` lenyomásával itt még tehetünk egy kísérletet a betöltés megszakítására. Ennek hatására a rendszertöltés megáll a második fázisban, lásd Az első fokozat (/boot/boot1) és a második fokozat (/boot/boot2). Itt a rendszerindításhoz megpróbálhatunk megadni egy másik partíciót, például egy olyat, amely a korábbi rendszerindító állományrendszert tartalmazza és amelyet az elõbb átneveztünk az `"a"`-ról.

Semmi sem indul, a rendszertöltõ hibákat ír

Ez a helyzet akkor állhat elõ, ha a Vinum telepítése során tönkretettük volna a rendszertöltõt. Sajnos a Vinum minden esetben 4 KB helyet hagy szabadon a partíció elején, a saját fejléc információjának rögzítése elõtt. Az ide kerülõ elsõ és második fázisú rendszertöltõk, illetve a bsdlabel adatai azonban jelenleg 8 KB helyet kívánnak meg. Így ha a Vinum-partíció egy rendszerindításra szánt slice vagy lemez 0. eltolásánál kezdõdik, a Vinum beállításai felül fogják írni a rendszertöltõt.

A rendszertöltõ is ugyanígy felülírja a Vinum fejlécét és akkor a Vinum nem találja a lemezeit, ha a fenti problémát orvosolva, például egy "Fixit" (Helyreállító) lemez segítségével, újratelepítjük a rendszertöltõt a Az első fokozat (/boot/boot1) és a második fokozat (/boot/boot2)ban bemutatott `bsdlabel -B` parancs segítségével. Noha a Vinum egyetlen konkrét konfigurációs beállítása vagy a kötetekben tárolt adat sem sérül meg és vissza tudjuk állítani az összes elveszett információt ugyannak a Vinum-konfigurációnak az újbóli megadásával, a helyzetet magát nehéz megoldani. A Vinum-fejléc és a rendszertöltõ ütközésének megszüntetéséhez ugyanis legalább 4 KB-tal arrébb kell mozgatnunk az egész Vinum-partíciót.

Chapter 22. Virtualizáció

22.1. Áttekintés

A virtualizációs szoftverek lehetõvé teszik, hogy ugyanazon a számítógépen egyszerre több operációs rendszert is futassunk. Ezeknek a programcsomagoknak gyakorta részük egy gazda operációs rendszer is, amely a virtualizációs szoftvert futattja és ismer bizonyos vendég operációs rendszereket.

A fejezet elolvasása során megismerjük:

- a gazda- és a vendég operációs rendszerek közti különbségeket;
- hogyan telepítsünk FreeBSD-t egy Intel®-alapú Apple® Macintosh® számítógépre;
- hogyan telepítsünk a Virtual PC használatával FreeBSD-t Microsoft® Windows®-ra;
- hogyan hozzuk ki a legtöbbet FreeBSD rendszerünkbõl virtualizáció alatt.

A fejezet elolvasásához ajánlott:

- alapvetõ UNIX®-os és FreeBSD-s ismeretek (A UNIX alapjai);
- a FreeBSD telepítésének ismerete (A FreeBSD telepítése);
- a hálózati kapcsolatok beállításának ismerete (Egyéb haladó hálózati témák);
- külsõs alkalmazások telepítésének ismerete (Alkalmazások telepítése. csomagok és portok).

22.2. A FreeBSD mint vendég

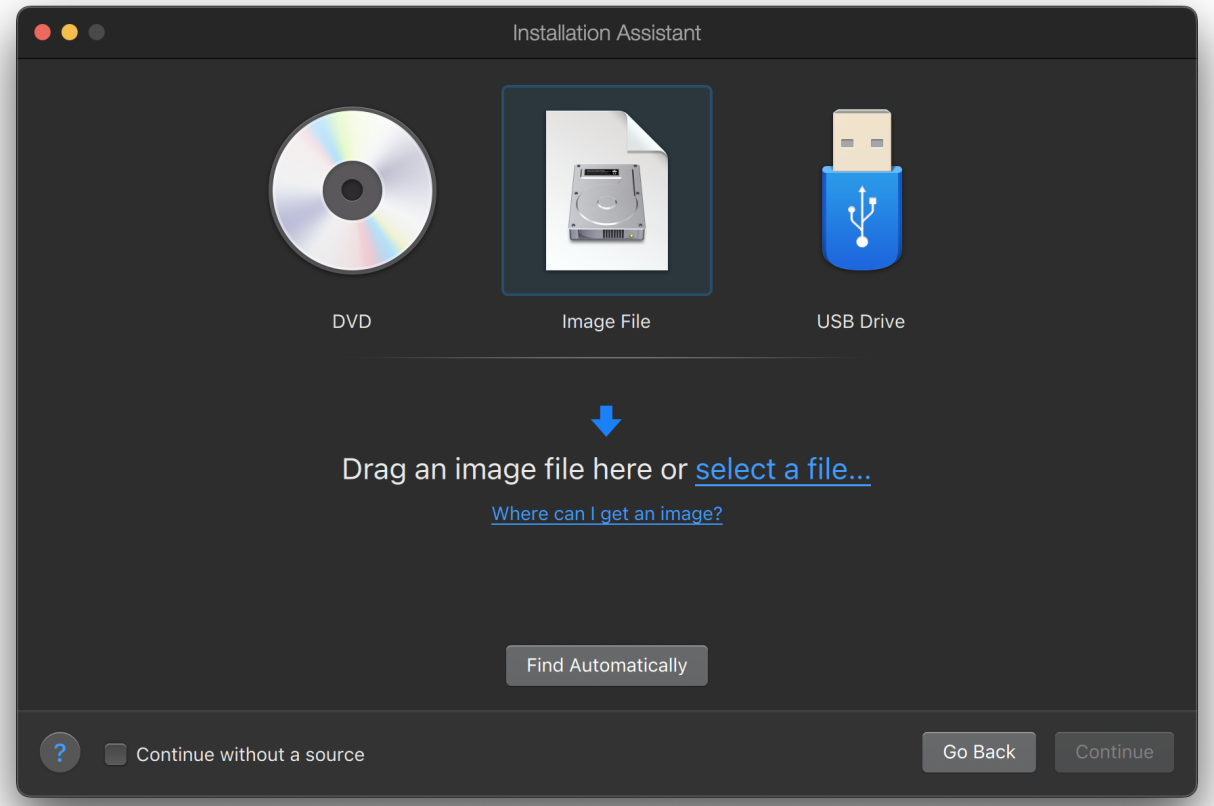
22.2.1. Parallelsszel Mac OS®-en

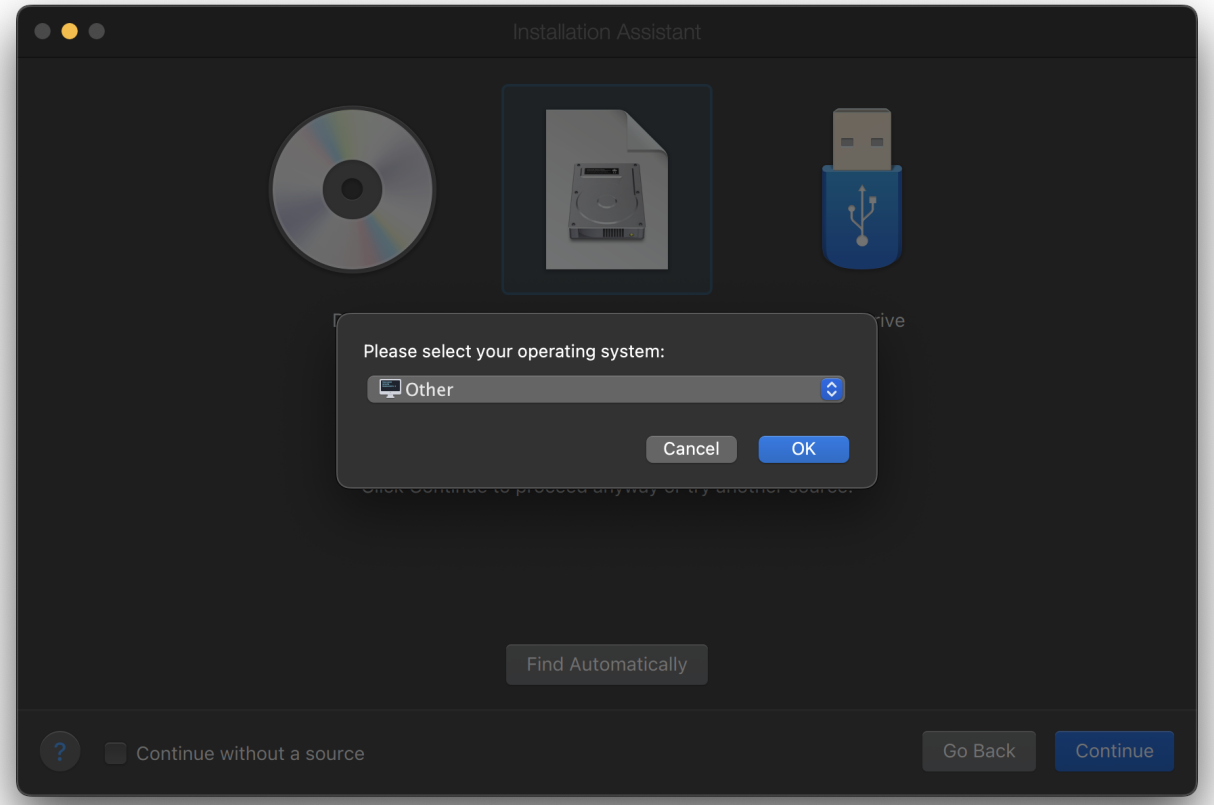
A Parallels Desktop a Mac OS® 10.4.6, vagy afeletti verzióját futattó, Intel®-alapú Apple® Mac® személyi számítógépekre fejlesztett kereskedelmi alkalmazás. A FreeBSD-t teljes mértékben támogatja vendégként. Miután telepítettük a Parallels-t a Mac OS® X-re, be kell állítanunk egy virtuális gépet, majd erre felraknunk a kívánt vendég operációs rendszert.

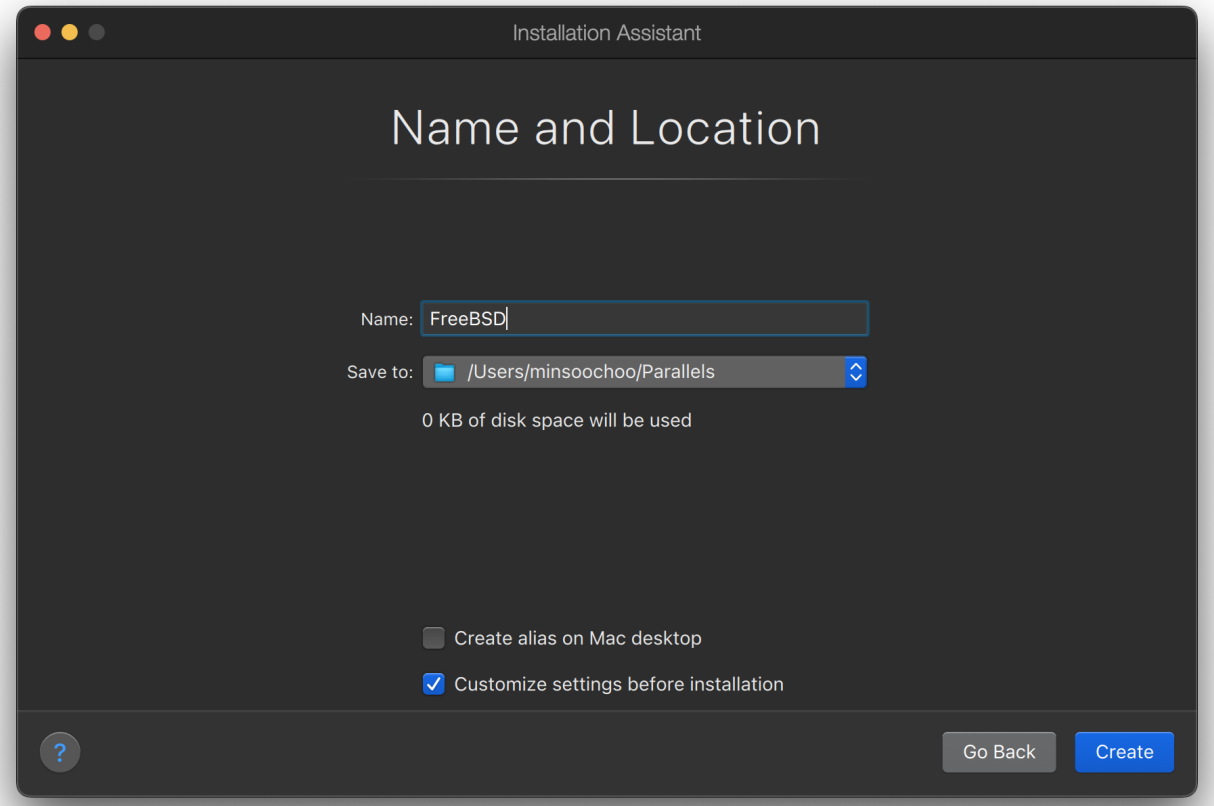
A FreeBSD telepítése Mac OS® X/Parallelsre

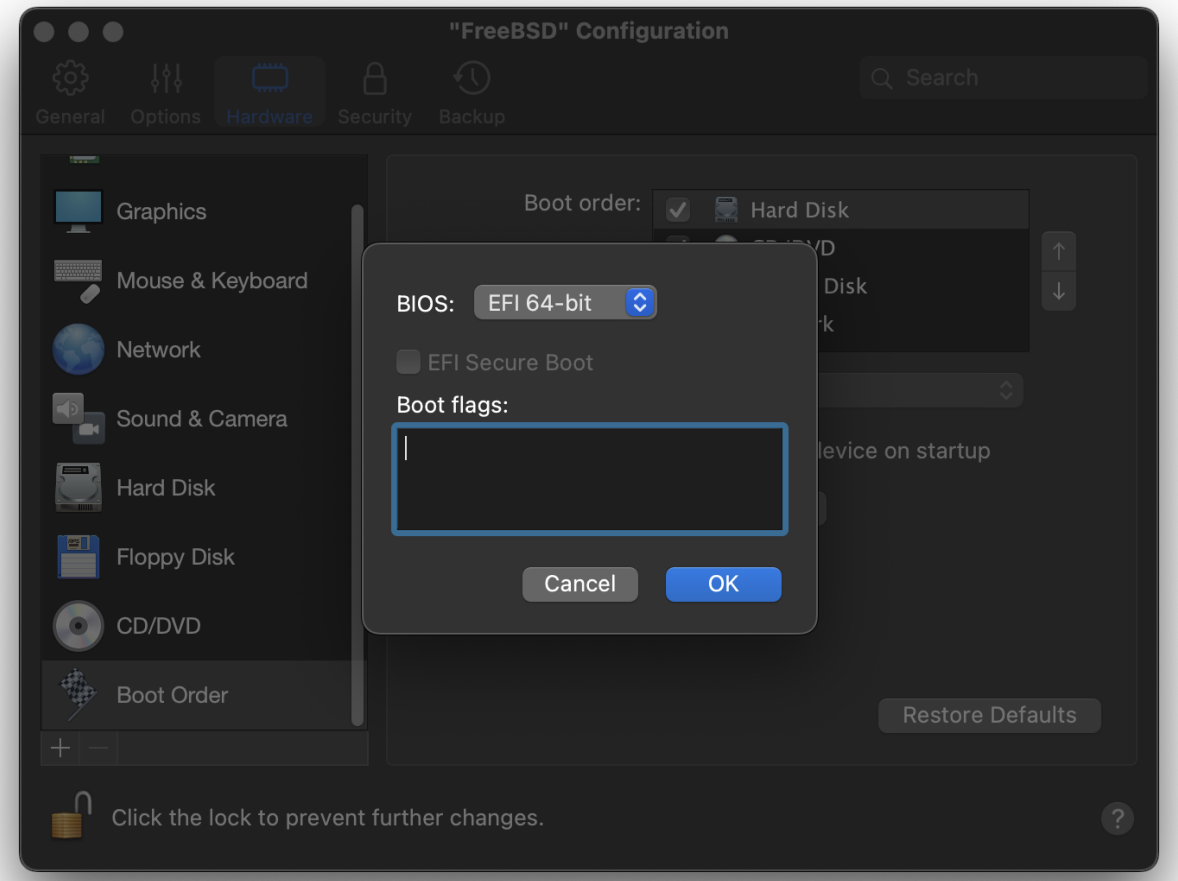
A FreeBSD Mac OS® X/Parallels párosra telepítéséhez elsõ lépésként készítenünk kell egy új virtuális számítógépet. A létrehozás során válasszuk a **Guest OS Type**-nak (a vendég operációs rendszer típusának) a FreeBSD-t:

Ezután adjunk meg egy nagyjából elfogadható méretet a virtuális merevlemezünknek, valamint annyi memóriát, amennyire szükségünk lehet a virtuális FreeBSD-nk használata során. Egy 4 GB-os lemez és 512 MB rendszermemória a legtöbb esetben jó választásnak bizonyulhat a FreeBSD Parallels alatti használata során:

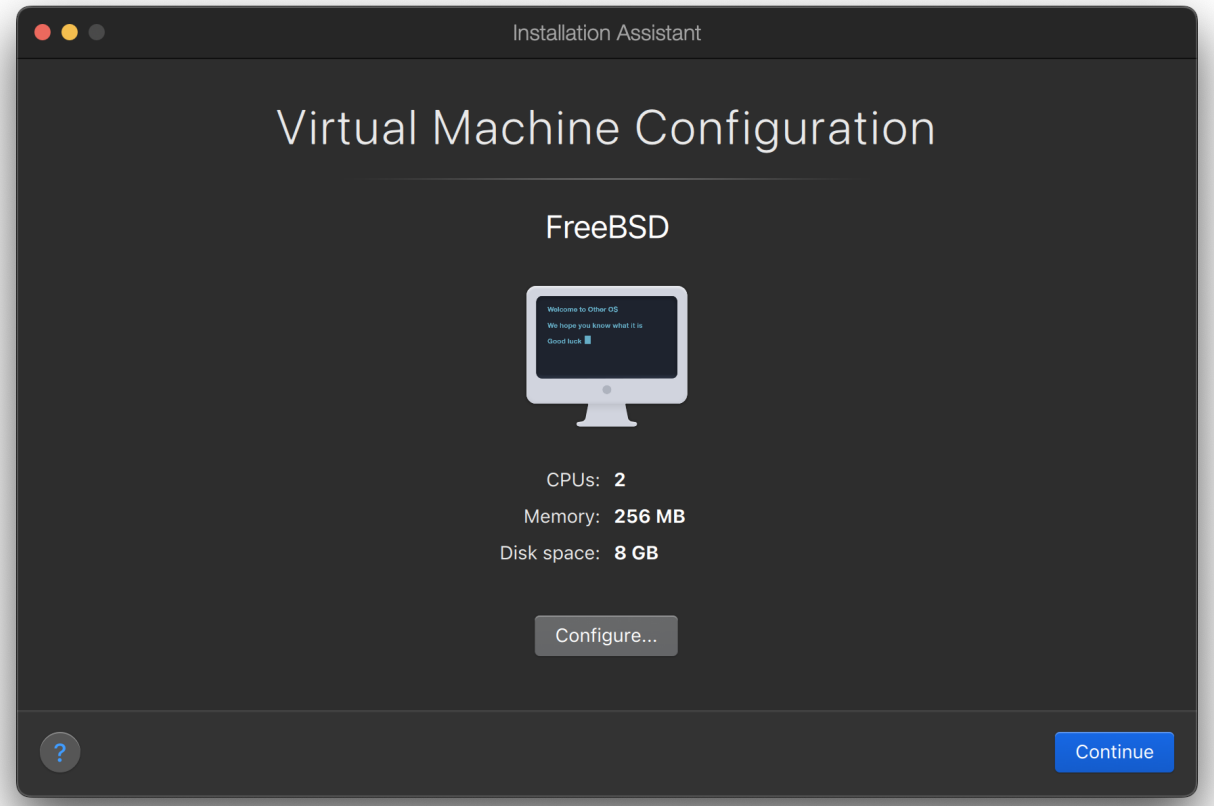

Installation Assistant
DVD
Image File
USB Drive
Drag an image file here or select a file...
Where can I get an image?
Find Automatically
Continue without a source
Go Back
Continue


Installation Assistant
Please select your operating system:
Other
Cancel
OK
Find Automatically
Continue without a source
Go Back
Continue

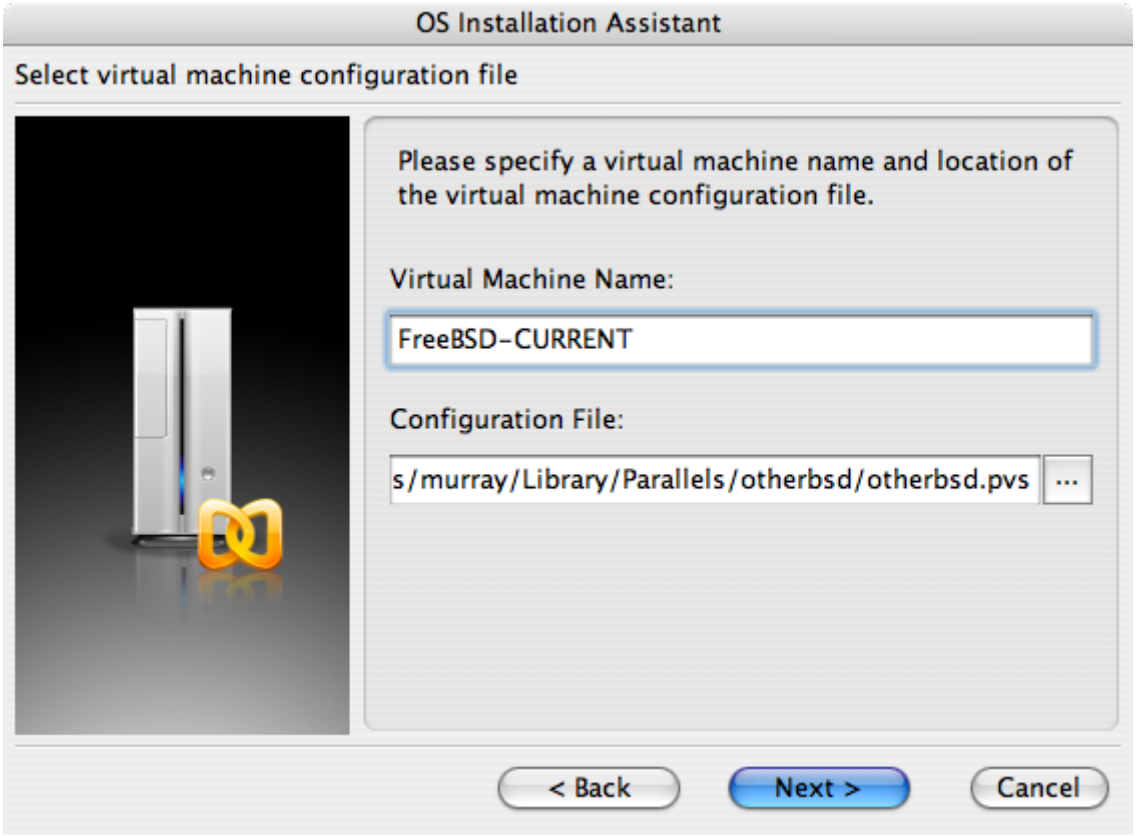

Installation Assistant
Name and Location
Name:
FreeBSD
Save to:
/Users/minsoochoo/Parallels
0 KB of disk space will be used
Create alias on Mac desktop
Customize settings before installation
Go Back
Create



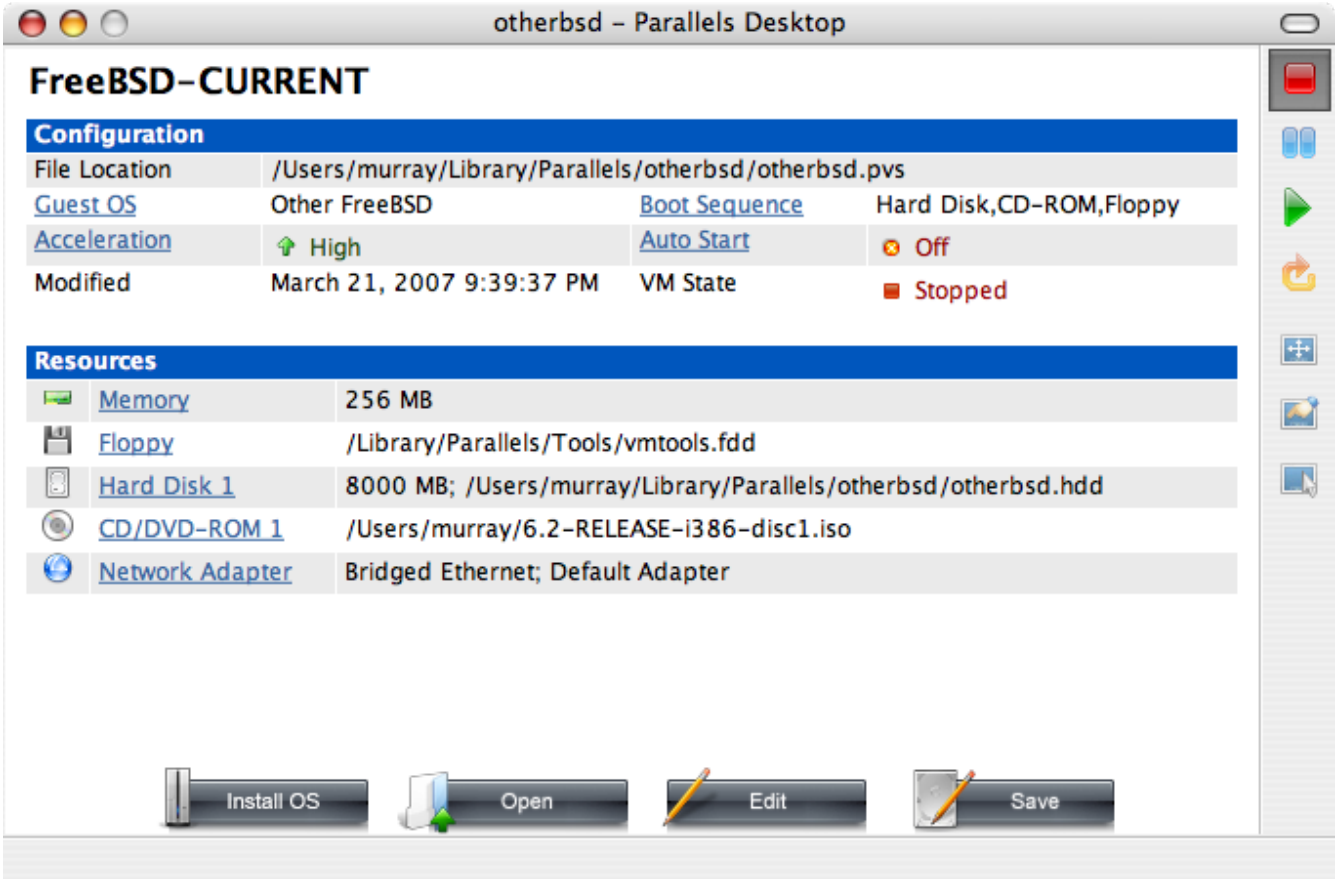
Válasszuk ki a hálózatkezelés típusát és a hálózati csatolót.


Installation Assistant
Virtual Machine Configuration
FreeBSD
Welcome to Other OS
We hope you know what it is
Good luck
CPUs: 2
Memory: 256 MB
Disk space: 8 GB
Configure...
Continue

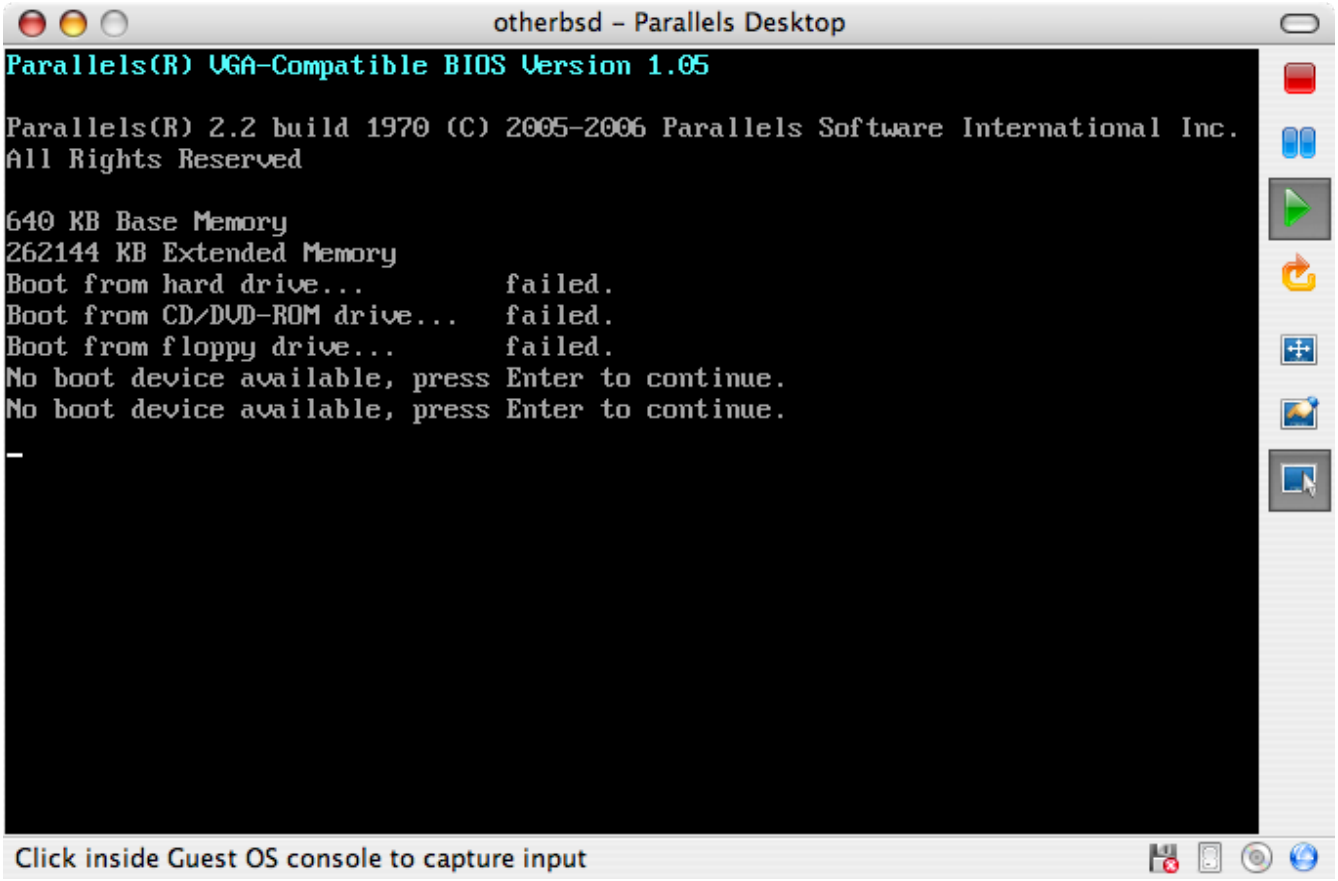
Mentsük el és fejezzük be a konfigurálást.



Miután a FreeBSD-s virtuális gépünk elkészült, telepítenünk kell rá magát az operációs rendszert is. Ezt a legegyszerûbben a hivatalosan FreeBSD telepítõ CD-rõl, vagy a hivatalos FTP oldalról letölthetõ CD-képpel tehetjük meg. Ha lemásoltuk a megfelelõ CD-képet a Mac® helyi állományrendszerére, vagy behelyeztük a telepítõ CD-t a CD-meghajtóba, kattintsunk a FreeBSD-s Parallels ablakunk jobb alsó sarkában található lemez ikonjára. Ekkor feljön egy párbeszédablak, ahol összerendelhetjük a virtuális gépünk CD-meghajtóját egy lemezen található képpel, vagy éppen a valódi CD-meghajtónkkal.

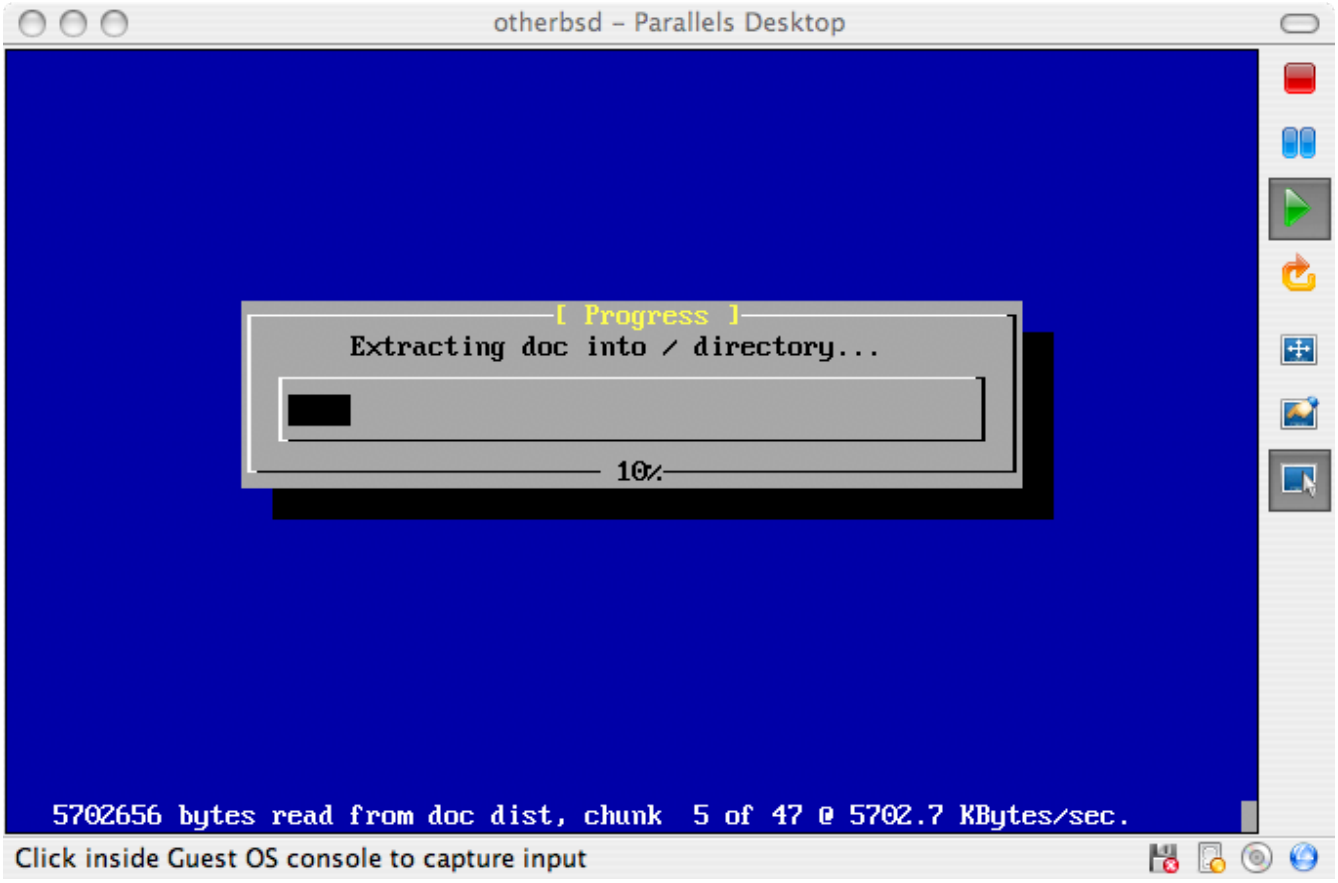


Ahogy megtettük az imént említett összerendelést, indítsuk is újra a FreeBSD-s virtuális gépünket a megszokott módon, az újraindítás ikonjára kattintva.

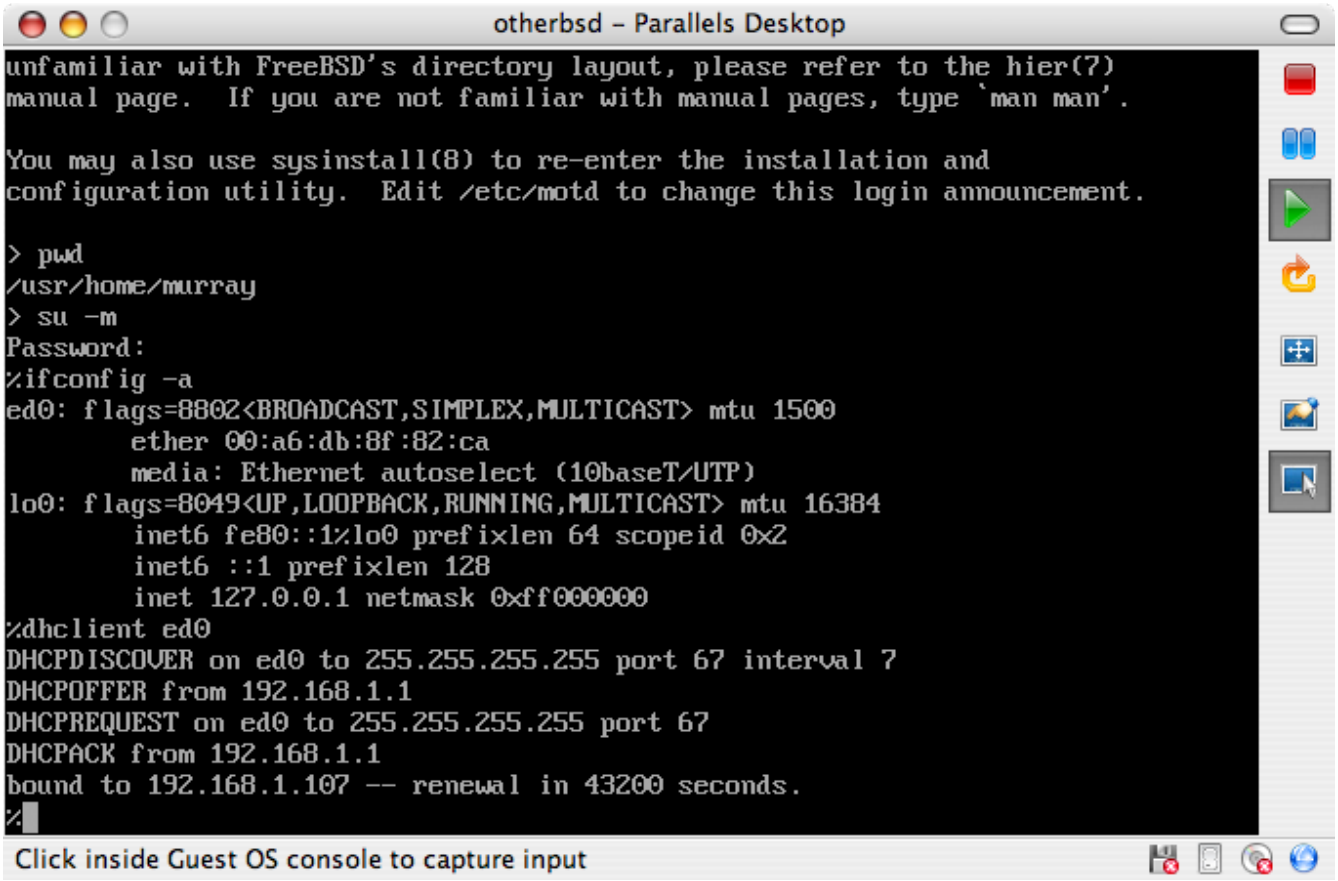


Ekkor a rendszer megtalálja a FreeBSD telepítõlemezt és a sysinstall segítségével megkezdi a telepítést a A FreeBSD telepítéseben leírtak szerint. Ha szükségünk van rá, telepíthetjük az X11-et is,

de egyelõre még ne próbáljuk beállítani.



A telepítés befejezését követõen indítsuk újra a frissen telepített FreeBSD-s virtuális gépünket.



A FreeBSD beállítása Mac OS® X/Parallelsen

Miután telepítettük a FreeBSD-t Mac OS® X/Parallels-re, még vár ránk néhány konfigurációs lépés a rendszer virtuálizált mûködésének optimalizálása érdekében.

1. A rendszerbetöltõ változóinak beállítása

   A legfontosabb lépés a `kern.hz` változó értékének csökkentése, amivel így a FreeBSD processzor-kihasználtságát is csökkentjük a Parallels alatt. Ezt a következõ sor hozzadásával tehetjük meg a /boot/loader.conf állományban:

   ```
   kern.hz=100
   ```

   Enélkül egy üresjáratban levõ FreeBSD Parallels-vendég az iMac® egy processzorának durván 15%-át foglalja le. A változtatás életbe léptetése után azonban ez megközelítõen 5%-ra redukálható.

2. Egy új konfigurációs állomány létrehozása a rendszermaghoz

   Nyugodtan eltávolíthatjuk az összes SCSI, FireWire és USB eszközmeghajtót. A Parallels által felkínált virtuális hálózati csatolót az ed(4) meghajtón keresztül tudjuk elérni, ezért az ed(4) és miibus(4) meghajtókon kívül az összes többi elhagyható.

3. A hálózati kapcsolat beállítása

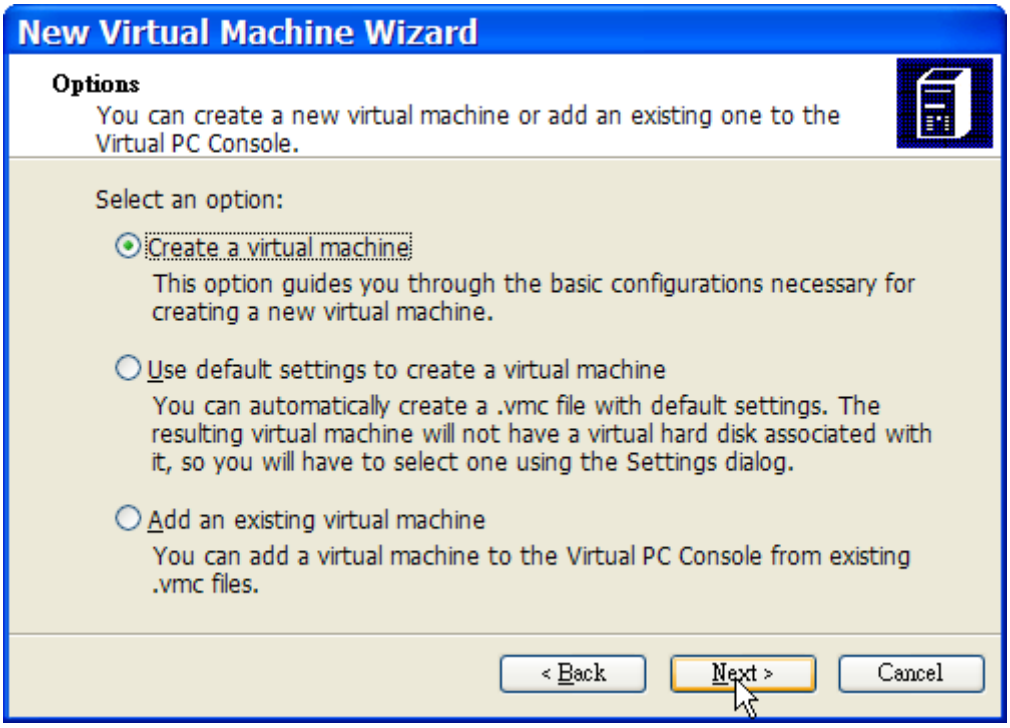
   Az alapvetõ hálózati beállítás a virtuális gépünkön a DHCP aktiválása, aminek segítségével csatlakozni tudunk arra a helyi hálózatra, amelyen maga a gazda Mac® is megtalálható. Ezt az alábbi sor felvételével tudjuk megoldani az /etc/rc.conf állományba: `ifconfig_ed0="DHCP"`. Bõvebb információkért járuljunk a Egyéb haladó hálózati témák fejezethez.

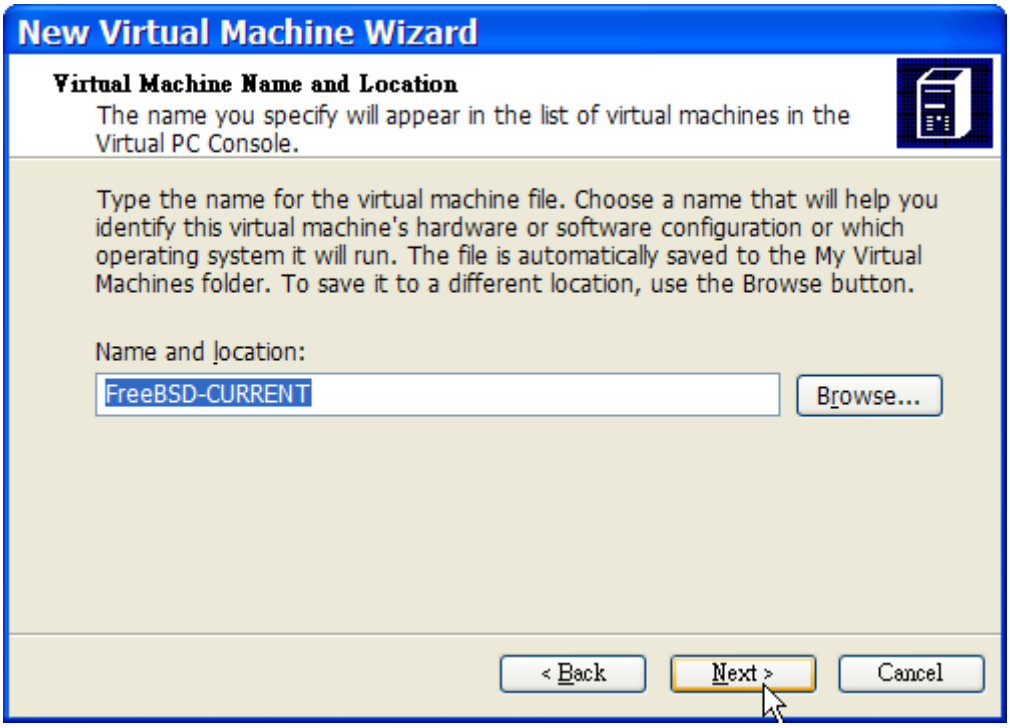
22.2.2. Virtual PC-vel Windows®-on

A Windows®-ra fejlesztett Virtual PC a Microsoft® egyik szabadon letölthetõ szoftverterméke. A rendszerkövetelményeit bõvebben lásd a http://www.microsoft.com/windows/downloads/virtualpc/sysreq.mspx linken. Miután telepítettük a Microsoft® Windows®-ra a Virtual PC alkalmazást, be kell állítanunk egy virtuális gépet, majd telepítenünk kell rá a kívánt vendég operációs rendszert.

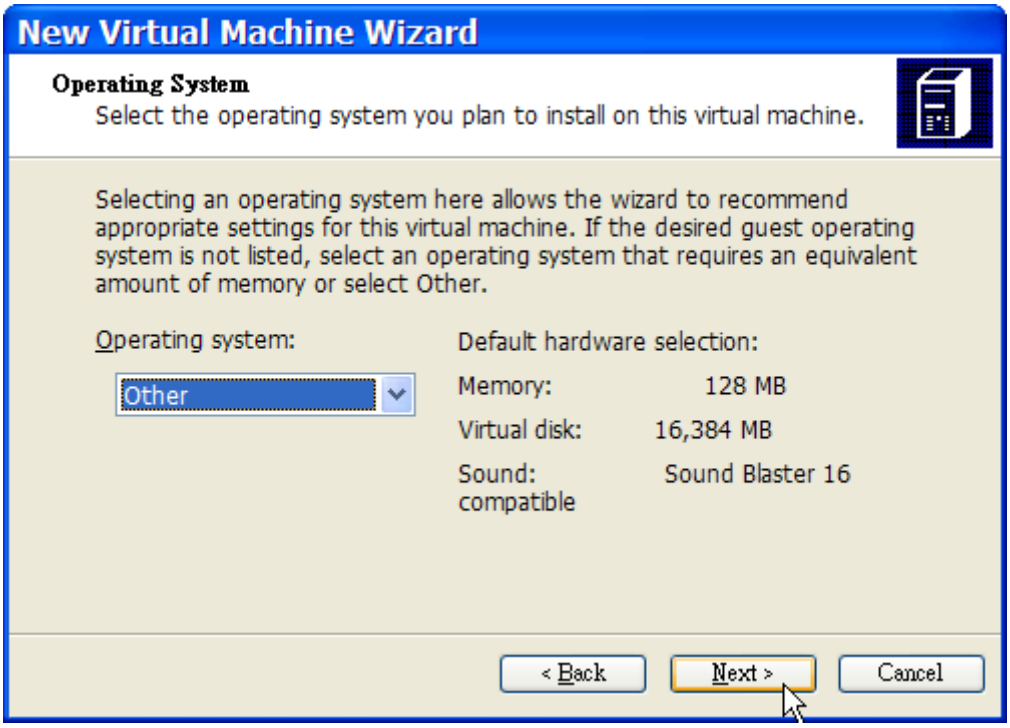
A FreeBSD telepítése Virtual PC/Microsoft® Windows®-ra

Amikor a FreeBSD-t a Microsoft® Windows® és Virtual PC párosra akarjuk telepíteni, akkor kezdjünk egy egy új virtuális gép létrehozásával. Ehhez válasszuk ki a menübõl a Create a virtual machine (Virtuális gép létrehozása) pontot.

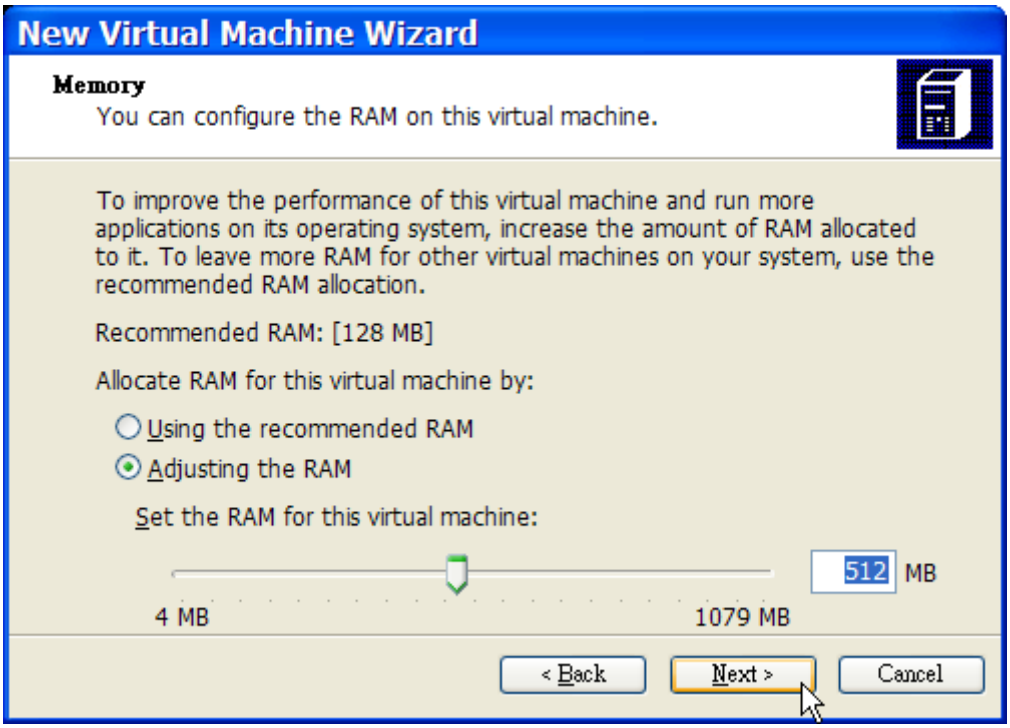


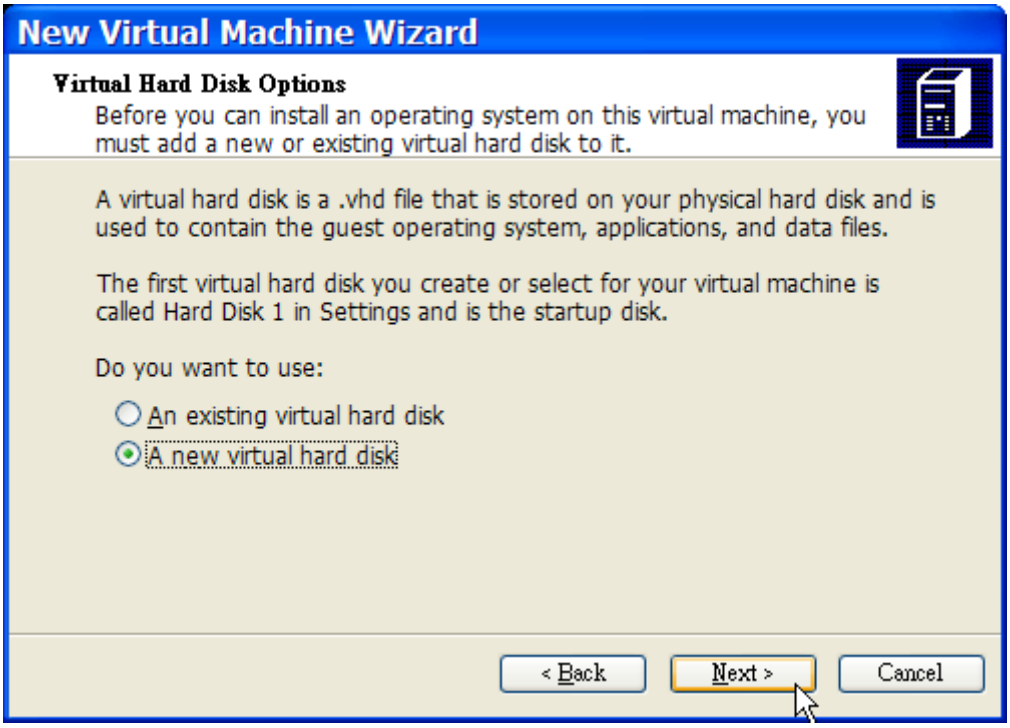


Majd válasszuk az Operating system (Operációs rendszer) beállításánál az Other (Egyéb) opciót.

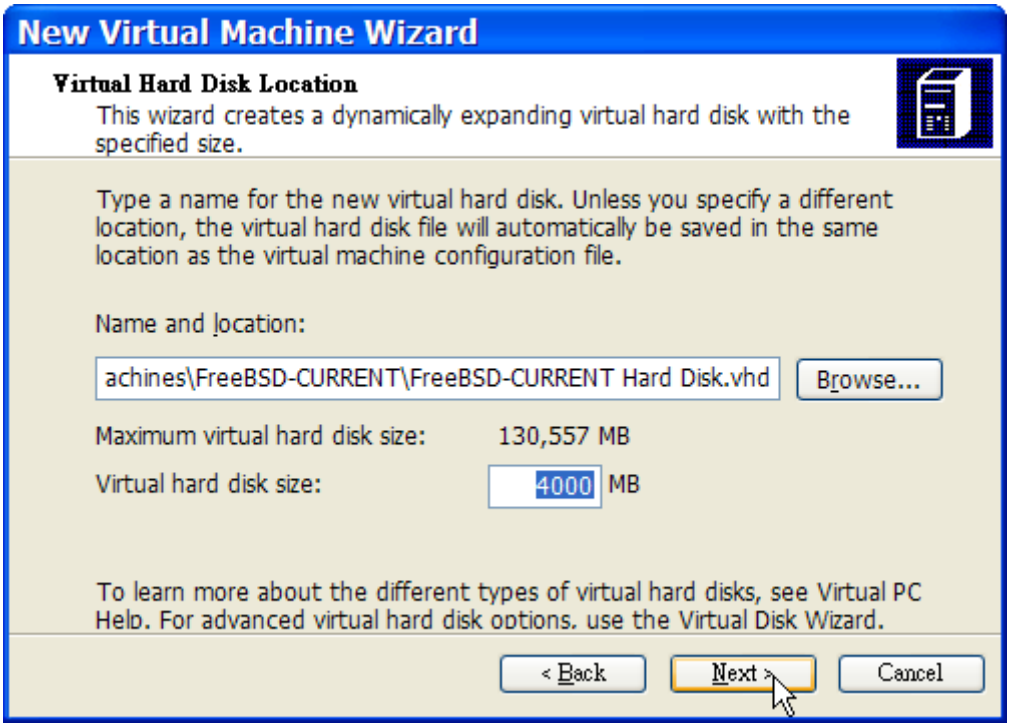


Ezután válasszuk ki a szándékainknak megfelelõen a telepítendõ FreeBSD példányhoz mért memória és lemezterület mennyiségét. Ahhoz, hogy a FreeBSD fusson Virtual PC alatt, 4 GB-nyi lemezterület és 512 MB RAM beállítása a legtöbb esetben kiválóan megfelelõ.

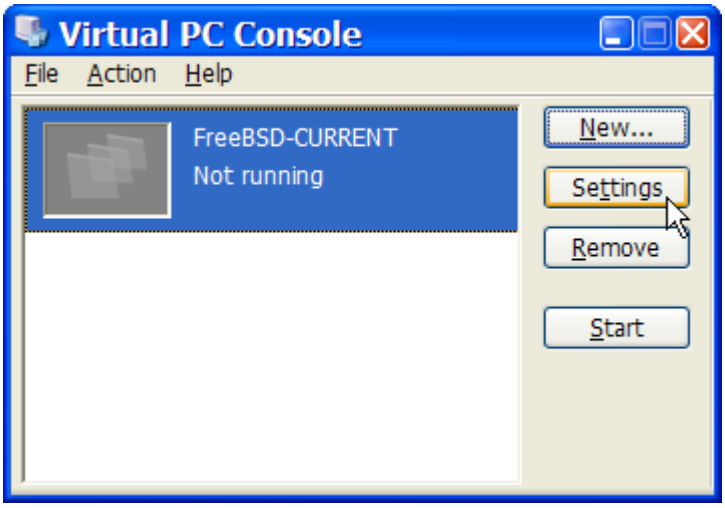


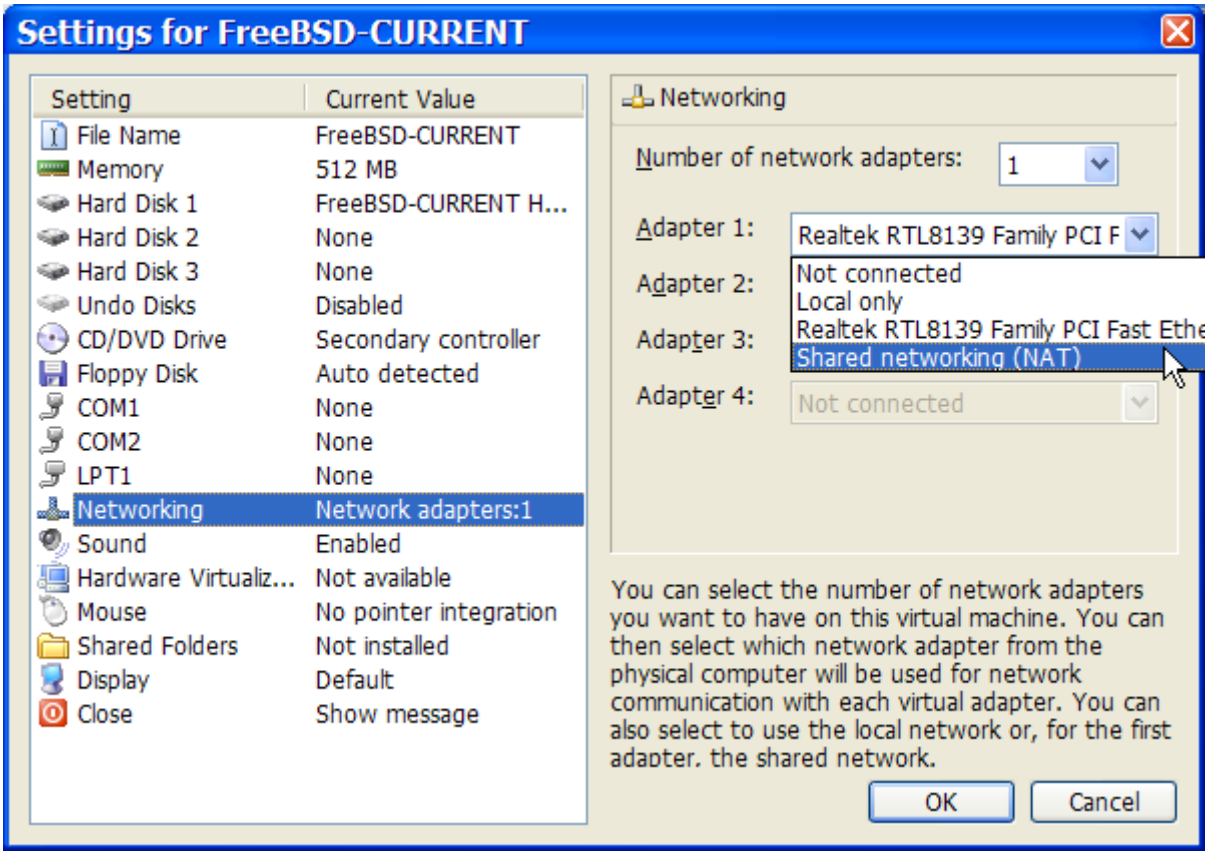


Mentsük el és fejezzük be a konfigurációt.

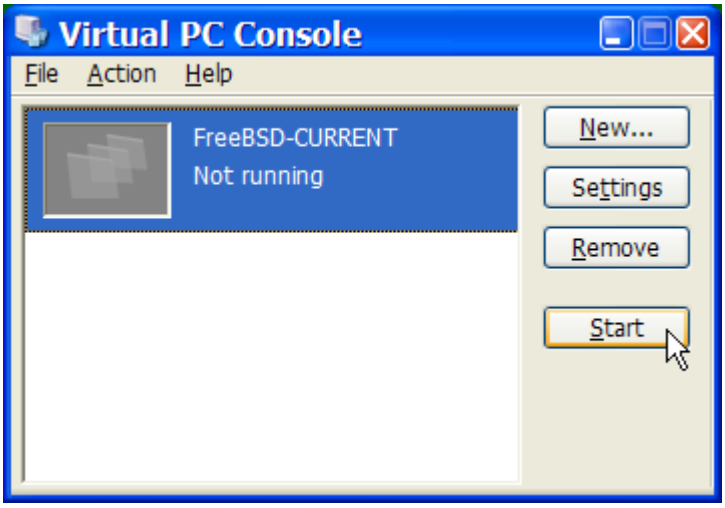


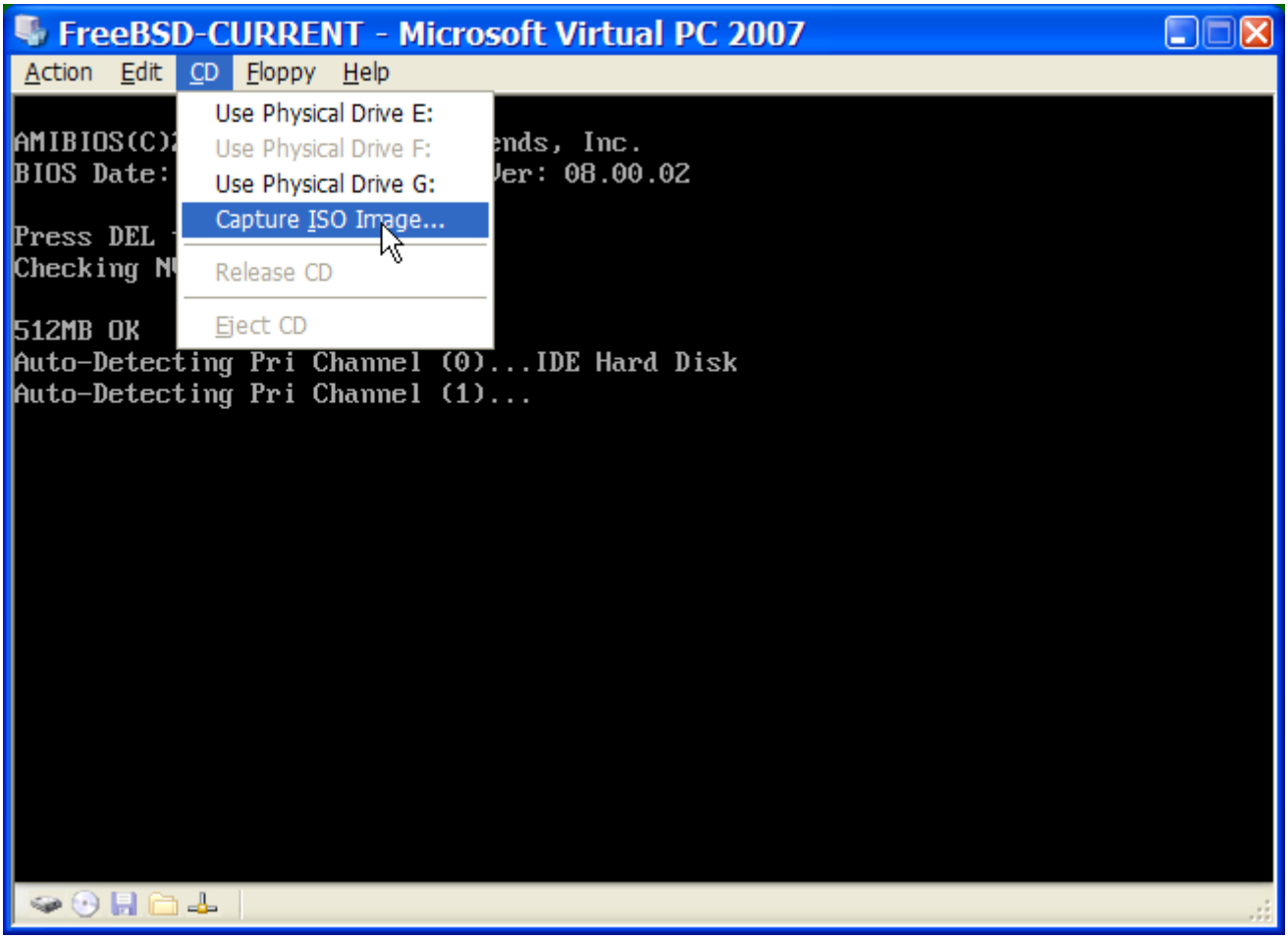
Válasszuk ki a FreeBSD-s virtuális gépünket, majd kattintsunk a **Settings** (Beállítások) menüre és állítsuk be hálózati csatoló és hálózatkezelés típusát.



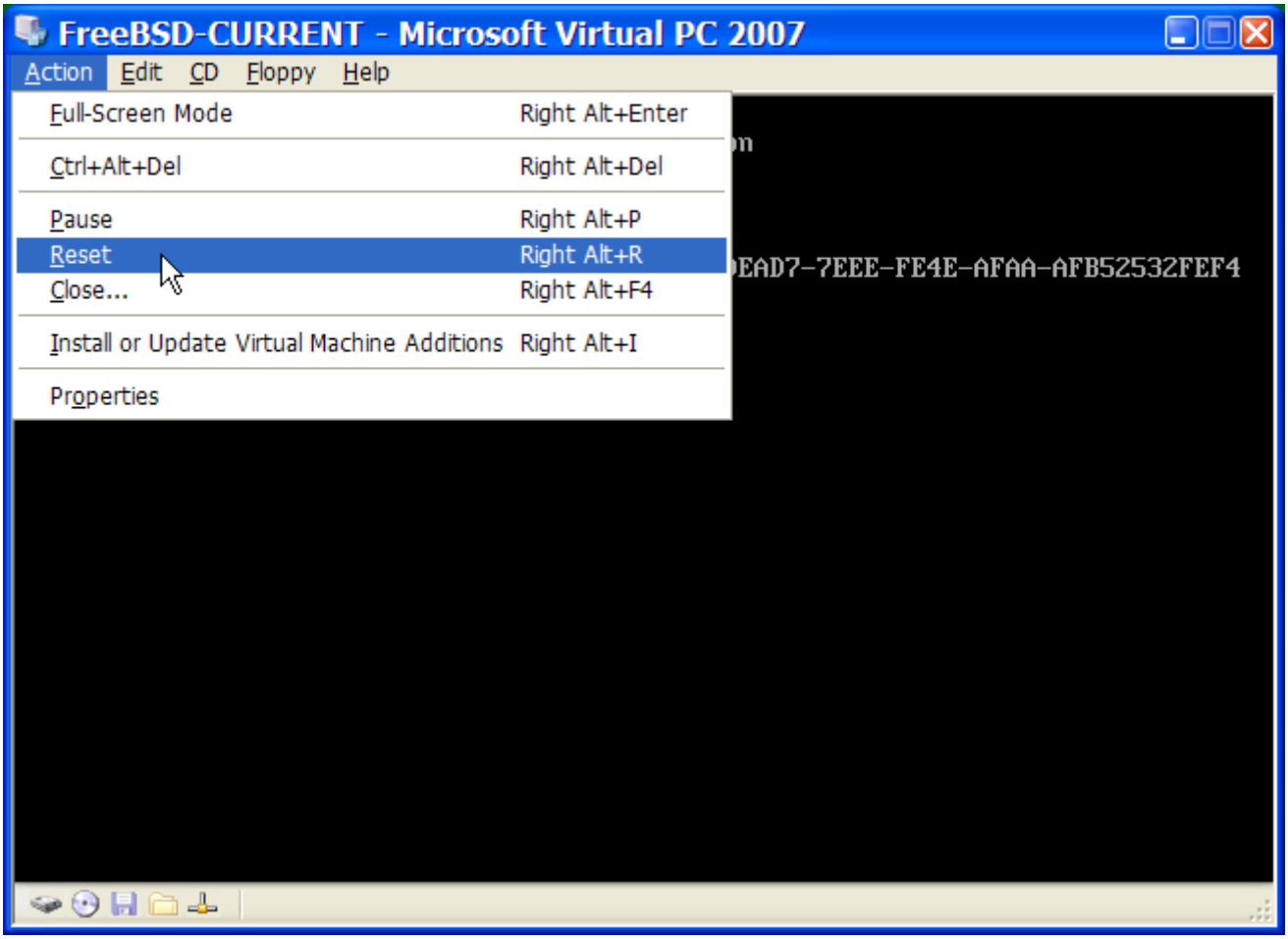


A FreeBSD-nek otthont adó virtuális gépünk létrehozása után telepítenünk is kell rá a rendszert. Ez legegyszerûbben a hivatalos FreeBSD telepítõ CD-vel vagy a hivatalos FTP oldalról letölthetõ CD-képpel tehetjük meg. Amikor letöltöttük a megfelelõ CD-képet a helyi Windows®-os állományrendszerünkre vagy behelyeztük a telepítéshez használható CD-t a CD-meghajtónkba, a FreeBSD-s virtuális gépünk elindításához kattintsunk rá duplán. Ezt követõen a Virtual PC ablakában kattintsunk a **CD** menüre és válasszuk ki belõle a **Capture ISO Image...** (Lemezkép használata...) pontot. Ennek hatására megjelenik egy ablak, amiben a virtuális gépünk CD-meghajtóihoz tudunk csatlakoztatni lemezképeket vagy akár létezõ CD-meghajtókat.

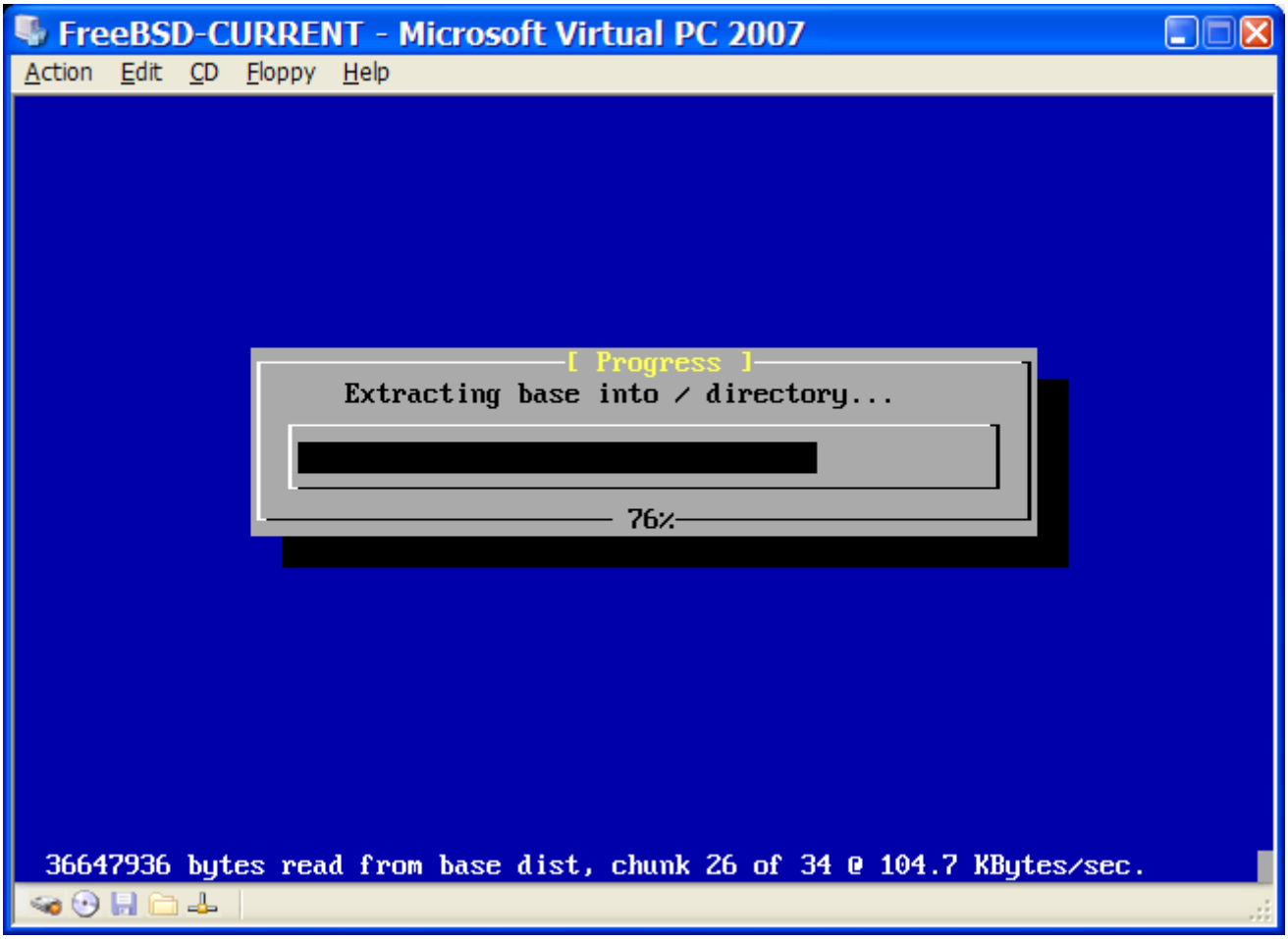




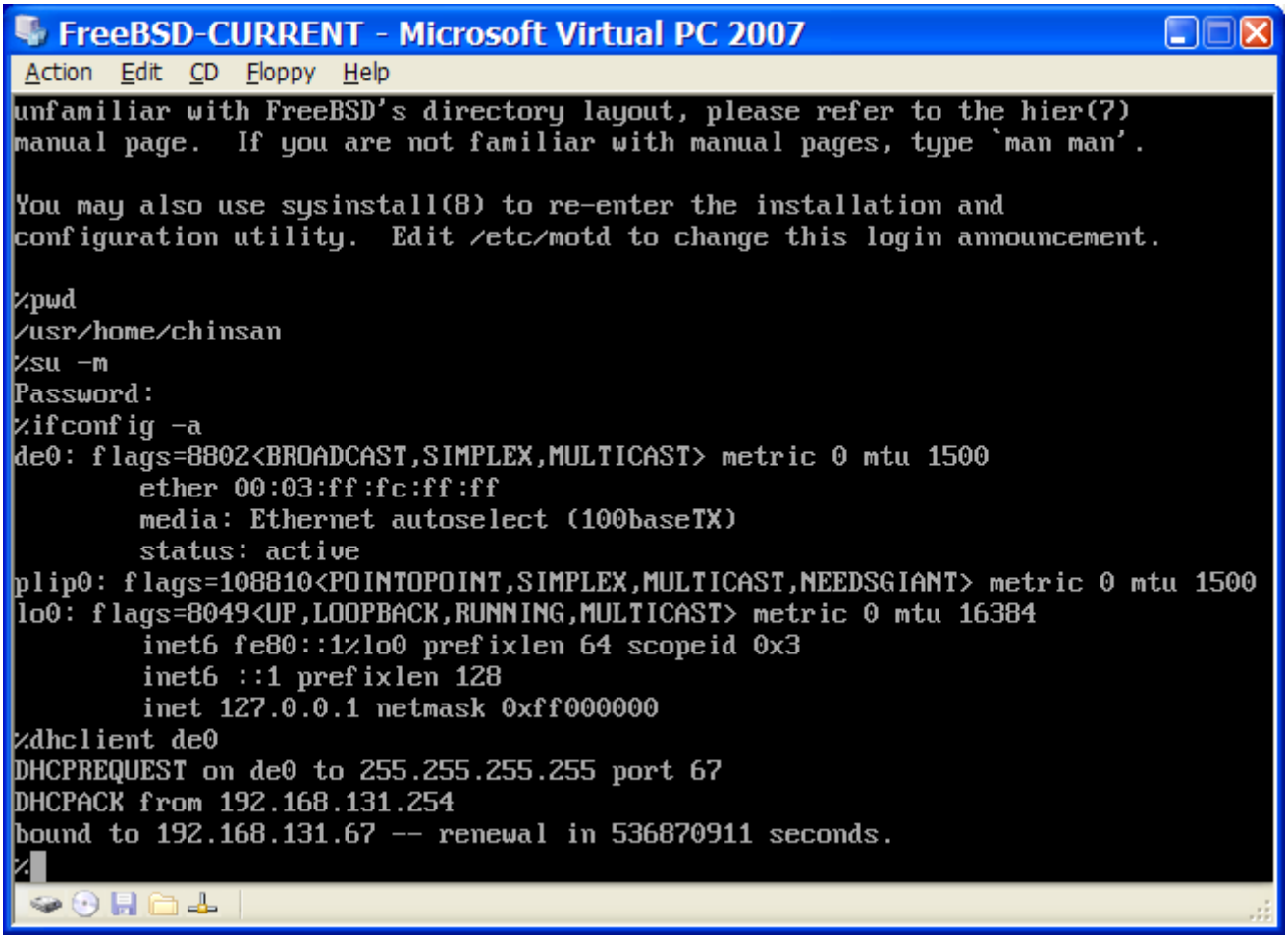
Miután sikeresen beállítottuk a telepítõ CD forrását, indítsuk újra a virtuális gépet az **Action** (Mûvelet) menün belül a **Reset** (Újraindítás) pont kiválasztásával. Így a Virtual PC újraindítja a virtuális rendszert egy olyan speciális BIOS használatával, amely a normális BIOS-hoz hasonlóan elõször megkeresi az elérhetõ CD-meghajtókat.



Ebben az esetben a FreeBSD telepítõeszközét fogja megtalálni és megkezdi a A FreeBSD telepítéseben ismertetett szokásos, sysinstall programra alapuló telepítési eljárást. Ennek során az X11-et is feltelepíthetjük, habár egyelõre még ne állítsuk be.



Ne felejtsük el kivenni a meghajtóból a telepítéshez használt CD-t vagy elengedni a megfelelõ lemezképet, amikor befejezõdõtt a telepítés. Végezetül indítsuk ismét újra a frissen telepített FreeBSD-s virtuális gépünket.



A FreeBSD beállítása a Microsoft® Windows®/Virtual PC-n

Miután a FreeBSD-t minden gond nélkül telepítettük a Microsoft® Windows®-on futó Virtual PC-re, még további beállítási lépéseket is meg kell tennünk a rendszer virtualizált mûködésének finomhangolásához.

1. A rendszertöltõ változóinak beállítása

   A legfontosabb teendõnk csökkenteni a `kern.hz` konfigurációs beállítás értéket, aminek köszönhetõen vissza tudjuk fogni a Virtual PC alatt futó FreeBSD processzorhasználatát. Ezt úgy tudjuk megtenni, ha a /boot/loader.conf állományba felvesszük a következõ sort:

   ```
   kern.hz=100
   ```

   Enélkül a Virtual PC alatt üresjáratban futó FreeBSD vendég operációs rendszer egy egyprocesszoros számítógép idejének durván 40%-át foglalja le. A változtatás után azonban ez az érték pusztán közel 3%-ra csökken le.

2. Új konfigurációs állomány létrehozása a rendszermaghoz

   Nyugodtan eltávolíthatjuk a SCSI, FireWire és USB eszközmeghajtókat. A Virtual PC által felajánlott virtuális hálózati csatolót a de(4) meghajtón keresztül tudjuk használni, ezért a de(4) és miibus(4) eszközön kívül az összes többi hálózati eszköz támogatása kiszedhetõ a rendszermagból.

3. A hálózati kapcsolat beállítása

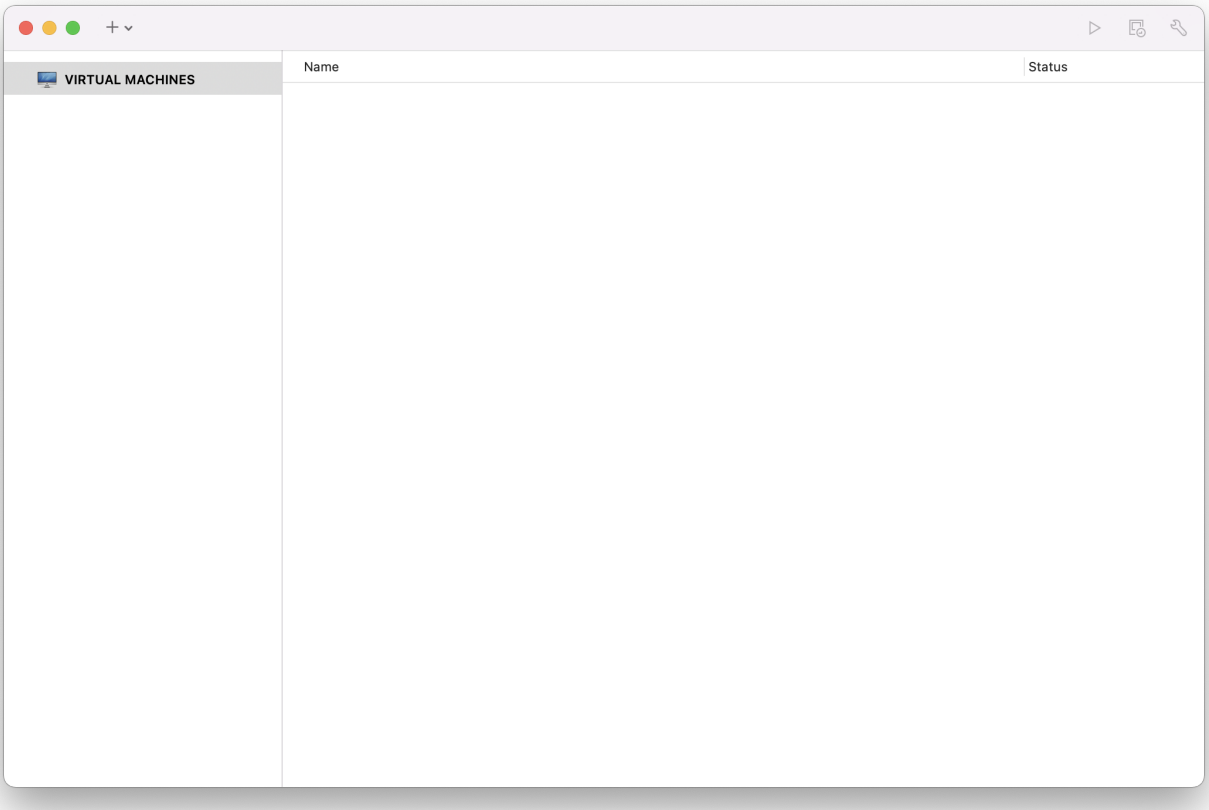
   A legalapvetõbb hálózati beállítás csupán annyiból áll, hogy DHCP-n keresztül csatlakoztatjuk a virtuális gépünket ugyanahhoz a helyi hálózathoz, amiben a gazda Microsoft® Windows®-os gépünk is megtalálható. Ezt úgy tudjuk elérni, ha a /etc/rc.conf állományba megadjuk a `ifconfig_de0="DHCP"` sort. A komolyabb hálózati beállításokat a Egyéb haladó hálózati témákben találhatjuk.

22.2.3. VMWare-rel MacOS-en

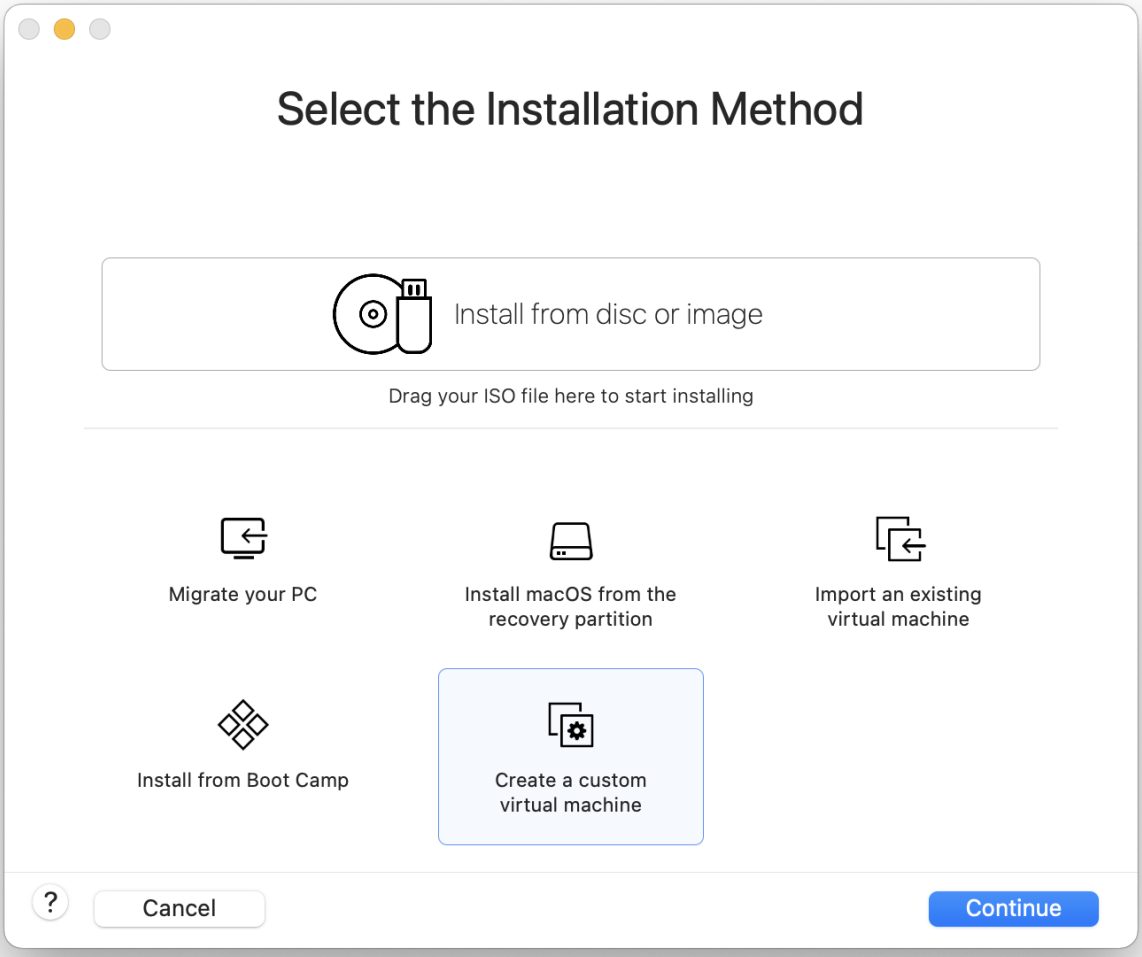
A Mac®-ek számára fejlesztett VMWare Fusion egy olyan kereskedelmi termék, amit az Intel® alapú Apple® Mac® gépekre tudunk telepíteni a Mac OS® 10.4.9 és késõbbi változatain. A FreeBSD itt egy teljesen támogatott vendég operációs rendszer. Miután a VMWare Fusion felkerült a Mac OS® X rendszerünkre, be kell állítanunk a virtuális gépet és telepítenünk rá a vendég operációs rendszert.

A FreeBSD telepítése a Mac OS® X/VMWare-re

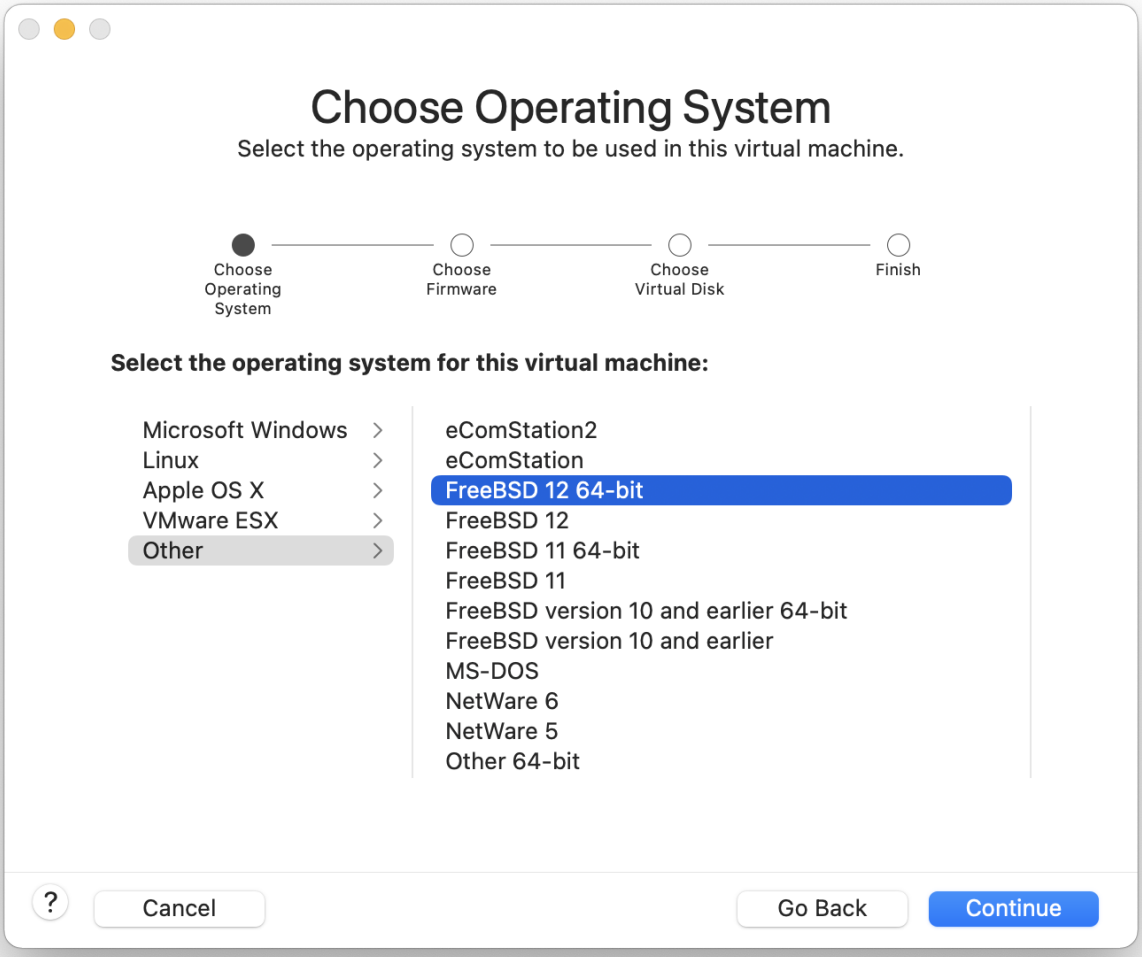
Elõször indítsuk el a VMWare Fusion-t, aminek eredményeképpen betöltõdik a Virtual Machine Library. Egy új virtuális gépre létrehozásához kattintsunk a "New" gombra:



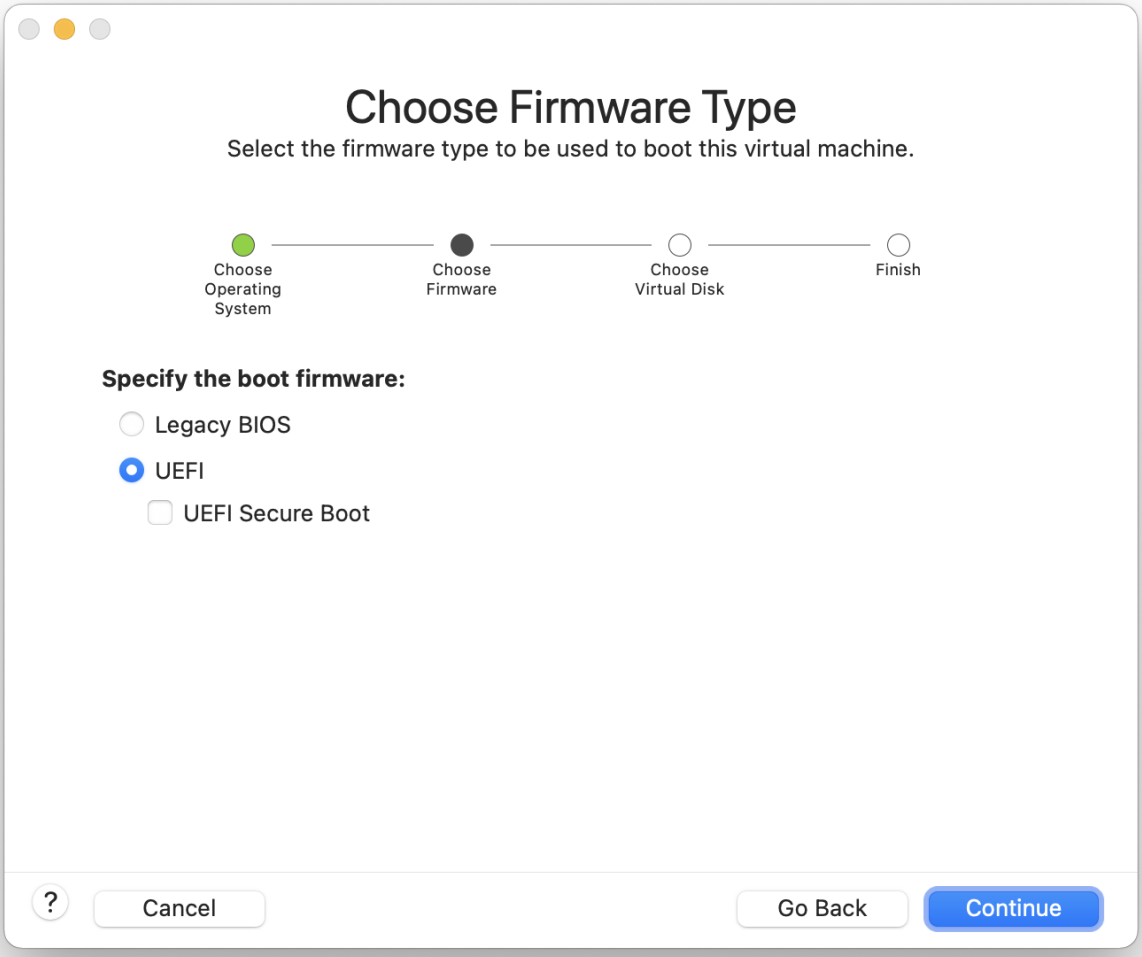
Ekkor bejön az új gép összeállítását segítõ New Virtual Machine Assistant, ahol a továbblépéshez kattintsunk a Continue gombra:



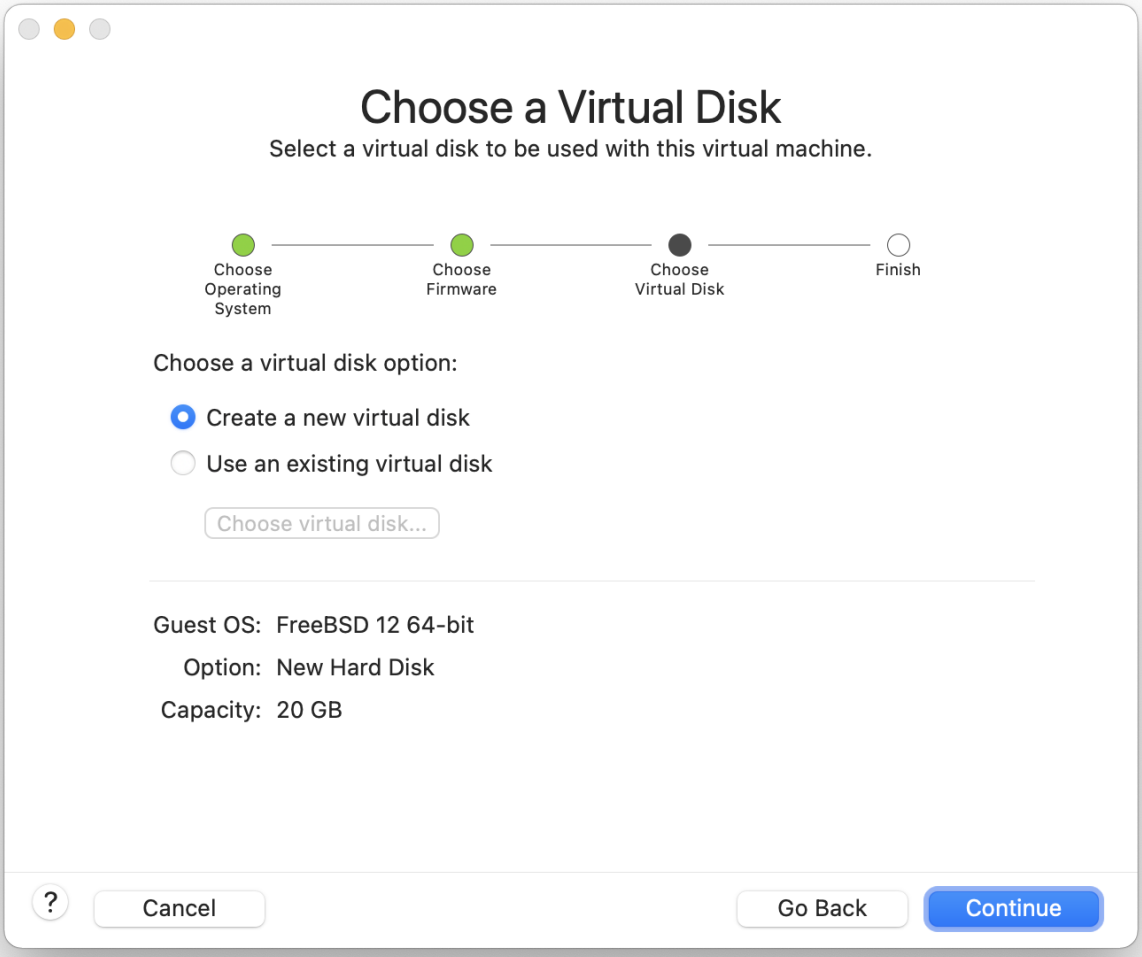
Az operációs rendszerek (Operating System) közül válasszuk az "egyéb" (Other) kategóriát, majd a Version fülön a FreeBSD vagy a FreeBSD 64-bit változatot attól függően, hogy 32 bites vagy 64 bites támogatásra van szükségünk:



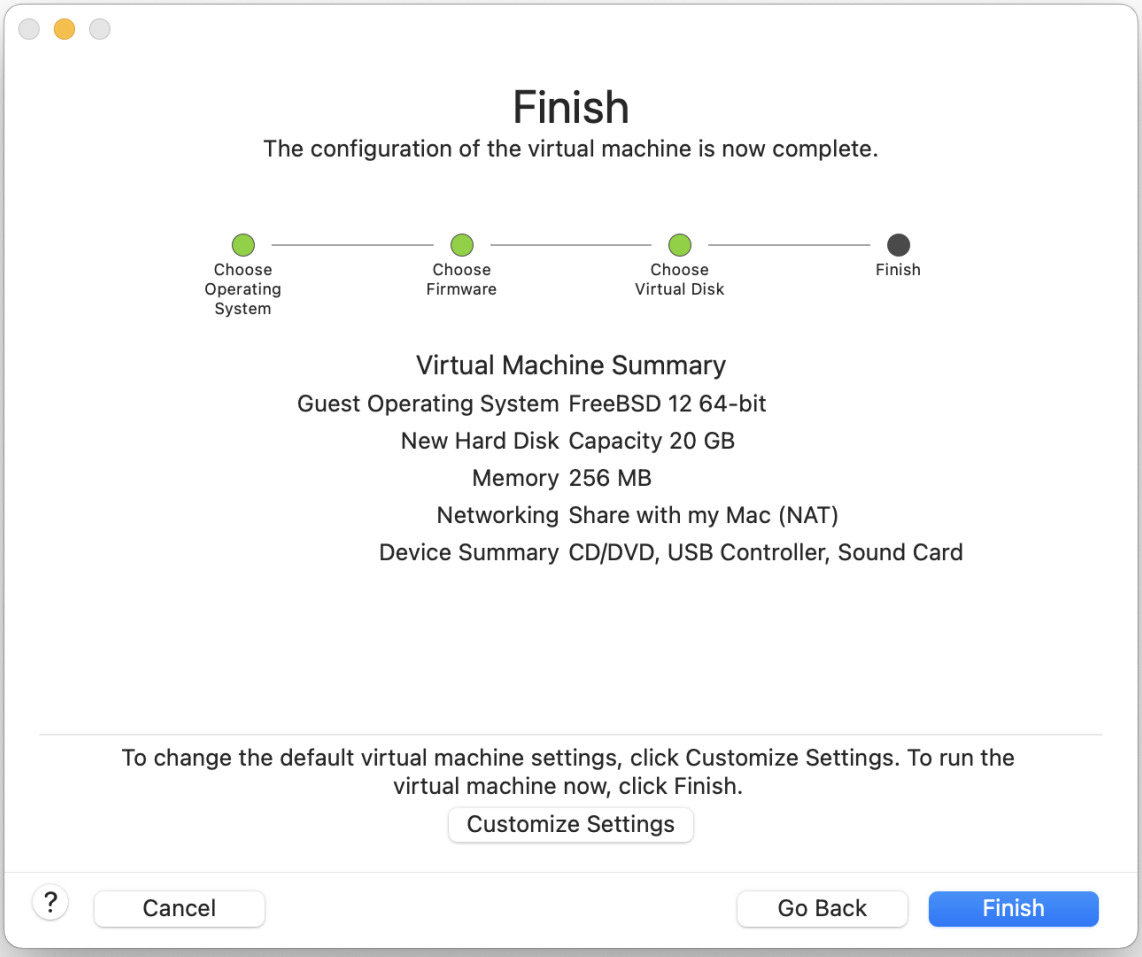
Adjuk meg a virtuális gép képének nevét és a könyvtárat, ahova el akarjuk menteni:



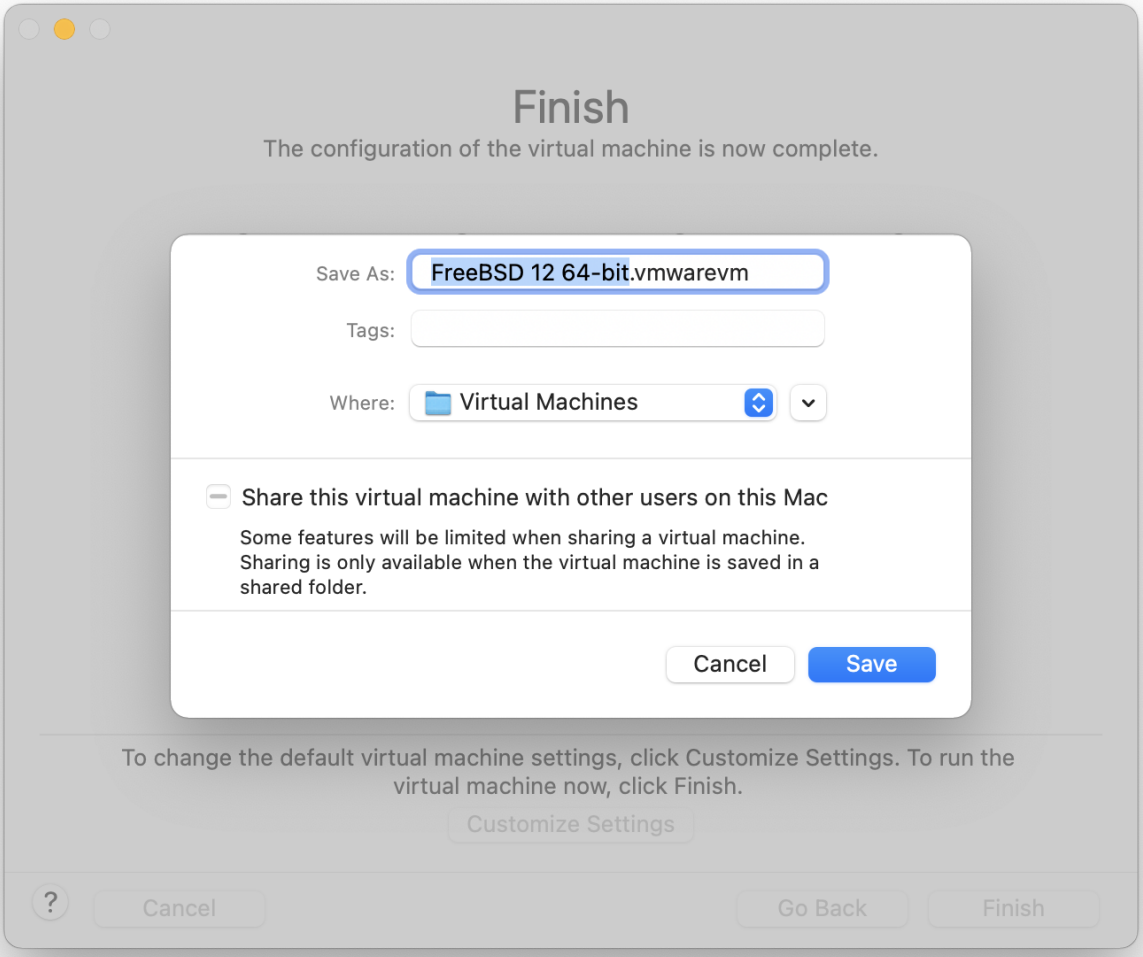
Válasszuk meg a virtuális géphez tartozó virtuális merevlemez méretét is:



Mondjuk meg, hogy milyen módon szeretnénk telepíteni a virtuális gépre, ISO formátumú lemezképről vagy CD-ről:



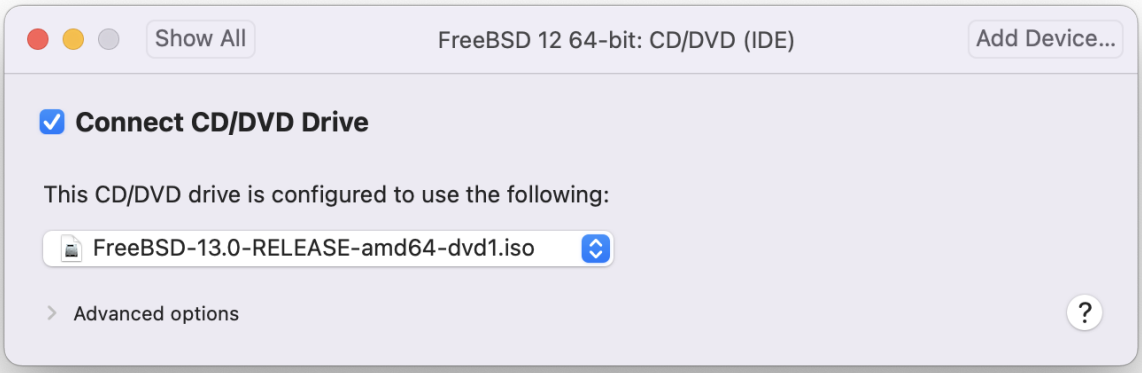
Ahogy a Finish feliratú gombra kattintunk, a virtuális gép máris elindul:



Telepítsük fel a FreeBSD-t a megszokott módon vagy a A FreeBSD telepítése utasításai mentén:

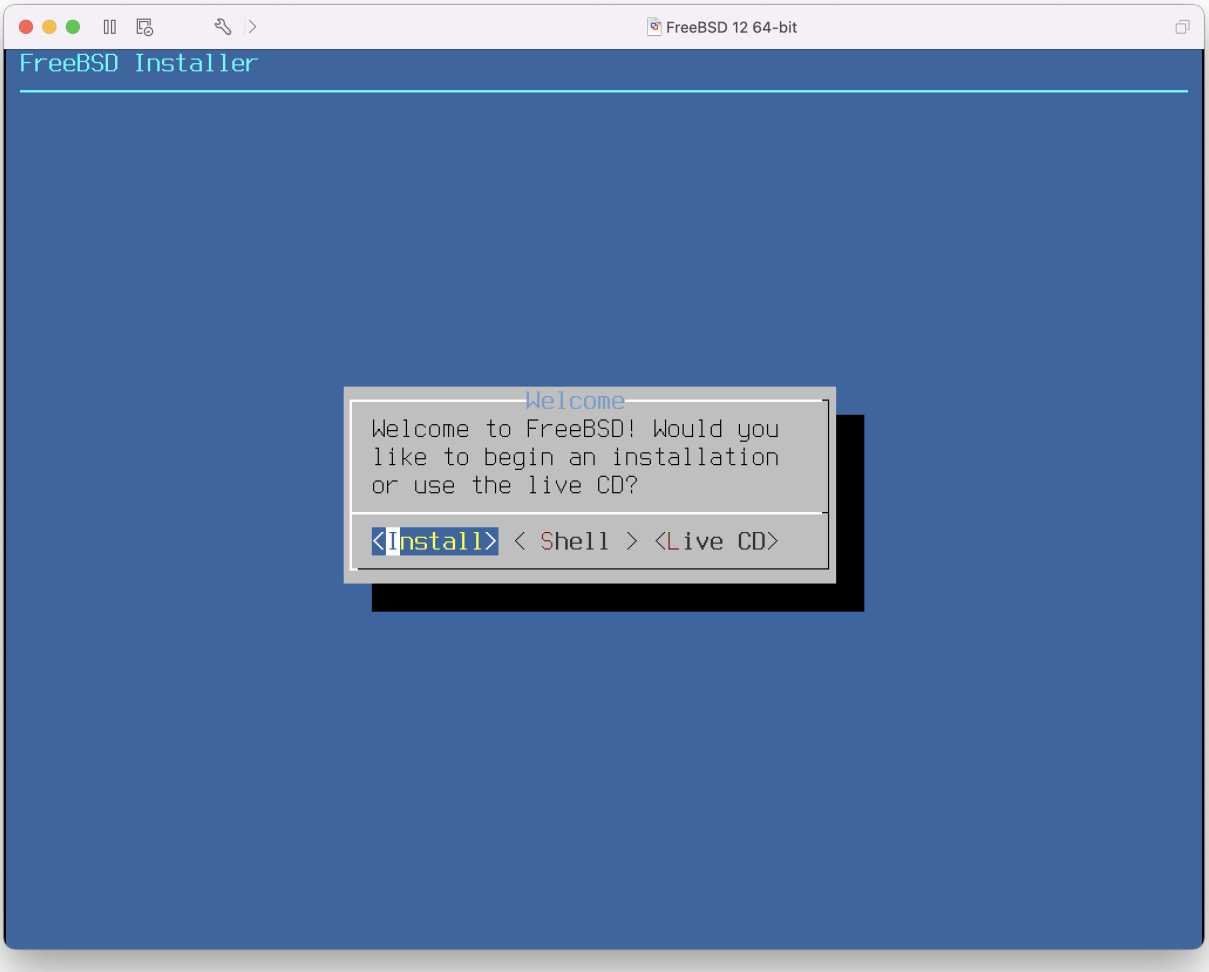
Miután befejezõdött a telepítés, módosítsuk a virtuális gép beállításait, például a memória mennyiségét:

A virtuális gép hardveres beállításai a futása alatt nem változtathatóak meg.

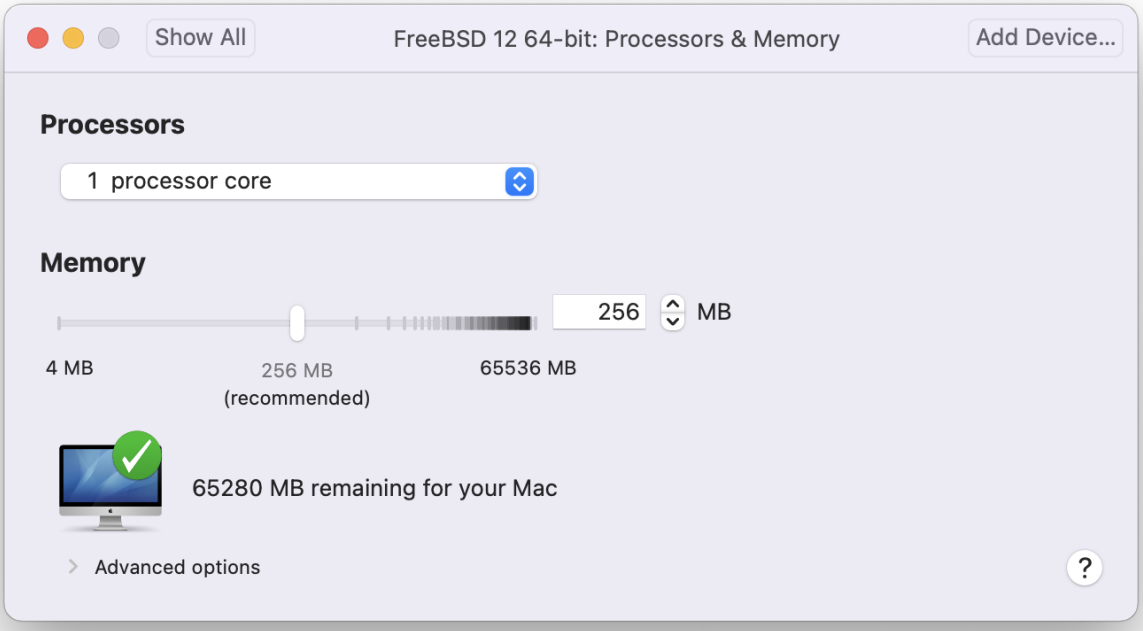


A virtuális gép által használható processzorok számát:

A CD-meghajtó állapotát. Általában lehetõségünk van a virtuális gépet leválasztani a CD-meghajtóról vagy ISO lemezképrõl, ha már nem használjuk.



A hálózati csatlakozás a virtuális géppel kapcsolatban utolsóként beállítandó tényezõ. Ha a befogadó gépen kívül még más gépeket is el akarunk érni a virtuális géprõl, akkor ehhez mindenképpen a Connect directly to the physical network (Bridged) opciót válasszuk. Minden más esetben a Share the host's internet connection (NAT) az ajánlott, mivel így a virtuális gép eléri az internetet, de a hálózatról nem lehet azt elérni.



Miután befejeztük a beállítások finomhangolását, indítsuk is el a frissen telepített FreeBSD-s virtuális gépünket.

A FreeBSD beállítása a Mac OS® X/VMWare-en

Ahogy a FreeBSD-t sikeresen telepítettük a Mac OS® X alatt futó VMWare-re, néhány konfigurációs lépést még meg kell tennünk a virtualizált rendszer teljesítmények optimalizálása érdekében.

1. A rendszertöltõ változóinak beállítása

   A legfontosabb lépés talán a `kern.hz` változó értékének csökkentése, amivel a VMWare alatt futó FreeBSD processzorhasználatát szoríthatjuk vissza. Ezt a következõ sor hozzáadásával érhetjük el a /boot/loader.conf állományban:

   ```
   kern.hz=100
   ```

   Enélkül az üresjáratban zakatoló FreeBSD-s VMWare vendég nagyjából az iMac® egyik processzorának 15%-át emészti fel. Ezzel a módosítással azonban ez lenyomható közel 5%-ra.

2. Új konfigurációs állomány létrehozása a rendszermaghoz

   Nyugodtan törölhetjük az összes FireWire és USB eszköz meghajtóját. A VMWare egy em(4) meghajtón keresztül elérhetõ virtuális hálózati kártyát biztosít, így az em(4) kivételével az összes hálózati eszköz meghajtóját kivehetjük a rendszermagból.

3. A hálózat beállítása

A legegyszerûbb hálózati beállítás mindösszesen a DHCP használatát igényli, aminek révén a virtuális gépünk a befogadó Mac®-kel egy helyi hálózatra kerül. Ezt úgy tudjuk engedélyezni, ha az /etc/rc.conf állományba felvesszük az `ifconfig_em0="DHCP"` sort. Ha ennél komolyabb hálózati beállítások is érdekelnek minket, akkor olvassuk el a Egyéb haladó hálózati témákat.

22.3. A FreeBSD mint gazda

Gazda operációs rendszerként a FreeBSD évekig nem kapott hivatalosan támogatást egyetlen elterjedtebb virtualizációs megoldás részérõl sem. Sokan erre a célra eddig a VMWare korábbi és inkább már elavult, a Linux® kompatibilitási rétegre épülõ változatait (mint például emulators/vmware3) használták. Nem sokkal azonban a FreeBSD 7.2 megjelenése után a Sun VirtualBox™OSE (Open Source Edition) natív FreeBSD alkalmazásként bukkant fel a Portgyûjteményben.

A VirtualBox™ egy folyamatos fejlesztés alatt álló, komplett virtualizációs csomag, amely immáron elérhetõ a legtöbb népszerû operációs rendszerre, mint a Windows®, Mac OS®, Linux® és a FreeBSD. Egyaránt képes Windows® és UNIX® fajtájú vendégrendszerek futattására. Nyílt- és zárt forráskódú változatban is elérhetõ. A felhasználók szempontjából a kettõ közti talán legfontosabb eltérés, hogy a nyílt forráskódú változat nem tartalmaz USB támogatást. A különbségek teljes listája megtalálható a VirtualBox™ wiki "Editions" oldalán, a http://www.virtualbox.org/wiki/Editions címen. FreeBSD alatt jelenleg csak a nyílt forráskódú változat érhetõ el.

22.3.1. A VirtualBox™ telepítése

A VirtualBox™ a emulators/virtualbox-ose könyvtárból érhetõ el portként, és onnan a következõ parancsokkal telepíthetõ:

```
# cd /usr/ports/emulators/virtualbox-ose
# make install clean
```

A beállítások közt az egyik leghasznosabb a `GuestAdditions` nevû programcsomag telepítése. A benne található programokon keresztül a vendégként futó operációs rendszer számos hasznos szolgáltatását el tudjuk érni, úgy mint az egérmutató integrációját (ekkor az egérkurzor zökkenõmentesen használható a gazda és a vendég rendszerben is) vagy a videomemória gyorsabb elérését (különösen Windows® esetében). A vendégekhez telepíthetõ ilyen jellegû kiegészítések az adott rendszer telepítése után a **Devices** menübõl érhetõek el.

A VirtualBox™ elsõ indítása elõtt el kell még végeznünk néhány további beállítást. Fontos tudnunk, hogy a port a telepítés során a /boot/modules könyvtárba tesz még egy rendszermagmodult is, amelyet még külön be kell töltenünk:

```
# kldload vboxdrv
```

Ehhez még vegyük fel a következõ sort a /boot/loader.conf állományba, így a modul a rendszer

minden egyes indításakor magától betöltõdik:

```
vboxdrv_load="YES"
```

A VirtualBox™ 3.1.2 elõtti változatai ezenkívül még igénylik a proc állományrendszer csatlakoztatását is. Az újabb változatokban erre már nincs szükség, mivel ezekben helyette már a sysctl(3) könyvtár függvényeit használják.

Ha viszont a port valamelyik korábbi változatát használjuk, akkor kövessük a lentebb szereplõ utasításokat és csatlakoztassuk a proc állományrendszert:

```
# mount -t procfs proc /proc
```

Ha hozzáadjuk az alábbi sort a /etc/fstab állományhoz, akkor ez a beállítás is megmarad a rendszer újraindítása után:

```
proc     /proc   procfs  rw        0       0
```

Nagyon valószínû, hogy proc állományrendszerrel van gondunk, amikor a következõ hibaüzenetet kapjuk a VirtualBox™ indításakor:

```
VirtualBox: supR3HardenedExecDir: couldn't read "", errno=2 cchLink=-1
```

Ilyenkor a `mount` parancs kiadásával ellenõrizzük az állományrendszer sikeres csatlakoztatását.

A VirtualBox™ telepítése során keletkezik még egy `vboxusers` nevû csoport. Ide azokat a felhasználókat vegyük fel, akik részére szeretnénk engedélyezni a VirtualBox™ használatát. A csoportba új tagokat például a `pw` paranccsal tudunk felvenni:

```
# pw groupmod vboxusers -m felhasználónév
```

Ezek után a VirtualBox™ indításához válasszuk a grafikus környezetünk menüjében található Sun VirtualBox menüpontot, vagy egy terminálban gépeljük be ezt a parancsot:

```
% VirtualBox
```

A VirtualBox™ beállításának további lehetõségeirõl a http://www.virtualbox.org/ címen elérhetõ hivatalos holnapon olvashatunk. Tekintettel arra, hogy a FreeBSD port még viszonylag friss és folyamatos fejlesztés alatt áll, ehhez még érdemes átolvasnunk a FreeBSD wikiben szereplõ http://wiki.FreeBSD.org/VirtualBox/ oldalt is, ahol a vele kapcsolatos legfrissebb információkat és egyéb tudnivalókat találhatjuk.

Chapter 23. Honosítás - Az I18N/L10N használata és beállítása

23.1. Áttekintés

A FreeBSD felhasználói földrajzi elhelyezkedésüket tekintve mindenhol megtalálhatóak a világon. Ebben a fejezetben ismertetjük a FreeBSD honosításához és idegennyelvre fordításához alkalmazható eszközöket, amelyek segítségével az angolt nem, vagy csak kevésbé ismerõ felhasználók is képesek lesznek komolyabban használni. Az i18n megvalósítása rengeteg szemszögbõl megközelíthetõ rendszer és alkalmazás szintjén egyaránt, ezért ahol szükséges, hivatkozni fogunk az odaillõ forrásokra.

A fejezet elolvasása során megismerjük:

- milyen nyelveket és nyelvi beállításokat találhatunk napjaink operációs rendszereiben;
- hogyan használjuk a nyelvi beállításokat a saját parancsértelmezõnkben;
- hogyan állítsuk be a konzolt az angolon kívül más nyelvekhez;
- hogyan használjuk ténylegesen az X Window Systemet a különbözõ nyelvekkel;
- hol olvashatunk többet az I18N-kompatibilis alkalmazások fejlesztésérõl.

A fejezet elolvasásához ajánlott:

- külsõ alkalmazáok telepítésének ismerete (Alkalmazások telepítése. csomagok és portok).

23.2. Az alapok

23.2.1. Mi az I18N/L10N?

A fejlesztõk az I18N elnevezést az angol "internationalization" (idegennyelvûség) szóból származtatják, amiben a szám az elsõ és utolsó betû (az "I" és "N") közt állók mennyiségére utal. Ehhez hasonlóan keletkezett az L10N a "localization" (honosítás) kifejezésbõl. Ezek házasságából jöttek létre az I18N/L10N módszerei, protokolljai és mindazon alkalmazásai, melyekkel a felhasználók a választott nyelvüket használni tudják.

Az I18N alkalmazások céljak eléréséhez függvénykönyvtárakban implementált I18N készleteket használnak. Ezzel lehetõvé válik a fejlesztõik számára, hogy összegyûjtsék a programukban megjelenõ összes szöveget egyetlen állományba, majd azt külön lefordítsák a különbözõ nyelvekre. Mi is ezen konvenció követésére szeretnénk bíztatni minden programozót.

23.2.2. Miért használjuk az I18N/L10N-t?

Az I18N/L10N mindenhol jól jöhet, ahol idegennyelvû adatot akarunk megjeleníteni, bekérni vagy feldolgozni.

23.2.3. Milyen nyelveket támogat az I18N?

Az I18N és L10N nem korlátozódik a FreeBSD tudására. Jelenleg a világban beszélt legelterjedtebb nyelvek mindegyikét használhatjuk bennük. Csak hogy néhányat említsünk közülük: kínai, német, japán, koreai, francia, orosz, vietnámi és még sok más.

23.3. A honosítás használata

Az I18N minden adottságával együtt független a FreeBSD-tõl, egy egyezményes rendszer. Mindenkit bátorítunk arra, hogy segítse a FreeBSD-t ennek az egyezménynek a betartásában.

A honosítás beállításai három fõbb részre tagolhatóak: a nyelv kódja, az ország kódja és a kódolás. A nyelvi beállítások nevei is ezekbõl állnak össze, az alábbi séma szerint:

```
NyelviKód_OrszágKód.Kódolás
```

23.3.1. A nyelv és az ország kódja

Ha a FreeBSD (vagy bármilyen más, az I18N-t ismerõ) rendszert honosítani akarunk az adott nyelvre, akkor a felhasználónak ismernie kell az adott országra és nyelvre vonatkozó kódokat (az országkód fogja elárulni az alkalmazásnak, hogy a nyelv melyik változatát használja). Ezenkívül a böngészõk, SMTP/POP szerverek és webszerverek stb. is ennek alapján fognak döntéseket hozni. Íme néhány nyelv/ország kódja:

Nyelv/ország kódja	Leírás
en_US	Angol - Egyesült Államok
ru_RU	Orosz - Oroszország
zh_TW	Hagyományos kínai - Tajvan

23.3.2. Kódolások

Bizonyos nyelvek 8 bites, széles vagy több byte-os, nem ASCII kódolású karaktereket használnak, melyekrõl a multibyte(3) man oldalán olvashatunk részletesebben. Ezeket régebbi alkalmazások egyáltalán nem ismerik fel, és hibásan vezérlõkaraktereknek tulajdonítják. Az újabbak általában már felismerik a 8 bites karaktereket. A felhasználóknak az alkalmazásokat a széles vagy a több byte-os karakterek használatához vagy újra kell fordítaniuk, vagy pedig megfelelõen be kell állítaniuk, az implementációtól függõen. A széles vagy több byte-os karakterek beolvasásához és feldolgozásához a FreeBSD Portgyûjtemény nyelvenként tartalmaz különféle programokat. A konkrét részletek megértéséhez olvassuk el az érintett FreeBSD portok I18N dokumentációját.

Vagyis a felhasználóknak át kell nézniük az alkalmazáshoz tartozó dokumentációt, mivel ebbõl tudhatják meg, hogyan állítsák be ezeket megfelelõen vagy milyen értékeket adjanak át a configure/Makefile/fordító hármasnak.

Amiket esetleg érdemes lehet ezzel kapcsolatban észben tartanunk:

- A nyelvfüggõ egyszerû karakteres készletek (lásd multibyte(3)), például ISO8859-1, ISO8859-15, KOI8-R, CP437.
- A széles vagy több byte-os kódolások, például az EUC, Big5.

A karakterkészletek jelenleg elérhetõ listáját meg tudjuk tekinteni az IANA adatbázisában.

A FreeBSD helyettük X11-kompatibilis nyelvi kódolásokat használ.

23.3.3. I18N alkalmazások

A FreeBSD port- és csomagrendszerében az I18N alkalmazások a könnyebb felismerhetõség érdekében a nevükben tartalmazzák az `I18N` megnevezést. Nem minden esetben támogatják a szükséges nyelvet.

23.3.4. A nyelvi beállítások megadása

Általában elegendõ annyi, hogy a kívánt nyelvi beállítás nevét exportáljuk az általunk használt parancsértelmezõ `LANG` környezeti változójába. Ez megtehetõ a felhasználói könyvtárunkban található ~/.login_conf, vagy a felhasználói parancsértelmezõ indító állományában (~/.profile, ~/.bashrc, ~/.cshrc). Nem szükséges a nyelvi beállítások részleteit, mint például az `LC_CTYPE`, `LC_CTIME` változókat, megadni. A pontosabb részleteket a FreeBSD adott nyelvre vonatkozó dokumentációjában találjuk meg.

A következõ két környezeti változót kell megadnunk az említett konfigurációs állományokban:

- A `LANG` változót a POSIX® setlocale(3) családjának
- A `MM_CHARSET` változót az alkalmazás MIME karakterkészletéhez

Ez magában foglalja a felhasználói parancsértelmezõ, az adott alkalmazás és az X11 beállítását.

A nyelvi beállítások megadásának módszerei

Két módszer létezik a nyelvi beállítások megadására, ezen kettõrõl fogunk a továbbiakban beszélni. Az elsõ (és egyben ajánlott) ezek közül a bejelentkezési osztályban levõ környezeti változók beállítása, a második pedig környezeti változók hozzáadása a parancsértelmezõ rendszerszintû indító állományához.

Beállítás a bejelentkezési osztályokkal

Ezzel a módszerrel a nyelvi beállítás nevéhez és a MIME karakterkészlethez kötõdõ környezeti változókat az összes létezõ parancsértelmezõ számára csak egyszer kell megadnunk ahelyett, hogy külön mindegyikük indítóállományában szerepeltetnénk. A felhasználó a saját részét maga is elvégezheti, míg a rendszer szintjén adminisztrátori jogosultságokat igényel.

Felhasználói szintû beállítás

Íme példa gyanánt a felhasználó könyvtárában egy egyszerû .login_conf állomány, amiben mind a két változót Latin-1 kódolásra állítottuk:

```
me:\
    :charset=ISO-8859-1:\
    :lang=de_DE.ISO8859-1:
```

Ebben a .login_conf példában a változókat BIG-5 kódolású hagyományos kínai nyelvre állítjuk. Észrevehetjük, hogy itt sokkal több változó beállítására van szükségünk, mivel egyes szoftverek nem kezelik megfelelõen a nyelvi beállításokat kínai, japán és koreai nyelvek esetén.

```
# Azok a felhasználók, akik nem kívánnak tajvani pénz- vagy idõ formátumot
# használni, egyenként írják át a változókat
me:\
    :lang=zh_TW.Big5:\
    :setenv=LC_ALL=zh_TW.Big5:\
    :setenv=LC_COLLATE=zh_TW.Big5:\
    :setenv=LC_CTYPE=zh_TW.Big5:\
    :setenv=LC_MESSAGES=zh_TW.Big5:\
    :setenv=LC_MONETARY=zh_TW.Big5:\
    :setenv=LC_NUMERIC=zh_TW.Big5:\
    :setenv=LC_TIME=zh_TW.Big5:\
    :charset=big5:\
    :xmodifiers="@im=gcin": # a gcin beállítása XIM szerverként
```

A többit lásd a Rendszergazdai szintû beállítások résznél és a login.conf(5) man oldalon.

Rendszergazdai szintû beállítás

Ellenõrizzük, hogy a felhasználó /etc/login.conf állományban szereplõ bejelentkezési osztálya a megfelelõ nyelvet állítja be. Gyõzõdjünk meg róla, hogy az alábbi beállítások helyet kapnak az /etc/login.conf állományban:

```
nyelv_neve|A hozzáférés típusának leírása:\
    :charset=MIME_karakterkészlet:\
    :lang=nyelvi_beállítás_neve:\
    :tc=default:
```

Folytassuk tovább az elõbbi Latin-1-es példánk szerint:

```
nemet|Nemet felhasznalok hozzaferesei:\
    :charset=ISO-8859-1:\
    :lang=de_DE.ISO8859-1:\
    :tc=default:
```

Mielõtt megváltoztatnánk a felhasználók bejelentkezési osztályait, adjuk ki a következõ parancsot:

```
# cap_mkdb /etc/login.conf
```

Ezzel a /etc/login.conf új tartalma láthatóvá válik a rendszer számára.

A bejelentkezési osztály megváltoztatása a vipw(8) programmal

A `vipw` segédprogramot új felhasználók hozzáadására használjuk, aminek eredményeképpen egy ehhez hasonló bejegyzést tudunk létrehozni:

```
felhasznalo:jelszo:1111:11:nyelv:0:0:Felhasznalo neve:/home/felhasznalo:/bin/sh
```

A bejelentkezési osztály megváltoztatása az adduser(8)-rel

Az `adduser`-rel az alábbiak szerint tudunk új felhasználókat felvenni a rendszerbe:

- Adjuk hozzá a `defaultclass = nyelv` sort az /etc/adduser.conf-hoz. Ne felejtsük el, hogy ezután minden olyan felhasználónál a `default` bejelentkezési osztályt meg kell adni, akik nem ezt a nyelvet használják.
- Egy másik megoldás lehet, hogy a adduser(8) használata során minden felhasználó esetén külön megadjuk a nyelvet az

  ```
  Enter login class: default []:
  ```

  rész megjelenésekor.

- Vagy használhatjuk az alábbit az egyes eltérõ nyelvû felhasználók hozzáadásánál:

  ```
  # adduser -class nyelv
  ```

A bejelentkezési osztály megváltoztatása a pw(8)-vel

Amennyiben a pw(8)-t használjuk új felhasználók hozzáadására, így érdemes meghívnunk:

```
# pw useradd felhasználó_neve -L nyelv
```

Beállítás a parancsértelmezõ indító állományával

Ezt a módszert nem javasoljuk, mivel parancsértelmezõnként eltérõ beállítást kíván. Használjuk helyette a bejelentkezési osztályokkal megvalósított módszert.

A nyelvi beállítás nevének és a MIME karakterkészlet beállításához egyszerûen csak adjuk meg a lenti /etc/profile és/vagy /etc/csh.login parancsértelmezõ indító állományokban bemutatott környezeti változót. Továbbra is a német nyelvet használjuk a példánkban:

Az /etc/profile esetén:

```
LANG=de_DE.ISO8859-1; export LANG
```

```
MM_CHARSET=ISO-8859-1; export MM_CHARSET
```

Vagy a /etc/csh.login esetén:

```
setenv LANG de_DE.ISO8859-1
setenv MM_CHARSET ISO-8859-1
```

Úgy is megoldhatjuk ezt a feladatot, ha fenti utasításokat a /usr/shared/skel/dot.profile (hasonló a fentebb említett /etc/profile állományhoz) vagy /usr/shared/skel/dot.login (hasonló a fentebb említett /etc/csh.login állományhoz) esetén hajtjuk végre.

X11 esetén:

Adjuk meg a $HOME/.xinitrc állományban:

```
LANG=de_DE.ISO8859-1; export LANG
```

Vagy:

```
setenv LANG de_DE.ISO8859-1
```

Attól függõen, milyen parancsértelmezõt használunk (lásd fentebb).

23.3.5. A konzol beállítása

Az összes egyszerû karakteres készlet esetén a kérdéses nyelvhez megfelelõ konzolos betûtípust az /etc/rc.conf állományban tudjuk beállítani:

```
font8x16=betûtípus_neve
font8x14=betûtípus_neve
font8x8=betûtípus_neve
```

Itt a *betûtípus_neve* az .fnt kiterjesztés elhagyásával a /usr/shared/syscons/fonts könyvtárban található állományok nevébõl adható meg.

Ha szükséges állítsuk még be a megfelelõ billentyû- és betûkiosztást is a `sysinstall` segítségével. Ahogy sikerült elindítanunk a sysinstallt, válasszuk a Configure (Beállítások) pontot, majd a Console (Konzol)-t! Vagy ehelyett beírhatjuk az alábbi sorokat a /etc/rc.conf állományba:

```
scrnmap=betûkiosztás_neve
keymap=billentyûkiosztás_neve
keychange="funkcióbillentyû_sorszáma szekvencia"
```

Itt a *betûkiosztás_neve* a /usr/shared/syscons/scrnmaps könyvtárban található állományok nevébõl

származtatható az .scm kiterjesztés elhagyásával. A betûkiosztásokat általában a 9 bites karaktermátrixszal rendelkezõ VGA megjelenítõk problémáinak megoldására lehet használni, mivel így az eredetileg 8 bittel ábrázolt betûket ki lehet tolni az ilyen típusú kártyák pszeudografikus területérõl.

Ha aktiváltuk a moused egérkezelõ démont az /etc/rc.conf állományban az alábbi sor megadásával:

```
moused_enable="YES"
```

akkor a következõ bekezdésben rá is térhetünk az egérmutató adatainak vizsgálatára.

A syscons(4) meghajtóban található egérmutató alapértelmezés szerint a 0xd0 - 0xd3 karaktereket foglalja el a karakterkészletben. Ha a nyelv ezeket használja, arrébb kell költöztetnünk ezt az egérmutató által elfoglalt sávot. A FreeBSD-ben az /etc/rc.conf állományon keresztül érhetjük el:

```
mousechar_start=3
```

A *billentyûkiosztás_neve* a /usr/shared/syscons/keymaps könyvtárból, a .kbd kiterjesztés elhagyásával keletkezik. Ha nem vagyunk benne biztosak, melyik kiosztást is kellene használnunk, a kbdmap(1) segítségével a rendszer újraindítása nélkül kipróbálhatjuk a rendelkezésre álló billentyûkiosztásokat.

A `keychange` használatára többnyire a funkcióbillentyûk adott terminháltípushoz egyeztetéséhez van szükség, mert a funkcióbillentyûk szekvenciái nem adhatóak meg a billentyûkiosztásban.

Ezeken felül érdemes megbizonyosodnunk róla, hogy a /etc/ttys állományban jól állítjuk be a terminál típusát minden `ttyv*` bejegyzés esetén. Az aktuálisan elõre beállított kapcsolatok a következõk:

Karakterkészlet	**Terminháltípus**
ISO8859-1 vagy ISO8859-15	`cons25l1`
ISO8859-2	`cons25l2`
ISO8859-7	`cons25l7`
KOI8-R	`cons25r`
KOI8-U	`cons25u`
CP437 (alapértelmezett VGA)	`cons25`
US-ASCII	`cons25w`

A széles és több byte-os karaktereket használó nyelvek esetén használjuk a /usr/ports/nyelv könyvtárban megfelelõ FreeBSD portot. Egyes portok konzolosként jelennek meg, miközben a rendszer soros virtuális terminálként látja ezeket, ezért fenn kell tartanunk elegendõ virtuális terminált mind az X11, mind pedig pszeudo-soros konzol számára. Itt látható a konzolon más nyelvet használó alkalmazások részleges listája:

Nyelv	Hely
Hagyományos kínai (BIG-5)	chinese/big5con
Japán	japanese/kon2-16dot vagy japanese/mule-freewnn
Koreai	korean/han

23.3.6. Az X11 beállítása

Habár az X11 nem része a FreeBSD projektnek, megemlítünk vele kapcsolatban néhány hasznos információt a FreeBSD felhasználók számára is. Még több részletet a Xorg honlapjáról vagy az általunk használt X11 szerver dokumentációjából tudhatunk meg.

Az ~/.Xresources állományban további I18N beállításokat finomíthatunk alkalmazásonként (például betûtípusok, menük stb.).

Betûtípusok megjelenítése

Telepítsük fel az Xorg (x11-servers/xorg-server) vagy az XFree86™ (x11-servers/XFree86-4-Server) szerverek valamelyikét, majd telepítsük a nyelvhez tartozó TrueType® betûtípusokat. Ezután a megfelelõ nyelvi beállítása megadása révén már látni fogjuk a kiválasztott nyelven megjelenõ menüket és egyéb szövegeket.

Idegennyelvû karakterek bevitele

Az X11 beviteli módszerének (X11 Input Method, XIM) protokollja egy új szabvány az összes X11 kliensh ez. Minden X11 alkalmazást olyan XIM-kliensként kell elkészíteni, amelyek a bemenõ adatokat az XIM beviteli szerverektõl kapják. Különbözõ XIM szerverek érhetõek el az eltérõ nyelvekhez.

23.3.7. Nyomtatók beállítása

Egyes egyszerû karakteres készletek általában hardveresen beépítve megtalálhatóak a nyomtatókban. A széles és több byte-os karakterkészletek azonban külön beállítást igényelnek, amire az apsfilter használatát javasoljuk. A megfelelõ nyelvhez szabott eszközökkel át is lehet konvertálni PostScript® vagy PDF formátumba a nyomtatni kívánt dokumentumot.

23.3.8. A rendszermag és az állományrendszerek

A FreeBSD gyors állományrendszere (Fast File System, FFS) szabályosan kezeli a 8 bites karaktereket, tehát tetszõleges egyszerû karakteres készlet (lásd multibyte(3)) használható vele, viszont a karakterkészlet nevét nem tárolja el az állományrendszerben. Emiatt a neveket nyersen kezeli, semmit sem tud a kódolásukról. Az FFS hivatalosan még nem támogat semmilyen fajta széles vagy több byte-os karakterkészletet. Léteznek azonban független javítások az FFS-hez, amelyek lehetõvé teszik ilyen széles vagy több byte-os karakterek használatát. Ezek csak átmeneti és nem hordozható megoldások, olyan módosítások, amelyekrõl úgy döntöttünk, nem vesszük fel ezeket a forrásfába. Az érintett nyelvek honlapjain elérhetjük ezeket a javításokat és többet megtudhatunk róluk.

A FreeBSD MS-DOS® állományrendszere konfigurálható úgy, hogy képes legyen konvertálni az MS-DOS® Unicode és a kiválasztott FreeBSD állományrendszerének karakterkészlete között. Errõl bõvebben a mount_msdosfs(8) man oldalon olvashatunk.

23.4. I18N programok fordítása

Számos FreeBSD port rendelkezik I18N támogatással. Ezek egy részének nevében szerepel az -I18N jelzés. Az ilyen és sok más hasonló program beépítetten ismeri az I18N-t, így nem igényelnek külön beállításokat.

Néhány alkalmazás azonban, mint például a MySQL, esetén az adott karakterkészletnek megfelelõ módon kell beállítani a Makefile állományt. Ezt általában magában a Makefile állományban tudjuk megtenni, vagy pedig a configure megfelelõ paraméterezésével.

23.5. A FreeBSD honosítása adott nyelvekre

23.5.1. Az orosz nyelv (KOI8-R kódolás)

A KOI8-R kódolásról bõvebben a KOI8-R oldalán (orosz hálózati karakterkészlet) tájékozódhatunk.

A nyelvi beállítások megadása

Írjuk a következõ sorokat a ~/.login_conf állományunkba:

```
me:Az en hozzaferesem:\
    :charset=KOI8-R:\
    :lang=ru_RU.KOI8-R:
```

Valamint lásd a fejezet korábbi részeiben említett példákat a nyelvi beállítások megadására.

A konzol beállítása

- Tegyük hozzá a következõ sort az /etc/rc.conf állományunkhoz:

  ```
  mousechar_start=3
  ```

- Illetve használjuk az /etc/rc.conf állományban még a következõ beállításokat is:

  ```
  keymap="ru.utf-8"
  scrnmap="utf-82cp866"
  font8x16="cp866b-8x16"
  font8x14="cp866-8x14"
  font8x8="cp866-8x8"
  ```

- A /etc/ttys állományban szereplõ mindegyik `ttyv*` bejegyzésnél adjuk meg terminéltípusnak a `cons25r`-t.

Valamint lásd a fejezet korábbi részében bemutatott példákat a konzol beállítására.

A nyomtatás beállítása

Mivel a legtöbb nyomtató hardveresen tartalmazza a CP866 kódlapot az orosz karakterek támogatásához, használnunk kell egy kimeneti szûrõt a KOI8-R kódolású karakterek CP866 kódolásúra konvertálásához. Egy ilyen szûrõ alapértelmezés szerint telepítésre kerül a /usr/libexec/lpr/ru/koi2alt állományba. Az orosz nyomtatóhoz tartozó bejegyzés valahogy így néz ki az /etc/printcap állományban:

```
lp|Orosz helyi sornyomtato:\
    :sh:of=/usr/libexec/lpr/ru/koi2alt:\
    :lp=/dev/lpt0:sd=/var/spool/output/lpd:lf=/var/log/lpd-errs:
```

A bõvebben magyarázathoz lásd a printcap(5) man oldalt.

Az MS-DOS® állományrendszere és az orosz állománynevek

A most következõ példa fstab(5) bejegyzés azt mutatja meg, hogy lehet bekapcsolni az orosz állománynevek támogatását a csatlakoztatandó MS-DOS® állományrendszereken:

```
/dev/ad0s2      /dos/c  msdos   rw,-Wkoi2dos,-Lru_RU.KOI8-R 0 0
```

Az `-L` kapcsolóval kiválasztjuk a használni kívánt nyelvi beállítás nevét, és a `-W` kapcsolóval megadjuk a karakterek átváltásához szükséges táblázatot. A `-W` kapcsoló használata során mindenképpen csatlakoztassuk a /usr állományrendszert még az MS-DOS® partíció elõtt, mivel az átváltáshoz használt táblázatok a /usr/libdata/msdosfs könyvtárban találhatóak meg! A részleteket a mount_msdosfs(8) man oldalon találhatjuk meg.

Az X11 beállítása

1. Adjuk meg elõször a leírtak szerint a nem X-es nyelvi beállításokat.
2. Ha Xorg-ot használunk, telepítsük a x11-fonts/xorg-fonts-cyrillic csomagot.

   Ellenõrizzük a /etc/X11/xorg.conf állományban a `"Files"` szakaszt. Az alábbi sort mindegyik más `FontPath` bejegyzés *elõtt* kell szerepeltetnünk:

   ```
   FontPath   "/usr/X11R6/lib/X11/fonts/cyrillic"
   ```

   A portok között találhatunk még további cirill betûtípusokat.

3. Az orosz billentyûzet életre keltéséhez írjuk be a következõket az xorg.conf állomány `"Keyboard"` szakaszába:

   ```
   Option "XkbLayout"   "us,ru"
   ```

```
Option "XkbOptions"  "grp:toggle"
```

Ellenõrizzük, hogy a `XkbDisable` ki van kapcsolva (ki van kommentezve) ebben a szakaszban.

A `grp:toggle` beállítás esetén az orosz/latin (RUS/LAT) átkapcsolás gombja a `jobb Alt` lesz, míg a `grp:ctrl_shift_toggle` beállításnál a `Ctrl` + `Shift`. A `grp:caps_toggle` esetén az orosz/latin váltás a `CapsLock` billentyûvel történik. Ilyenkor (de csak latin módban) a megszokott `CapsLock` funkció továbbra is elérhetõ a `Shift` + `CapsLock` kombinációval. A `grp:caps_toggle` valamiért nem mûködik az Xorgban.

Ha van "Windows®" billentyûnk a billentyûzeten és azt tapasztaljuk, hogy egyes nem-alfabetikus billentyûk rosszul kerülnek kiosztásra orosz módban, adjuk hozzá a következõ sort az xorg.conf állományhoz:

```
Option "XkbVariant" ",winkeys"
```

Az orosz XKB billentyûzet egyes nem honosított alkalmazások esetén nem mûködik.

A kis mértékben honosított alkalmazások esetén javasolt meghívni a `XtSetLanuageProc(NULL, NULL, NULL);` függvényt valahol a program elején.

Az X11 alkalmazások honosításához további útmutatásokat a KOI8-R X Window-ra címû leírásban találhatunk.

23.5.2. Hagyományos kínai honosítás tajvaniak számára

A FreeBSD-Taiwan projekt készített a FreeBSD-hez egy kínainak szóló hogyant, amely elérhetõ a http://netlab.cse.yzu.edu.tw/~statue/freebsd/zh-tut/ címen és számos kínai portot használ. A `FreeBSD kínai hogyan` jelenlegi szerkesztõje Shen Chuan-Hsing (statue@freebsd.sinica.edu.tw).

Chuan-Hsing Shen (statue@freebsd.sinica.edu.tw) létrehozta a Kínai FreeBSD gyûjteményt (Chinese FreeBSD Collection, CFC) a FreeBSD-Taiwan `zh-L10N-tut` munkáját felhasználva. A hozzá tartozó csomagok és szkriptek elérhetõek a ftp://freebsd.csie.nctu.edu.tw/pub/taiwan/CFC/ címen.

23.5.3. Honosítás német (és minden más ISO 8859-1 kódolású) nyelvre

Slaven Rezic (eserte@cs.tu-berlin.de) készített egy írást, amely elmagyarázza, hogyan használjunk német nemzeti karaktereket a FreeBSD alatt. Ez a leírás németül készült és a http://user.cs.tu-berlin.de/~eserte/FreeBSD/doc/umlaute/umlaute.html címen érhetõ el.

23.5.4. Honosítás görög nyelvre

Nikos Kokkalis nickkokkalis@gmail.com egy teljes cikket írt a FreeBSD görög nyelvi támogatásáról. Ez elérhetõ a FreeBSD hivatalos görög nyelvû dokumentációjában, a https://www.FreeBSD.org/doc/el/articles/greek-language-support/ címen. Felhívjuk a figyelmet, hogy az *csak* görög nyelven érhetõ el.

23.5.5. Honosítás japán és koreai nyelvekre

A japán honosításhoz lásd http://www.jp.FreeBSD.org/, a koreaihoz pedig lásd http://www.kr.FreeBSD.org/.

23.5.6. Idegennyelvû FreeBSD dokumentáció

Néhány FreeBSD felhasználó lefordította a FreeBSD dokumentációjának egyes részeit más nyelvekre is. Munkájuk elérhetõ a fõoldalon található linkeken keresztül vagy a /usr/shared/doc könyvtárban.

Chapter 24. A FreeBSD frissítése és frissen tartása

24.1. Áttekintés

A FreeBSD a kiadások közt is állandó fejlõdésben van. Vannak felhasználók, akik a hivatalosan kiadott változatokat használják, és vannak, akik szeretik folyamatosan nyomonkövetni a fejlesztéseket. Emellett viszont a hivatalos kiadások esetében szükség lehet bizonyos biztonsági frissítések és kritikus javítások alkalmazására. Függetlenül a pillanatnyilag használt változattól, a FreeBSD alaprendszerében megtalálható minden olyan eszköz, amellyel könnyedén frissíteni tudunk a különbözõ verziók között. Ebben a fejezetben segítünk dönteni a fejlesztõi változat és a kiadások használata között. Továbbá megismerhetjük a rendszer frissítéséhez használható alapvetõ eszközöket.

A fejezet elolvasása során megismerjük:

- milyen segédprogramokkal tudjuk frissíteni az alaprendszert és a Portgyûjteményt;
- hogyan tartsuk naprakészen rendszerünket a freebsd-update, CVSup, CVS vagy CTM használatával;
- hogyan vessük össze a telepített rendszerünk aktuális állapotát egy ismert eredeti változattal;
- hogyan frissítsük a dokumentációt CVSup vagy dokumentációs portok segítségével.
- a két fejlesztõi ág, a FreeBSD-STABLE és a FreeBSD-CURRENT közti különbséget;
- a `make buildworld` (stb.) segítségével hogyan fordítsuk és telepítsük újra az egész alaprendszert.

A fejezet elolvasásához ajánlott:

- a hálózati kapcsolatunk helyes beállítása (Egyéb haladó hálózati témák);
- a külsõ szoftverek telepítésének ismerete (Alkalmazások telepítése. csomagok és portok).

A fejezetben a FreeBSD forrásainak frissítését a `cvsup` parancs segítségével fogjuk elvégezni. Ehhez telepítsük a net/cvsup portot vagy csomagot (ha a `cvsup` parancsot nem akarjuk grafikus felületen keresztül használni, akkor elegendõ csak a net/cvsup-without-gui portot). Ha a FreeBSD 6.2-RELEASE vagy késõbbi változatával rendelkezünk, akkor elegendõ csak az alaprendszer részeként elérhetõ csup(1) programot használnunk.

24.2. A FreeBSD frissítése

A biztonsági javítások telepítése minden számítógépes szoftver, különösen az operációs rendszerek számára lényeges mozzanat. Nagyon hosszú ideig ez a FreeBSD esetében nem volt könnyen megoldható: a javításokat közvetlenül a forráskódon kellett elvégezni, ezekbõl újrafordítani a rendszert, majd telepíteni.

Ez a nehézség mostanra viszont már elhárult, mivel a FreeBSD legfrissebb verziói már

tartalmaznak egy `freebsd-update` nevû segédprogramot, amellyel mindez leegyszerûsödik. Ez a program két külön funkciót lát el. Elõször is, lehetõvé teszi, hogy a FreeBSD alaprendszer újrafordítása és -telepítése nélkül javítsunk biztonsági és egyéb apró hibákat, valamint másodsorban támogatja a kisebb és nagyobb verziójú kiadások közti váltást.

Ezek a bináris frissítések azonban csak a FreeBSD biztonsági csapata által is felügyelt architektúrák és kiadások esetén érhetõek el. Emellett bizonyos lehetõségek használatához, például a FreeBSD verziói közti átállás támogatásához a freebsd-update(8) legújabb változata szükségeltetik. Ezért ne felejtsük el alaposan átolvasni a legújabb kiadásokról szóló bejelentéseket mielõtt frissítenénk rájuk, mivel ezzel kapcsolatban fontos információkat tartalmazhatnak. Az említett bejelentések a http://www.FreeBSD.org/releases/ címen érhetõek el.

Ha a `crontab` már hivatkozik a `freebsd-update` programra, akkor a most következõ mûvelet elkezdése elõtt tiltsuk le.

24.2.1. A konfigurációs állományok

Ha változtatnénk szeretnénk a frissítési folyamaton, ekkor a programhoz tartozó, /etc/freebsd-update.conf nevû konfigurációs állományt kell módosítanunk. Az opciók részletes ismertetéssel rendelkeznek, habár némelyiknél még további magyarázat kellhet:

```
# Az alaprendszerben frissíteni kívánt komponensek
Components src world kernel
```

Ezzel a paraméterrel határozhatjuk meg, hogy a FreeBSD mely részei kerüljenek frissítésre. Alapértelmezés szerint a program frissíti a forrásokat, a teljes alaprendszert és a rendszermagot. Komponensként a telepítésnél választható elemeket adhatjuk meg, például "world/games" hozzáadásakor a games kategória elemei is folyamatosan frissülni fognak. Az "src/bin" megadásakor pedig az src/bin könyvtár tartalma frissül.

Ezt a beállítást a legjobb meghagyni az alapértelmezett értéken, mivel a további elemek megadásánál egyenként fel kell sorolni a frissítendõ komponenseket. Ha itt viszont kifelejtünk valamit, akkor könnyen megeshet, hogy a források és a binárisok verziója elcsúszik egymástól.

```
# Az IgnorePaths beállítás után megadott szövegre illeszkedõ összes
# bejegyzés frissítése kimarad
IgnorePaths
```

Ennél a beállításnál azokat a könyvtárakat kell megadnunk, amelyeket (és tartalmukat) ki szeretnénk hagyni a frissítés során. Ezek lehetnek például a /bin vagy az /sbin. Így meg tudjuk akadályozni, hogy `freebsd-update` esetleg felülírjon valamilyen helyi változtatást a rendszerünkben.

```
# Az UpdateIfUnmodified beállítás után megadott elérési útvonalakon csak
# a felhasználó által még nem módosított állományok fognak frissülni
# (hacsak a módosításokat össze nem fésüljük, lásd lentebb)
```

```
UpdateIfUnmodified /etc/ /var/ /root/ /.cshrc /.profile
```

A megadott könyvtárakban csak azokat a konfigurációs állományokat fogja frissíteni, amelyeket nem változtattuk meg. Amennyiben bármelyikük eltér az eredetileg frissítendõ változattól, azt a program nem módosítja. Létezik egy másik hasonló beállítás, a `KeepModifiedMetadata`, amely hatására a `freebsd-update` az összefésülés során elmenti a változtatásokat.

```
# A MergeChanges beállításnál szereplõ állományok helyi módosításait
# automatikusan összefésüljük a FreeBSD újabb verziójára frissítése közben
MergeChanges /etc/ /var/named/etc/
```

Itt azokat a könyvtárakat adhatjuk meg, amelyekben a `freebsd-update` számára engedélyezzük a konfigurációs állományok új verziójának összefésülését a jelenlegi állapottal. Az összefésülés lényegében a mergemaster(8) használatánál már megszokott módon, diff(1) formátumban érkezõ módosítások sorozata alapján történik. Ekkor egy szövegszerkesztõ segítségével felügyelhetjük az összefésülés menetét vagy megállíthatjuk a `freebsd-update` futását. Ha kétségeink adódnak, akkor egyszerûen mentsük le az /etc könyvtárat és fogadjuk el mindegyik összefésülés eredményét. A `mergemaster` mûködésérõl a A `mergemaster` ad részletesebb tájékoztatást.

```
# A FreeBSD frissítésekor ezt a könyvtárat fogja a program használni a
# letöltött módosítások és az egyéb ideiglenes állományok tárolására
# WorkDir /var/db/freebsd-update
```

Az itt megadott könyvtárba fognak kerülni az elvégzendõ módosítások és az egyéb ideiglenesen keletkezõ állományok. A verziók közti váltás során ebben a könyvtárban ajánlott legalább 1 GB szabad tárterületnek lennie.

```
# A kiadások közti váltás során a Components beállításnál megadott
# elemek kerüljenek csak frissítésre (StrictComponents yes), vagy a
# program próbálja meg magától kitalálni, hogy milyen komponesek
# *lehetnek* fenn a rendszeren és azokat frissítse (StrictComponents
# no)?
# StrictComponents no
```

Ha ennél a beállításnál a `yes` értéket adjuk meg, akkor a `freebsd-update` feltételezni fogja, hogy a `Components` opciónál felsoroltunk minden frissítendõ komponenst és nem próbál meg mást is megváltoztatni. Ilyenkor tehát a `freebsd-update` tulajdonképpen egyedül csak a `Components` által meghatározott elemekhez tartozó állományokat fogja frissíteni.

24.2.2. Biztonsági javítások

A biztonsági javítások mindig egy távoli gépen tárolódnak, a következõ parancsok használatával tölthetõek le és telepíthetõek:

```
# freebsd-update fetch
```

```
# freebsd-update install
```

Amennyiben a rendszermagot is érintik javítások, úgy a rendszert a mûvelet befejezõdésével újra kell indítanunk. Ha minden a megfelelõ módon történt, akkor a rendszerünk már tartalmazni fogja a korábban letöltött és telepített javításokat, és a `freebsd-update` akár beállítható egy naponta végrehajtandó cron(8) feladatnak. Ehhez mindössze a következõ bejegyzést kell elhelyeznünk az /etc/crontab állományban:

```
@daily                                  root    freebsd-update cron
```

A bejegyzés szerint naponta egyszer le fog futni a `freebsd-update`. Ilyenkor, vagyis a `cron` paraméter megadásakor a `freebsd-update` csak ellenõrzi, hogy vannak-e telepítendõ frissítések. Ha talál, akkor automatikusan letölti ezeket a lemezre, de nem telepíti. Helyette levélben értesíti a `root` felhasználót, aki ezután bármikor manuálisan kérheti a telepítést.

Probléma esetén az alábbi paranccsal megkérhetjük a `freebsd-update` programot a legutóbb telepített módosítások visszavonására:

```
# freebsd-update rollback
```

Ha ez a visszavonás a rendszermagra vagy annak moduljaira is vonatkozott, akkor a rendszert újra kell indítanunk a parancs futásának befejezõdésével. A FreeBSD csak ilyenkor képes betölteni az új binárisokat betölteni a memóriába.

A `freebsd-update` önmagától csak a `GENERIC` típusú rendszermagokat képes frissíteni. Ha saját rendszermagot használunk, akkor azt a rendszer többi komponensének frissítését követõen újra kell fordítanunk és telepítenünk. A `freebsd-update` azonban még akkor is érzekelni és frissíteni fogja a `GENERIC` rendszermagot (amennyiben az létezik), ha az éppen nem az aktuális(an futó) rendszermag.

Mindig érdemes tartani egy másolatot a `GENERIC` rendszermagról a /boot/GENERIC könyvtárban. Rengeteg különbözõ probléma felderítésében tud segíteni, illetve ez a Váltás kisebb és nagyobb verziók között szakaszban leírt `freebsd-update` programmal végzett frissítéseknél is hasznos lehet.

Hacsak nem változtatjuk meg az /etc/freebsd-update.conf állományt, a `freebsd-update` a rendszermag forrásait is frissíti a többivel együtt. A saját rendszermag újrafordítása és telepítése ezután a már a megszokott módon elvégezhetõ.

A `freebsd-update` által terjesztett frissítések nem mindig érintik a rendszermagot. Ha a rendszermag forrásai nem változnak egy `freebsd-update install` parancs kiadása során, akkor nem kötelezõ újrafordítani a saját rendszermagot. A `freebsd-update` viszont mindig módosítani fogja a /usr/src/sys/conf/newvers.sh állományt. Itt az aktuális hibajavítás sorszáma szerepel (amelyet a `-p` (mint "patch level" elõtaggal kapcsolnak a rendszer verziójához, és a `uname -r` paranccsal lehet lekérdezni). Ennek megfelelõen tehát a saját rendszermag újrafordítása után, még

> ha semmi más nem is változott, a uname(1) képes pontosan jelezni a rendszerhez készült hibajavítás sorszámát. Ez különösen fontos több rendszer karbantartása során, mivel így könnyen és gyorsan tájékozódhatunk azok naprakészségéről.

24.2.3. Váltás kisebb és nagyobb verziók között

Verziók közti váltás során a külsõ alkalmazások mûködését akadályozó régi tárgykódok és függvénykönyvtárak törlõdni fognak. Ezért javasoljuk, hogy vagy töröljük le az összes portot és telepítsük újra, vagy az alaprendszer frissítése után hozzuk ezeket is naprakész állapotba a ports-mgmt/portupgrade segédprogram segítségével. Elõször minden bizonnyal szeretnék kipróbálni a frissítést, ezt a következõ paranccsal tehetjük meg:

```
# portupgrade -af
```

Ezzel gondoskodunk róla, hogy a minden a megfelelõen telepítõdjön újra. Ha a BATCH környezeti változót a yes értékre állítjuk, akkor a folyamat során megjelenõ összes kérdésre automatikusan a yes választ adjuk, ezáltal önállósítani tudjuk.

Ha saját rendszermagot használunk, akkor ennél valamivel azért több feladatunk van. Szükségünk lesz a GENERIC rendszermagot egy példányára, amelyet másoljunk a /boot/GENERIC könyvtárba. Amennyiben nincs GENERIC típusú rendszermag a rendszerünkön, a következõ módok valamelyikén keresztül tudunk szerezni:

- Ha a saját rendszermagot még csak egyszer fordítottuk, akkor a /boot/kernel.old könyvtárban még megtalálható a GENERIC. Ezt nevezzük át egyszerûen /boot/GENERIC könyvtárra.
- Ha fizikailag hozzá tudunk férni az érintett géphez, akkor a GENERIC egy példányát akár CD-rõl is átmásolhatjuk. Helyezzük be a telepítõlemezt és adjuk ki a következõ parancsokat:

  ```
  # mount /cdrom
  # cd /cdrom/X.Y-RELEASE/kernels
  # ./install.sh GENERIC
  ```

  Itt a X.Y-RELEASE könyvtár nevében értelemszerûen helyettesítsük be az általunk használt változatot. A GENERIC rendszermag ekkor alapértelmezés szerint a /boot/GENERIC könyvtárba kerül.

- Ha az elõbbiek közül egyik sem lehetséges, akkor a GENERIC rendszermagot közvetlenül akár forrásból is lefordíthatjuk és telepíthetjük:

  ```
  # cd /usr/src
  # env DESTDIR=/boot/GENERIC make kernel
  # mv /boot/GENERIC/boot/kernel/* /boot/GENERIC
  # rm -rf /boot/GENERIC/boot
  ```

  A freebsd-update akkor fogja ezt GENERIC rendszermagként felismerni, ha a hozzá tartozó konfigurációs állományt nem módosítjuk. Továbbá javasoljuk, hogy semmilyen speciális

beállítást ne alkalmazzunk a fordítás során (érdemes üresen hagyni ehhez az /etc/make.conf állományt).

Nem kötelezõ újraindítani a rendszert a `GENERIC` rendszermaggal.

A `freebsd-update` képes frissíteni rendszerünket egy adott kiadásra. Például a következõ paraméterek megadásával válthatunk a FreeBSD 6.4 használatára:

```
# freebsd-update -r 6.4-RELEASE upgrade
```

A parancs elindulása után nem sokkal, a váltáshoz szükséges információk összegyûjtéséhez a `freebsd-update` elemzi a konfigurációs állományában megadott beállításokat és a rendszer jelenleg használt verzióját. A képernyõn ekkor sorban megjelennek a program részérõl érzékelt és nem érzékelt komponensek. Mint például ahogy itt látható:

```
Looking up update.FreeBSD.org mirrors... 1 mirrors found.
Fetching metadata signature for 6.3-RELEASE from update1.FreeBSD.org... done.
Fetching metadata index... done.
Inspecting system... done.

The following components of FreeBSD seem to be installed:
kernel/smp src/base src/bin src/contrib src/crypto src/etc src/games
src/gnu src/include src/krb5 src/lib src/libexec src/release src/rescue
src/sbin src/secure src/share src/sys src/tools src/ubin src/usbin
world/base world/info world/lib32 world/manpages

The following components of FreeBSD do not seem to be installed:
kernel/generic world/catpages world/dict world/doc world/games
world/proflibs

Does this look reasonable (y/n)? y
```

Ekkor a `freebsd-update` megpróbálja letölteni a verziók közti váltáshoz szükséges összes állományt. Bizonyos esetekben kérdésekkel fordul a felhasználó felé arra vonatkozóan, hogy miket telepítsen fel vagy mit csináljon.

A saját rendszermag használatakor az iménti lépés valamilyen ehhez hasonló figyelmeztetést fog adni:

```
WARNING: This system is running a "SAJÁT RENDSZERMAG" kernel, which is not a
kernel configuration distributed as part of FreeBSD 6.3-RELEASE.
This kernel will not be updated: you MUST update the kernel manually
before running "/usr/sbin/freebsd-update install"
```

Ez a figyelmeztetés most nyugodtan figyelmen kívül hagyható. A folyamat során a frissített `GENERIC` rendszermagot fogjuk használni.

A javítások letöltését követõen megkezdõdik a telepítésük. A váltás ezen lépése az adott gép aktuális terhelésétõl és sebességétõl függõen változó hosszúságú lehet. Ezután a konfigurációs állományok összefésülése zajlik le - itt általában a emberi felügyeletre is szükség van az állományok összefésülésének irányításához, amelynek folyamatosan láthatóak az eredményei. A meghiúsult vagy kihagyott összefésülések a teljes frissítési folyamat leállását vonják maguk után. Az /etc könyvtárban tárolt fontosabb állományokról, mint például a master.passwd vagy group javasolt elõzetesen biztonsági mentést készíteni és késõbb kézzel hozzájuk adni a változtatásaikat.

A rendszerben ekkor még nem lesz jelen semmilyen konkrét változás, az összes említett javítás és összefésülés egy külön könyvtárban történik. A telepített javításokat és az összefésült konfigurációs állományokat a folyamat végén magának a felhasználónak kell véglegesíteni.

A frissítési eljárás végén a következõ parancs kiadásával tudjuk ténylegesen érvényesíteni az eddig elvégzett módosításokat:

```
# freebsd-update install
```

Elõször mindig a rendszermag és a hozzá tartozó modulok cserélõdnek le. Ahogy ez végrehajtódott, újra kell indítanunk a rendszert. Ha saját rendszermagot használunk, akkor a nextboot(8) parancs segítségével állítsuk be a következõ rendszerindítás során betöltendõ rendszermagot a /boot/GENERIC könyvtárban levõre (ezt frissítettük):

```
# nextboot -k GENERIC
```

Mielõtt újraindítanánk a gépünket a `GENERIC` rendszermaggal, gyõzõdjünk meg róla, hogy szerepel benne minden olyan meghajtó, amely elengedhetetlen a rendszer hiánytalan indításához (és képes lesz újra csatlakozni a hálózathoz, ha éppen távolról adminisztráljuk). Ez különösen olyan esetben fontos, amikor a saját rendszermagunkban beépítetten szerepeltek bizonyos modulok. Ilyenkor a `GENERIC` rendszermag használatakor ezeket a /boot/loader.conf állományon keresztül töltethetjük be ideiglenesen. A frissítés befejezéséig érdemes viszont minden nem létfontosságú szolgáltatást leállítani, leválasztani lemezeket és hálózati megosztásokat stb.

A rendszerünk most már újraindítható a frissített rendszermaggal:

```
# shutdown -r now
```

A rendszer sikeres újraindulása után ismét el kell indítanunk a `freebsd-update` programot, amely korábban már elmentette a frissítés állapotát, emiatt a legutóbbi pontról fog folytatódni, illetve törli az osztott könyvtárak és tárgykódok régebbi változatait. Innen az alábbi paranccsal léphetünk tovább:

```
# freebsd-update install
```

A függvénykönyvtárak verziói közti eltérések mértékétõl függõen elképzelhetõ, hogy a telepítés az említett három fázis helyett kettõben történik.

Most pedig újra kell fordítanunk vagy telepítenünk az összes általunk korábban használt külsõ alkalmazást. Erre azért van szükségünk, mert bizonyos alkalmazások a verziók közti váltás során törölt programkönyvtáraktól függtek. Ennek automatizálásában a ports-mgmt/portupgrade lesz segítségünkre. Az alkalmazások frissítésének elindításához a következõ parancsokat használjuk:

```
# portupgrade -f ruby
# rm /var/db/pkg/pkgdb.db
# portupgrade -f ruby18-bdb
# rm /var/db/pkg/pkgdb.db /usr/ports/INDEX-*.db
# portupgrade -af
```

A parancsok lefutását követõen a `freebsd-update` utolsó hívásával zárjuk le a frissítést. Ezzel a paranccsal tudunk tehát pontot tenni a frissítési procedúra végére:

```
# freebsd-update install
```

Ha a `GENERIC` rendszermagot csak átmenetileg használtuk, akkor most már a megszokott módon fordíthatunk és telepíthetünk magunk egy saját rendszermagot.

Indítsuk újra a rendszert a FreeBSD frissített változatával. A folyamat ezzel véget ért.

24.2.4. Rendszerek állapotainak összehasonlítása

A `freebsd-update` ragyogóan felhasználható a FreeBSD egy telepített változatának és egy általunk garantáltan megbízható példányának összevetésére. Ilyenkor a rendszerhez tartozó segédprogramokat, programkönyvtárakat és konfigurációs állományokat ellenõriztethetjük le. Az összehasonlítást ezzel a paranccsal kezdhetjük meg:

```
# freebsd-update IDS >> eredmeny.idk
```

Habár a parancs neve IDS (intrusion detection system), nem helyettesít semmilyen olyan behatolásjelzõ megoldást, mint amilyen például a security/snort. Mivel a `freebsd-update` adatokat tárol a lemezen, teljesen kézenfekvõ a hamisítás lehetõsége. Míg ennek eshetõsége adott mértékben visszaszorítható a `kern.securelevel` csökkentésével és a `freebsd-update` által használt adatok írásvédett állományrendszerre helyezésével, erre a problémára az ideális megoldást mégis egy teljes biztonságban tudható referencia rendszer jelentheti. Ennek tárolására alkalmas lehet például egy DVD vagy egy külsõ USB-egység.

A parancs kiadása után megkezdõdik a rendszer vizsgálata, és az ellenõrzés során folyamatosan jelennek meg az átvizsgált állományok a hozzájuk tartozó ismert és kiszámított sha256(1)-kódjukkal együtt. Mivel a képernyõn túlságosan gyorsan elúsznának az eredmények, ezért ezeket egy eredmeny.idk nevû állományba mentjük a késõbbi elemzésekhez.

Az így keletkezõ állomány sorai ugyan meglehetõsen hosszúak, de szerencsére viszonylag könnyen értelmezhetõek. Például az adott kiadásban szereplõ állományoktól eltérõeket ezzel a paranccsal kérdezhetjük le:

```
# cat eredmeny.idk | awk '{ print $1 }' | more
/etc/master.passwd
/etc/motd
/etc/passwd
/etc/pf.conf
```

A példában most csak az elsõ néhány állományt hagytuk meg, gyakran tapasztalhatunk viszont ennél többet. Ezek közül bizonyos állományok értelemszerûen eltérnek, mint itt például az /etc/passwd, mert idõközben új felhasználókat adtunk a rendszerhez. Máskor egyéb állományok, például modulok nevei is felbukkanhatnak, mert tegyük fel, hogy a `freebsd-update` már frissítette ezeket. Ha ki szeretnénk zárni valamilyen állományokat vagy könyvtárakat az ellenõrzésbõl, egyszerûen csak soroljuk fel ezeket az /etc/freebsd-update.conf állományban megjelenõ `IDSIgnorePaths` beállításnál.

A korábban tárgyaltaktól függetlenül ez a rendszer alkalmas bonyolultabb frissítési folyamatok kisegítésére is.

24.3. A Portgyûjtemény frissítése a Portsnap használatával

A FreeBSD alaprendszer a Portgyûjtemény frissítéséhez is tartalmaz egy portsnap(8) elnevezésû segédprogramot. Ez a program elindítása után csatlakozik egy távoli géphez, ellenõrzi a biztonsági kulcsát és letölti a portok legfrissebb változatait. A biztonsági kulcs feladata a frissítés közben letöltött állományok sértetlenségének szavatolása, ezzel gondoskodik róla, hogy az adatok átvitelük közben nem változtak meg. A Portgyûjtemény legújabb változatát így érhetjük el:

```
# portsnap fetch
Looking up portsnap.FreeBSD.org mirrors... 3 mirrors found.
Fetching snapshot tag from portsnap1.FreeBSD.org... done.
Fetching snapshot metadata... done.
Updating from Wed Aug  6 18:00:22 EDT 2008 to Sat Aug 30 20:24:11 EDT 2008.
Fetching 3 metadata patches.. done.
Applying metadata patches... done.
Fetching 3 metadata files... done.
Fetching 90 patches.....10....20....30....40....50....60....70....80....90. done.
Applying patches... done.
Fetching 133 new ports or files... done.
```

A példában látható, hogy a portsnap(8) eltéréseket talált a helyi és a távoli rendszerekben fellelhetõ portok között, majd azokat ellenõrizte. Emellett az is megfigyelhetõ, hogy korábban már futtatuk a programot, mivel ha most indítottuk volna az elsõ alkalommal, akkor egyszerûen letöltötte volna a teljes Portgyûjteményt.

Ahogy a portsnap(8) sikeresen befejezi az imént kiadott `fetch` mûvelet végrehajtását, a helyi rendszeren már telepítésre készen fognak várakozni a Portgyûjtemény és az hozzá tartozó ellenõrzött módosítások. A `portsnap` elsõ használatakor az `extract` parancs segítségével telepíthetjük a frissített állományokat:

```
# portsnap extract
/usr/ports/.cvsignore
/usr/ports/CHANGES
/usr/ports/COPYRIGHT
/usr/ports/GIDs
/usr/ports/KNOBS
/usr/ports/LEGAL
/usr/ports/MOVED
/usr/ports/Makefile
/usr/ports/Mk/bsd.apache.mk
/usr/ports/Mk/bsd.autotools.mk
/usr/ports/Mk/bsd.cmake.mk
...
```

Egy korábban már telepített Portgyûjteményt a `portsnap update` paranccsal tudunk frissíteni:

```
# portsnap update
```

Ezzel lezárult a portok frissítése, innentõl már az aktualizált Portgyûjtemény felhasználásával tetszõlegesen telepíthetõek vagy frissíthetõek az alkalmazások.

A `fetch`, `extract` vagy `update` mûveletek egyetlen parancsba is összefûzhetõek, ahogy ezt az alábbi példában is láthatjuk:

```
# portsnap fetch update
```

Ez a parancs letölti a Portgyûjtemény legfrissebb változatát, majd kitömöríti azt a helyi /usr/ports könyvtárba.

24.4. A dokumentáció frissítése

Az alaprendszer és a Portgyûjtemény mellett a dokumentáció is a FreeBSD operációs rendszer szerves részét képezi. Noha a FreeBSD dokumentációjának legfrissebb változata folyamatosan elérhetõ a FreeBSD honlapjáról, egyes felhasználók ezt csak lassan vagy nem képesek folyamatosan elérni. Szerencsére egy helyi másolat megfelelõ karbantartásával az egyes kiadásokhoz tartozó dokumentáció is frissíthetõ.

24.4.1. A dokumentáció frissítése CVSup használatával

A FreeBSD telepített dokumentációjának forrásai az alaprendszeréhez hasonlóan (lásd Az alaprendszer újrafordítása) a CVSup segítségével frissíthetõek. Ebben a szakaszban megismerhetjük:

- hogyan telepítsük a dokumentáció elõállításához szükséges eszközöket, amelyekkel a forrásokból újra tudjuk generálni a FreeBSD dokumentációját;
- hogyan töltsük le a dokumentáció forrását CVSup segítségével a /usr/doc könyvtárba;
- a dokumentáció elõállításához alkalmazott rendszer milyen beállításokkal rendelkezik, vagyis hogyan korlátozzuk a generálást bizonyos nyelvekre vagy formátumokra.

24.4.2. A CVSup és a dokumentációs eszközök telepítése

Viszonylag sokféle eszközre lesz szükségünk, ha a FreeBSD dokumentációját a forrásokból akarjuk elõállítani. Ezek az segédprogramok nem részei a FreeBSD alaprendszerének, mivel alapvetõen nagyon sok helyet foglalnak el, és leginkább olyan FreeBSD felhasználók számára fontosak, akik folyamatosan a dokumentációval dolgoznak vagy gyakran frissítik azt forrásból.

A feladathoz szükséges összes eszköz elérhetõ a Portgyûjteménybõl. Ebben a FreeBSD Dokumentációs Projekt összeállított egy textproc/docproj nevû portot, amellyel az említett programok telepítését és frissítését igyekezték megkönnyíteni.

Ha nem tartunk igényt a dokumentáció PostScript® vagy PDF változatára, akkor ehelyett inkább érdemes megfontolnunk a textproc/docproj-nojadetex port telepítését. Ebben a változatban a teTeX betûszedõ rendszer kivételével az összes segédprogram megtalálható. Mivel a teTeX önmagában nagyon sok segédeszköz telepítését jelenti, ezért amennyiben a PDF változat ténylegesen nem szükséges, érdemes eltekinteni a telepítésétõl.

A CVSup telepítésével kapcsolatban pedig részletesebb információkat a CVSup használatával foglalkozó szakaszban olvashatunk.

24.4.3. A dokumentáció forrásának frissítése

A /usr/shared/examples/cvsup/doc-supfile konfigurációs állomány segítségével a CVSup képes letölteni a dokumentáció forrásállományainak legfrissebb példányait. Itt a frissítést alapértelmezés szerint egy nem létezõ géptõl fogjuk kérni (mivel ezt kötelezõ kitölteni), azonban a cvsup(1) programnak egy parancssori paraméter segítségével megadhatjuk melyik CVSup szerverrõl töltse le a forrásokat:

```
# cvsup -h cvsup.FreeBSD.org -g -L 2 /usr/shared/examples/cvsup/doc-supfile
```

Ne felejtsük el a *cvsup.FreeBSD.org* helyére beírni a hozzánk földrajzilag legközelebb elhelyezkedõ CVSup szervert. Ezek teljes listáját a CVSup oldalak tartalmazza.

Egy ideig eltarthat, amíg elõször letöltjük a forrásokat. Várjuk meg türelmesen, amíg befejezõdik a

mûvelet.

Késõbb a forrásokat ugyanezzel a paranccsal tudjuk frissíteni. A CVSup ugyanis mindig csak a legutóbbi futtatása óta történt változásokat tölti le, ezért késõbb már ez a lépés jelentõsen felgyorsulhat.

A források letöltése után a dokumentációt például az ekkor keletkezett /usr/doc könyvtárban található Makefile használatával állíthatjuk elõ. Tehát miután az /etc/make.conf állományban beállítottuk a SUP_UPDATE, SUPHOST és DOCSUPFILE változókat, le tudjuk futtatni a következõ parancsot:

```
# cd /usr/doc
# make update
```

Az elõbb említett make(1) változók jellemzõ értékei:

```
SUP_UPDATE= yes
SUPHOST?= cvsup.freebsd.org
DOCSUPFILE?= /usr/shared/examples/cvsup/doc-supfile
```

Mivel a SUPHOST és a DOCSUPFILE változók értékét a ?= szimbólummal állítottuk be, lehetõségünk van a parancssorból ezeknek más értékeket adni. Az /etc/make.conf állományba általában így érdemes felvenni a változókat, így nem kell minden alkalommal módosítani, amikor valamilyen új beállítást akarunk kipróbálni.

24.4.4. A dokumentáció különbözõ beállításai

A FreeBSD dokumentációjához tartozó, frissítést és elõállítást végzõ rendszernek van néhány olyan beállítása, amelyekkel kérhetjük kizárólag csak a dokumentáció egyes részeinek frissítését vagy bizonyos kimeneti formátumok használatát. Ezek vagy globálisan az /etc/make.conf állományban, vagy pedig a parancssorból, a make(1) program paramétereként adhatóak meg.

Ízelítõül néhány közülük:

DOC_LANG
: Az elõállítandó és telepítendõ nyelvû dokumentáció felsorolása, tehát például csak az angol dokumentáció esetén ez en_US.ISO8859-1.

FORMATS
: Az elõállítandó dokumentáció kimeneti formátumainak felsorolása. Itt pillanatnyilag értékként a html, html-split, txt, ps, pdf és rtf jelenhet meg.

SUPHOST
: A frissítéshez használt CVSup szerver hálózati neve.

DOCDIR
: Az elkészült dokumentáció telepítésének helye. Ez alapértelmezés szerint a /usr/shared/doc.

A folyamathoz kapcsolódóan további rendszerszintû make(1) változókról a make.conf(5) man oldalon olvashatunk.

A FreeBSD dokumentációjának elõállításáért felelõs rendszerben használható make(1) további változók bemutatásával kapcsolatban pedig olvassuk el az A FreeBSD Dokumentációs Projekt irányelvei kezdõknek címû könyvet.

24.4.5. A FreeBSD dokumentációjának telepítése forrásból

Miután sikerült letöltenünk a /usr/doc könyvtárba a dokumentáció legfrissebb forrásait, készen állunk a rendszerünkön telepített példány frissítésére.

A DOCLANG értékeként megadott nyelven készült dokumentációkat a következõ paranccsal tudjuk frissíteni:

```
# cd /usr/doc
# make install clean
```

Ha a make.conf állományban korábban már megadtuk a DOCSUPFILE, SUPHOST és SUP_UPDATE változók értékeit, akkor a telepítés fázisa könnyedén össze is vonatható a források frissítésével:

```
# cd /usr/doc
# make update install clean
```

Ha pedig csak bizonyos nyelvekhez tartozó dokumentációt szeretnénk frissíteni, akkor a make(1) akár a /usr/doc könyvtáron belül az egyes nyelvekhez tartozó alkönyvtárakon belül is meghívható, például:

```
# cd /usr/doc/en_US.ISO8859-1
# make update install clean
```

A dokumentáció formátumát a FORMATS változó felhasználásával tudjuk meghatározni:

```
# cd /usr/doc
# make FORMATS='html html-split' install clean
```

24.4.6. A dokumentációs portok használata

Ez elõzõ szakaszban megmutattuk hogyan lehet a FreeBSD dokumentációját a források felhasználásával frissíteni. A források használatával végzett frissítés azonban nem minden FreeBSD rendszer esetében lehetséges vagy hatékony. Ha ugyanis a dokumentációs forrásból akarjuk elõállítani, viszonylag sok eszköz és segédprogram, az ún. *dokumentációs eszközök* használatával kell tisztában lennünk, valamint bizonyos mértékig ismernünk kell a CVS használatát, tudunk kell kikérni a legfrissebb változatot és elõállítatattnunk belõle a végleges változatot. Ezért ebben a szakaszban most szót ejtünk egy olyan módszerrõl, ahol a FreeBSD dokumentációját a

Portgyûjteményen keresztül tudjuk frissíteni, ezáltal:

- anélkül le tudjuk tölteni és telepíteni a dokumentáció adott pillanatban generált változatát, hogy a rendszerünkön bármi további teendõre szükség lenne (ennek köszönhetõen nem kell telepítenünk a dokumentációs eszközöket);
- letölthetjük a dokumentáció forrását és a Portgyûjtemény eszközeivel elõállíthatjuk belõle a megfelelõ változatot (ez a források beszerzésében és feldolgozásában segít valamelyest).

A FreeBSD dokumentáció frissítésének fentebb említett módjait támogatják tehát a *dokumentációs portok*, amelyeket a Documentation Engineering Team <doceng@FreeBSD.org> havi rendszerességgel tart karban. Ezek a portok a FreeBSD Portgyûjteményén belül a docs nevû virtuális kategóriában találhatóak meg.

A dokumentációs portok fordítása és telepítése

A dokumentáció könnyebb elõállításához a dokumentációs portok a Portgyûjtemény lehetõségeit veszik igénybe. Segítségükkel automatikussá teszik a dokumentáció forrásának letöltését, a make(1) parancs meghívását a megfelelõ környezetben, beállításokkal és parancssori paraméterekkel. Rajtuk keresztül a dokumentáció eltávolítása ugyanolyan egyszerûen megtehetõ, mint akármelyik másik FreeBSD port vagy csomag esetében.

Továbbá, amikor a dokumentációs portokat a saját rendszerünkön fordítjuk, a *dokumentációs eszközök* függõségként automatikusan települni fognak.

A dokumentációs portok a következõ módon szervezõdnek:

- Létezik egy ún. "fõport", a misc/freebsd-doc-en, ahol az összes fontosabb állomány megtalálható. Ez lényegében a dokumentációs portok közös õse. Alapértelmezés szerint kizárólag csak az angol nyelvû dokumentációt állítja elõ.
- Létezik egy "mindenes port", a misc/freebsd-doc-all, amely az összes elérhetõ nyelven és formátumban elõállítja a dokumentációt.
- Végezetül minden nyelvhez létezik egy-egy "alport", ilyen például a magyar dokumentáció esetén a misc/freebsd-doc-hu port. Mindegyikük a fõporttól függ és az adott nyelvû dokumentációt telepítik.

Az eddigi összefoglaltaknak megfelelõen a dokumentációs portokat forrásból a következõ paranccsal lehet telepíteni (`root` felhasználóként):

```
# cd /usr/ports/misc/freebsd-doc-en
# make install clean
```

Ennek hatására elõáll és telepítõdik a /usr/local/shared/doc/freebsd könyvtárba az angol nyelvû dokumentáció állományokra bontott HTML formátumban (hasonlóan a http://www.FreeBSD.org tartalmához).

Gyakori beállítások

A dokumentációs portok alapértelmezett viselkedése több különbözõ opció segítségével is befolyásolható. Ezek közül most összefoglalunk néhányat:

`WITH_HTML`

Minden dokumentum egyetlen HTML állományba kerüljön. A végeredmény ekkor az adott dokumentum típusának megfelelõen article.html (cikk) vagy book.html (könyv) néven keletkezik (képekkel együtt).

`WITH_PDF`

Minden dokumentum Adobe® Portable Document Format típusú állományban jön létre. Ezek az állományok a Ghostscript vagy más egyéb PDF nézegetõkkel nyithatóak meg. Ekkor a dokumentáció konkrét típusától függõen az állományok article.pdf (cikk) vagy book.pdf (könyv) néven állítódnak elõ.

`DOCBASE`

A dokumentáció telepítésének helye. Alapértelmezés szerint ez a /usr/local/shared/doc/freebsd könyvtár.

Ügyeljünk arra, hogy a telepítés alapértelmezett célkönyvtára eltér a CVSup módszerétõl. Ugyanis mivel ilyenkor egy portot telepítünk, a tartalma alapértelmezés szerint a /usr/local könyvtáron belülre kerül. Ez azonban a `PREFIX` változó átállításával tetszõleges megváltoztatható.

Az elõbbieket most egy rövid példán keresztül összefoglaljuk. A következõ paranccsal tudjuk tehát a magyar nyelvû dokumentáció Portable Document Format változatát telepíteni:

```
# cd /usr/ports/misc/freebsd-doc-hu
# make -DWITH_PDF DOCBASE=share/doc/freebsd/hu install clean
```

A dokumentációs csomagok használata

A dokumentációs portok elõzõ szakaszban bemutatott forrásból telepítésével kapcsolatban már említettük, hogy szükséges hozzá a dokumentációs eszközök telepítése, valamint némi szabad tárterület. Ha a dokumentációs eszközök telepítéséhez nem elengedõek a rendelkezésre álló erõforrásaink vagy a források feldolgozása túlságosan sokat foglalna a rendszerünkön, akkor lehetõségünk van a dokumentációs portok elõre lefordított, csomagolt változatát használni.

A Documentation Engineering Team <doceng@FreeBSD.org> minden hónapban elõkészíti a FreeBSD dokumentációs csomagok legfrissebb változatát. Az így karbantartott bináris csomagok azután tetszõlegesen használhatóak a szabványos csomagkezelõ eszközökkel, mint amilyen például a pkg_add(1), pkg_delete(1) és így tovább.

A bináris csomagok használata esetén a FreeBSD dokumentációja az adott nyelvhez az *összes* elérhetõ formátumban telepítésre kerül.

Például az alábbi paranccsal a magyar nyelvû dokumentációhoz tartozó legfrissebb bináris

csomagot tudjuk telepíteni:

```
# pkg_add -r hu-freebsd-doc
```

A csomagok elnevezése eltér a hozzá tartozó port nevétõl. Alakja a következõ: `nyelv-freebsd-doc`, ahol a *nyelv* az adott nyelv rövid kódja, vagyis a magyar esetén a `hu`, illetve az egyszerûsített kínai esetén a `zh_ch`.

A dokumentációs portok frissítése

Az elõzetesen telepített dokumentációs portok bármilyen portok frissítésére alkalmas eszközzel frissíthetõek. Például a telepített magyar nyelvû dokumentáció a ports-mgmt/portupgrade eszközön keresztül így frissíthetõ csomagok használatával:

```
# portupgrade -PP hu-freebsd-doc
```

24.5. A fejlesztõi ág követése

A FreeBSD-nek két fejlesztési ága van: a FreeBSD.current és a FreeBSD-STABLE. Ebben a szakaszban mindegyikükrõl monduk pár szót, és megmutatjuk, miként lehet az adott ághoz igazítani a rendszerünk frissítését. Elõször a FreeBSD-CURRENT, majd a FreeBSD-STABLE változata kerül tárgyalásra.

24.5.1. A FreeBSD friss változatának használata

Ahogy arról már az imént is szó esett, nem szabad elfelejtenünk, hogy a FreeBSD-CURRENT a FreeBSD fejlesztésének "frontvonala". Emiatt a FreeBSD-CURRENT használóinak szakmailag jólképzetteknek kell lenniük, és sosem szabad visszariadniuk a használat közben felmerülõ rendszerszintû problémák önálló megoldásától. Ha korábban még nem foglalkoztunk FreeBSD-vel, kétszer is gondoljuk meg a telepítését!

Mi a FreeBSD-CURRENT?

A FreeBSD-CURRENT a FreeBSD mögött álló legfrissebb forráskódot képviseli. Itt találkozhatunk különféle olyan fejlesztés alatt álló részekkel, kísérletezésekkel és átmeneti megoldásokkal, amelyek nem feltétlenül kerülnek bele a szoftver következõ hivatalos kiadásába. Noha a FreeBSD fejlesztõi a FreeBSD-CURRENT forráskódját naponta fordítják, adódhatnak olyan idõszakok, amikor a források mégsem használhatóak maradéktalanul. Az ilyen gondokat általában a lehetõ leggyorsabban igyekeznek megoldani, azonban attól függõen, hogy éppen a forráskód melyik verzióját sikerült kifogni, a FreeBSD-CURRENT használata kész katasztrófa vagy akár a fejlõdésben igazi továbblépés is lehet.

Kinek van szüksége a FreeBSD-CURRENT-re?

A FreeBSD-CURRENT használata elsõsorban az alábbi 3 csoportot érinti:

1. A FreeBSD közösség azon tagjait, akik aktívan dolgoznak a forrásfa valamelyik részén, és mindazokat, akik számára a "legfrissebb" verzió használata feltétlen elvárás.
2. A FreeBSD közösség azon tagjait, akik aktívan tesztelnek, és a FreeBSD-CURRENT kordában tartásához hajlandóak idõt áldozni a menet közben felbukkanó problémák megoldására. Vannak olyanok is, akik a FreeBSD változásaival és fejlesztési irányával kapcsolatban kívánnak javaslatokat tenni, melyeket javítások és módosítások formájában tesznek közzé.
3. Mindazokat, akik pusztán kíváncsiak a fejlesztésben zajló eseményekre, vagy hivatkozási szándékkal töltik le a legfrissebb forrásokat (például csak *nézegetik*, de nem futtatják). Az ilyen emberek esetenként megjegyzéseket fûznek a fejlesztéshez vagy kódot küldenek be.

Mi *nem* a FreeBSD-CURRENT?

1. Az olyan kiadás elõtt álló funkciók kipróbálásának egyszerû módja, amelyekrõl hallottunk, hogy milyen remek újdonságokat hoznak és mi akarunk lenni az elsõk, akik ezt használni is fogják. Ne feledjük azonban, hogy amikor mindenki elõtt kezdünk el használni egy újítást, mi leszünk egyben az elsõk is, akik szembesülnek a benne rejlõ hibákkal.
2. A gyors hibajavítások eszköze. A FreeBSD-CURRENT szinte bármelyik változata pontosan ugyanakkora valószínûséggel hoz magával új hibákat, mint ahogy eltünteti a régieket.
3. Akármilyen értelemben is "hivatalosan támogatott". Képességeinktõl függõen õszintén igyekszünk a lehetõ legtöbbet megtenni a 3 "törvényes" FreeBSD-CURRENT csoportba tartozó emberekért, azonban egyszerûen *nincs idõnk* komolyabb segítségnyújtást adni. Ez viszont nem azt jelenti, hogy komisz és fukar emberek vagyunk, akik utálnak segíteni a másiknak (de máskülönben nem tudna fejlõdni a FreeBSD). Csupán a FreeBSD fejlesztése *közben* fizikailag képtelenek vagyunk a naponta érkezõ ezernyi üzenetet rendre megválaszolni! A FreeBSD elõremozdítása és a kísérleti stádiumban álló kóddal kapcsolatos kérdések megválaszolása közül a fejlesztõk általában az elsõt részesítik elõnyben.

A FreeBSD-CURRENT használata

1. Iratkozzunk fel az FreeBSD-CURRENT levelezési lista és Az src fa head/-current ágának SVN commit üzenetei listákra. Ez nem egyszerûen hasznos, hanem *elengedhetetlen*. Ha nem vagyunk a *FreeBSD-CURRENT levelezési lista* listán, akkor nem fogjuk látni a rendszer aktuális állapotára vonatkozó megjegyzéseket, és így esetleg feleslegesen öljük az idõnket olyan problémák megoldásába, amelyeket mások már korábban megoldottak. Ami viszont ennél is fontosabb, hogy így elszalasztjuk a rendszerünk folyamatos életbentartására vonatkozó létfontosságú bejelentéseket.

   Az Az src fa head/-current ágának SVN commit üzenetei listán láthatjuk az a forráskód egyes változtatásaihoz tartozó naplóbejegyzéseket, a hozzájuk tartozó esetleges mellékhatások ismertetésével együtt.

   A listákra vagy a https://lists.freebsd.org oldalon található többi lista valamelyikére úgy tudunk feliratkozni, ha rákattintunk a nevére. A további lépésekrõl ezt követõen itt kapunk értesítést. Amennyiben a teljes forrásfa változásai érdekelnek minket, javasoljuk az A teljes src fa SVN commit üzenetei (kivéve "user" és "projects") lista olvasását.

2. A tükrözések egyikérõl töltsük le a FreeBSD forrását. Erre két mód is kínálkozik:

a. Használjuk a cvsup programot a /usr/shared/examples/cvsup könyvtárban található standard-supfile állománnyal. Ez a leginkább ajánlott módszer, hiszen így csak egyszer kell letölteni az egész gyûjteményt, majd ezután már csak a változásokat. Sokan a `cvsup` parancsot a `cron` parancson keresztül adják ki, és ezzel mindig automatikusan frissítik a forrásaikat. A cvsup mûködését a fentebb említett minta supfile állomány megfelelõ módosításával tudjuk a saját környezetünkhöz igazítani.

Az említett standard-supfile állomány eredetileg nem a FreeBSD-CURRENT, hanem inkább a FreeBSD biztonsági problémáit érintõ javítások követésére használatos. A FreeBSD-CURRENT forrásainak eléréséhez a következõ sort kell kicserélnünk ebben az állományban:

```
*default release=cvs tag=RELENG_X_Y
```

Erre:

```
*default release=cvs tag=.
```

A `tag` paramétereként megadható egyéb címkékrõl a kézikönyv CVS címkék szakaszában olvashatunk.

b. Használjuk a CTM alkalmazás nyújtotta lehetõségeket. Amennyiben nagyon rossz netkapcsolattal rendelkezünk (drága vagy csak levelezésre használható) a CTM megoldást jelenthet számunkra. Legyünk azonban tekintettel arra, hogy helyenként zûrös lehet a használata és néha hibás állományokat gyárt. Emiatt viszont csak ritkán használják, így elõfordulhat, hogy hosszabb ideig nem is mûködik. A 9600 bps vagy annál nagyobb sebességû kapcsolatok esetén ezért inkább a CVSup használatát javasoljuk.

3. Ha nem csak böngészésre, hanem fordításra is szedjük a forrásokat, mindig töltsük le a FreeBSD-CURRENT *egészét*, ne csak egyes részeit. Ez azzal magyarázandó, hogy a forráskód bizonyos részei más helyeken található részektõl is függenek, és ezért az önálló fordításuk szinte garantáltan gondot fog okozni.

   A FreeBSD-CURRENT lefordítása elõtt figyelmesen olvassuk át a /usr/src könyvtárban található Makefile állományt. A frissítési folyamat részeként elõször mindenképpen érdemes telepíteni egy új rendszermagot és újrafordítani az alaprendszert. Olvassuk el a FreeBSD-CURRENT levelezési lista üzeneteit és a /usr/src/UPDATING állományt, ahol megtalálhatjuk az ezzel kapcsolatos legújabb információkat, melyek egy-egy újabb kiadás közeledtével egyre fontosabbá válnak.

4. Foglalkozzunk vele! Ha már a FreeBSD-CURRENT változatát használjuk, ne legyünk restek véleményt formálni róla, különösen abban az esetben, ha továbbfejlesztésekrõl vagy hibákra van szó. Leginkább a forráskóddal együtt érkezõ javaslatoknak szoktak örülni a fejlesztõk!

24.5.2. A FreeBSD stabil változatának használata

Mi a FreeBSD-STABLE?

A FreeBSD-STABLE az a fejlesztési ág, ahonnan az egyes kiadások származnak. Ebbe az ágba már más ütemben kerülnek a változások, mivel általánosan elfogadott, hogy ide a korábban már kipróbált módosítások vándorolnak át a FreeBSD-CURRENT ágból. Ez azonban *még mindig* csak egy fejlesztési ág, ami arra utal, hogy a FreeBSD-STABLE által adott pillanatban képviselt források nem feltétlenül felelnek meg bizonyos célokra. Ez csupán egy újabb fejlesztési nyomvonal, nem pedig a végfelhasználók kenyere.

Kinek van szüksége a FreeBSD-STABLE-re?

Ha szeretnénk figyelemmel kísérni vagy valamilyen módon kiegészíteni a FreeBSD fejlesztési folyamatát, különösen a FreeBSD következõ "nagyobb" kiadását illetõen, akkor érdemes követnünk a FreeBSD-STABLE forrásait.

Habár a FreeBSD-STABLE ágba is bekerülnek a biztonsági jellegû javítások, ettõl még nem kell feltétlenül ezt követnünk. A FreeBSD-hez kiadott biztonsági figyelmeztetések mindig leírják, hogyan kell javítani a hibát az érintett kiadásokban , azonban az egész fejlesztési ágat felesleges csak biztonsági okból kifolyólag követni, mivel így olyan változások is kerülhetnek a rendszerbe, amire nincs szükségünk.

Habár igyekszünk gondoskodni a FreeBSD-STABLE ágban található források lefordíthatóságáról és mûködõképességérõl, nem minden esetben szavatolható. Ráadásul mivel a FreeBSD-STABLE ágba kerülõ kódokat elõször a FreeBSD-CURRENT ágban fejlesztik ki, és mivel a FreeBSD-STABLE felhasználói többen vannak a FreeBSD-CURRENT változaténál, ezért szinte elkerülhetetlen, hogy ilyenkor a FreeBSD-STABLE változatban bizonyos hibák és szélsõséges esetek be ne következzenek, amelyek a FreeBSD-CURRENT használata során még nem buktak ki.

Ezért a FreeBSD-STABLE ág vakon követését senkinek *sem* ajánljuk, és különösen fontos, hogy éles szervereken elõzetes kimerítõ tesztelések nélkül ne futassunk FreeBSD-STABLE rendszert.

Ha ehhez nem rendelkezünk elegendõ erõforrással, akkor egyszerûen használjuk a FreeBSD legfrissebb kiadását, és az egyes kiadások között pedig bináris frissítéssel közlekedjünk.

A FreeBSD-STABLE használata

1. Iratkozzunk fel a FreeBSD-STABLE; levelezési lista listára. Ezen keresztül értesülhetünk a FreeBSD-STABLE használata során felmerülõ fordítási függõségekrõl vagy más, külön figyelmet igénylõ problémákról. Gyakran ezen a levelezési listán elmélkednek a fejlesztõk a vitatott javításokról vagy frissítésekrõl, amibe a felhasználók is beleszólhatnak, ha a szóbanforgó változtatással kapcsolatban bármilyen problémájuk vagy ötletünk van.

   Iratkozzunk fel a követni kívánt ághoz tartozó SVN levelezési listára. Például ha a 7-STABLE ág változásait követjük, akkor az svn-src-stable-7 listára érdemes feliratkoznunk. Ennek segítségével elolvashatjuk az egyes változtatásokhoz tartozó naplóbejegyzéseket, a rájuk vonatkozó esetleges mellékhatások ismertetésével együtt.

   Ezekre, valamint a https://lists.freebsd.org címen elérhetõ listák valamelyikére úgy tudunk feliratkozni, ha a nevükre kattintunk. A további teendõk ezután itt jelennek meg.

2. Amennyiben egy új rendszert akarunk telepíteni és a FreeBSD-STABLE havonta készült pillanatképeit akarjuk rajta futtatni, akkor erről bővebb felvilágosítást a Pillanatképek honlapján találhatunk (angolul). Emellett a legfrissebb FreeBSD-STABLE kiadást telepíthetjük a tükrözések valamelyikéről is, majd innen a lentebb található utasítások szerint tudunk hozzáférni a FreeBSD-STABLE forráskódjának legfrissebb változatához.

   Ha már fut a gépünkön a FreeBSD egy korábbi kiadása, és ezt akarjuk forráson keresztül frissíteni, akkor ezt a FreeBSD tükrözéseivel könnyedén megtehetjük. Két módon is: .. Használjuk a cvsup programot a /usr/shared/examples/cvsup könyvtárból származó stable-supfile állománnyal. Ez a leginkább ajánlott módszer, mivel így csak egyszer kell letölteni a teljes gyűjteményt, utána már csak a hozzá tartozó változtatásokra van szükségünk. A `cvsup` parancsot sokan a `cron` segítségével futtatják, és ezzel automatikusan frissülnek a forrásainak. A cvsup működését környezetünkhöz az előbb említett minta supfile megfelelő módosításával tudjuk behangolni. .. Használjuk a CTM programot. Ha nincs olcsó vagy gyors internetkapcsolatunk, akkor érdemes ezt a módszert választani.

3. Alapvetően azonban ha gyorsan szeretnénk hozzájutni a forrásokhoz és a sávszélesség nem meghatározó tényező, akkor helyette válasszuk a `cvsup` vagy az `ftp` használatát, és csak minden más esetben CTM-et.

4. Mielőtt lefordítanánk a FreeBSD-STABLE változatát, figyelmesen olvassuk át a /usr/src könyvtárban levő Makefile állományt. Az átállási folyamat részeként először minden bizonnyal telepítenünk kell egy új rendszermagot és újra kell fordítanunk az alaprendszert. A FreeBSD-STABLE; levelezési lista valamint a /usr/src/UPDATING elolvasásából értesülhetünk azokról az egyéb, gyakran nagyon fontos változásokról, melyek elengedhetetlenek lesznek a következő kiadás használatához.

24.6. A forrás szinkronizálása

Az internet (vagy elektronikus levelek) használatán keresztül számos mód kínálkozik az FreeBSD Projekthez tartozó források frissen tartásához egy adott, vagy éppen az összes területen attól függően, hogy mik érdekelnek minket. Ehhez elsősorban az Anonim CVS, CVSup és CTM szolgáltatásokat ajánljuk fel.

Habár lehetséges csupán a forrásfa egyes részeit letölteni, a támogatott frissítési eljárás során azonban szükségünk lesz az egész fa szinkronizálására és a rendszerhez tartozó felhasználói programok (vagyis minden olyan program, amely a felhasználói térben fut, ilyeneket találhatunk többek közt a /bin és /sbin könyvtárakban) valamint rendszermag újrafordítására is. Ha csak a felhasználói programok forrásait, vagy csak a rendszermagot, esetleg csupán a forrásfa egyes részeit frissítjük, akkor az gondokat okozhat. Az itt előforduló problémák fordítási hibáktól kezdve rendszerösszeomlásokon keresztül akár adatvesztésbe is torkollhatnak.

Az Anonim CVS és a CVSup alkalmazások ún. *lehúzással* frissítik a forrásokat. A CVSup használatakor a felhasználó (vagy a `cron` szkript) meghívja a `cvsup` programot, amely az állományok aktualizálásához felveszi a kapcsolatot egy máshol megtalálható `cvsupd` szerverrel. Az így nyert frissítések az adott pillanatig visszemenőleg érkeznek meg, de csak akkor, ha igényeljük ezeket. A

frissítést könnyedén le tudjuk szabályozni a számunkra érdekes egyes állományokra és könyvtárakra. A frissítéseket a szerver hozza létre menet közben annak megfelelõen, hogy milyen verziókkal rendelkezünk, és mihez akarunk szinkronizálni. Az Anonim CVS a CVSupnál valamivel egyszerûbb abban a tekintetben, hogy ez a CVS-nek egy olyan kiterjesztése, amely lehetõvé teszi a változtatások közvetlen lehúzását egy távoli CVS tárházból. Miközben a CVSup mindezt sokkal hatékonyabb valósítja meg, addig az Anonim CVS jóval könnyebben használható.

Velük szemben a CTM nem hasonlítja össze interaktívan a saját és a központi szerveren tárolt forrásokat és nem is húzza át ezeket. Ehelyett egy olyan szkriptõl van szó, amely naponta többször megvizsgálja a központi CTM szerveren tárolt állományok a legutóbbi futtatás óta keletkezett változtatásait, majd az észlelt módosulásokat betömöríti, felcímkézi egy sorozatszámmal és (nyomtatható ASCII formátumban) elõkészíti ezeket az e-mailen keresztüli küldésre. Az így létrehozott "CTM delták" megérkezésük után a ctm_rmail(1) segédprogrammal kerülnek feldolgozásra, amely magától visszaalakítja, ellenõrzi és alkalmazza a változtatásokat a forrásfa felhasználó birtokában levõ másolatára. Ez a megoldás hatékonyabb a CVSup használatánál, mert kisebb terhelést jelent a szerverek számára, hiszen a frissítéshez nem a *lehúzást*, hanem a *küldést* alkalmazzák.

Természetesen minden említett eljárásnak megvannak a maga kompromisszumai. Ha véletlenül kitöröljük a forrásfánk egyes részeit, a CVSup képes ezt észrevenni és helyreállítani a sérült részeket. A CTM ezzel szemben ezt nem végzi el, szóval ha (biztonsági mentés nélkül) letöröljük a forrásainkat, akkor az egész szinkronizálást az elejérõl kell kezdenünk (pontosabban a legfrissebb CVS-es "alapdeltától") és a CTM-mel újraépíteni az egészet, esetleg a Anonim CVS-sel letörölni a hibás adatokat és újraszinkronizálni.

24.7. Az alaprendszer újrafordítása

Miután sikerült a helyi forrásfánkat a FreeBSD egy nekünk szimpatikus (FreeBSD-STABLE, FreeBSD-CURRENT és így tovább) változatához igazítanunk, elérkezett az idõ, hogy a segítségével újrafordítsuk az egész rendszert.

Készítsünk biztonsági mentést

Nem tudjuk eléggé nyomatékosítani, hogy *mielõtt* nekikezdenénk, készítsünk egy biztonsági mentést a rendszerünkrõl. Míg az alaprendszer újrafordítása nem túlságosan bonyolult feladat (egészen addig, amíg a megadott utasításokat követjük), saját magunk vagy mások hibájából fakadóan kialakulhatnak olyan helyzetek, amikor a rendszer nem lesz képes elindulni.

Mindenképpen gyõzödjünk meg róla, hogy tisztességesen elvégeztük a mentést és akad a kezünk ügyében egy javításra felhasználható rendszerindító floppy vagy CD. Valószínûleg soha nem lesz ténylegesen szükségünk rájuk, azonban jobb félni, mint megijedni!

Iratkozzunk fel a megfelelõ levelezési listákra

A FreeBSD-STABLE és FreeBSD-CURRENT ágak természetüknél fogva *fejlesztés alatt állnak*. A FreeBSD fejlesztését is emberek végzik, ezért elõfordulhatnak benne tévedések.

Ezek a tévedések gyakran csak ártalmatlan apróságok, amelyek hatására kapunk például egy ismeretlen diagnosztikai hibát. De ezzel szemben létrejöhetnek pusztító erejû hibák is, amelyek hatására a rendszerünk nem lesz képes elindulni, károsodnak az állományrendszerek (vagy még rosszabb).

Ha ilyen történik, akkor egy "felszólítást" (egy "heads up" témájú üzenetet) küldenek az érintett változatokhoz tartozó listákra, amelyben igyekeznek kifejteni a probléma természetét és a rendszerre mért hatását. Miután "minden rendbejött", a probléma megoldásáról is küldenek egy értesítést.

Ha a FreeBSD-STABLE; levelezési lista vagy a FreeBSD-CURRENT levelezési lista olvasása nélkül próbáljuk meg használni a FreeBSD-STABLE és FreeBSD-CURRENT verziókat, akkor csak magunknak keressük a bajt.

Ne használjuk a `make world` parancsot

Rengeteg régebben készült dokumentáció erre a feladatra a `make world` parancs kiadását javasolja. Ennek használatával azonban átlépünk olyan fontos lépéseket, amelyek valójában csak akkor lennének kihagyhatóak, ha pontosan tudjuk mit csinálunk. Ezért az esetek döntõ többségében nem a `make world` használatára van szükségünk, hanem a most bemutatandó eljárásra.

24.7.1. A rendszer frissítése dióhéjban

A frissítés megkezdése elõtt érdemes elolvasnunk a /usr/src/UPDATING állományt, ahol a letöltött források használatához elvégzendõ elõzetes intézkedésekrõl kaphatunk hírt. Ezután kövessük az alábbiakban körvonalazott módszer egyes lépéseit.

Ezek a lépések feltételezik, hogy egy korábbi FreeBSD verziót használunk, tehát a fordító, a rendszermag, az alaprendszer és a konfigurációs állományok valamelyik régebbi változatát. Alaprendszer alatt, amelyet sokszor csak a "world" néven hivatkozunk, a rendszer számára alapvetõ fontosságú binárisokat, programkönyvtárakat és programfejlesztéshez szükséges egyéb állományokat értjük. Maga a fordítóprogram is része ennek, azonban tartalmaz néhány speciális megszorítást.

Mindezek mellett továbbá feltételezzük, hogy elõzetesen már valamilyen módon letöltöttük a friss forrásokat. Ha rendszerünkön ezt még nem tettük volna meg, akkor a A forrás szinkronizálása segítségével tájékozódhatunk részletesen arról, hogyan tölthetjük le a legfrissebb verziót.

A rendszer forráskódon keresztüli frissítése egy kicsivel körülményesebb, mint amennyire elsõre látszik. A FreeBSD fejlesztõk az évek során fontosnak találták, hogy a folyamatosan felszínre bukkanó, elkerülhetetlen függõségek tükrében meglehetõsen drámai módon megváltoztassák az erre javasolt módszert. Ezért a szakasz további részében a pillanatnyilag javasolt frissítési megoldás nyomán fogunk haladni.

A sikeres frissítések során az alábbi akadályokkal kell mindenképpen szembenéznünk:

- A fordító régebbi változata nem feltétlenül lesz képes lefordítani az új rendszermagot. (Illetve a régebbi fordítóprogramok tartalmazhatnak hibákat.) Ezért az új rendszermagot már a fordító új változatával kell elõállítanunk. Ebbõl következik, hogy az új rendszermag elkészítéséhez elõször

a fordítóprogram újabb változatát kell lefordítanunk. Ez viszont nem feltétlenül jelenti azt, hogy az új rendszermag fordítása elõtt az új fordítóprogramot *telepítenünk* is kellene.

- Az új alaprendszer esetenként bizonyos új funkciókat igényelhet a rendszermagtól. Ezért a frissebb alaprendszer telepítése elõtt telepítenünk kell a frissebb rendszermagot.
- Ez az elõbb említett két akadály képzi az okát a következõ bekezdésekben bemutatott `buildworld`, `buildkernel`, `installkernel`, `installworld` sorozatnak. Természetesen léteznek további egyéb indokok is, amiért még érdemes az itt leírtak szerint frissíteni a rendszerünket. Ezek közül most vegyünk néhány kevésbé nyilvánvalóbbat:
  - A régebbi alaprendszer nem minden esetben fog problémamentesen együttmûködni az új rendszermaggal, ezért az alaprendszer újabb változatát szinte azonnal az új rendszermagot követõen kell telepítenünk.
  - Vannak olyan konfigurációs változtatások, amelyeket még az új alaprendszer telepítése elõtt el kell végeznünk, a többi viszont veszélyes lehet a korábbi alaprendszerre. Ezért a konfigurációs állományokat általában két külön lépésben kell frissíteni.
  - A frissítés során nagyrészt csak állományok cserélõdnek el és újabbak érkeznek, a korábbiak nem törlõdnek. Ez bizonyos esetekben azonban gondokat okozhat. Ennek eredményeképpen a frissítés során idõnként elõfordulhat, hogy magunknak kell manuálisan némely megadott állományokat törölnünk. Elképzelhetõ, hogy ezt a jövõben még majd automatizálni fogják.

Ezek a megfontolások vezettek tehát az ismertetendõ eljárás kialakításához. Ettõl függetlenül adódhatnak olyan helyzetek, amikor további lépéseket is be kell iktatnunk, viszont az itt bemutatott folyamat egy ideje már viszonylag elfogadottnak tekinthetõ:

a. `make buildworld`

   Elõször lefordítja az új fordítóprogramot és néhány hozzá tartozó eszközt, majd ennek felhasználásával elkészíti az alaprendszer többi részét. Az eredmény a /usr/obj könyvtárban keletkezik.

b. `make buildkernel`

   Eltérõen a config(8) és make(1) programok korábban javasolt alkalmazásától, ezzel a paranccsal már a /usr/obj könyvtárban létrehozott *új* fordítót használjuk. Ez védelmet nyújt a fordító és rendszermag változatai közti eltérésekbõl fakadó problémák ellen.

c. `make installkernel`

   Telepíti a lemezre az új rendszermagot és a hozzá tartozó modulokat, ezáltal lehetõvé válik a frissített rendszermag betöltése.

d. Átváltás egyfelhasználós módba.

   Egyfelhasználós módban a minimálisra csökkenthetjük a futó szoftverek frissítésébõl adódó bonyodalmakat. Ezzel együtt minimálissá válik a régi alaprendszer és az új rendszermag eltéréseibõl eredõ problémák elõfordulása is.

e. `mergemaster -p`

   Az új alaprendszer telepítéséhez elvégzi a konfigurációs állományok részérõl szükséges

frissítéseket. Például felvesz még nem létezõ csoportokat vagy felhasználókat. Ez gyakran elengedhetetlennek bizonyulhat, mivel ha a rendszer legutóbbi frissítése óta újabb csoportok vagy felhasználók kerültek be az alaprendszerbe, a `installworld` csak akkor tud hibamentesen lefutni, ha ezek már a futásakor is elérhetõek.

f. `make installworld`

   Átmásolja a /usr/obj könyvtárból a korábban elkészített új alaprendszert. Lefutása után már mind az új rendszermag és az új alaprendszer a megfelelõ helyén található.

g. `mergemaster`

   Feldolgozzuk a korábbi fázisból fennmaradó konfigurációs állományok frissítését, mivel most már elérhetõ az új alaprendszer.

h. A rendszer újraindítása.

   Az új rendszermag és az új konfigurációs állományokkal futó alaprendszer használatához teljesen újra kell indítanunk a számítógépünket.

   Ha a FreeBSD ugyanazon fejlesztési ágán belül frissítjük a rendszerünket, például a 7.0 kiadásról a 7.1 kiadásra, akkor értelemszerûen nem kell az iménti eljárás minden lépését szorosan követni, hiszen nagyon valószínûtlen, hogy komoly eltérések lennének a fordítóprogram, a rendszermag, az alaprendszer és a konfigurációs állományok között. Ilyenkor akár nyugodtan kiadhatjuk a `make world` parancsot, majd kérhetjük a rendszermag fordítását és telepítését.

   A fejlesztési ágak közti váltás során azonban könnyen érhetnek minket meglepetések, ha nem a megadottak szerint járunk el.

   Egyes váltásokhoz (például 4.*X* és 5.0 között) további lépések megtétele is szükséges lehet (például adott állományok törlése vagy átnevezése még az `installworld` elõtt). Ilyenkor mindig figyelmesen olvassuk át a /usr/src/UPDATING állományt, különös tekintettel a végére, mivel gyakran ott adják meg a konkrét verzióváltáshoz szükséges teendõket.

   A szakaszban összefoglalt lépések egyfajta evolúciós folyamat eredményei, melynek során a fejlesztõk felismerték, hogy nem tökéletesen kivédeni az összes frissítéssel járó problémát. A javasolt eljárás remélhetõleg viszont még sokáig érvényes marad.

   A FreeBSD 3.*X* vagy annál is korábbi változatok frissítése még ennél is több ügyességet kíván. Ha ilyen verziót akarunk frissíteni, akkor feltétlenül olvassuk el az UPDATING állományt!

   Röviden tehát a FreeBSD forráskódon keresztüli frissítését így foglalhatjuk össze:

```
# cd /usr/src
# make buildworld
# make buildkernel
# make installkernel
```

```
# shutdown -r now
```

Néhány ritka esetben a `buildworld` lépés elõtt szükségünk lehet a `mergemaster -p` parancs lefuttatására is. Errõl az UPDATING állományból tudakozódhatunk. Általában azonban nyugodt szívvel kihagyhatjuk ezt a lépést, kivéve, ha nem egy vagy több fõbb FreeBSD változatot átívelõ frissítést végzünk.

Miután az `installkernel` sikeresen befejezte a munkáját, indítsuk újra a számítógépet egyfelhasználós módban (a betöltõ parancssorában adjuk ki `boot -s` parancsot). Itt futtassuk a következõket:

```
# adjkerntz -i
# mount -a -t ufs
# mergemaster -p
# cd /usr/src
# make installworld
# mergemaster
# reboot
```

Olvassuk el a magyarázatokat

Az iménti leírt folyamat csupán rövid összefoglalás, amivel némi gyorstalpalást igyekeztünk adni. Az egyes lépések megértéséhez azonban javasolt átolvasni a most következõ szakaszokat is, különösen abban az esetben, ha saját rendszermagot akarunk használni.

24.7.2. Nézzük meg a /usr/src/UPDATING állományt

Mielõtt bármihez is nekifognánk, keressük meg a /usr/src/UPDATING (vagy hasonló, a forráskód másolatunk tényleges helyétõl függõ) állományt. Ebben adják hírül az esetlegesen felmerülõ problémákra vonatkozó fontosabb információkat, vagy határozzák meg az egyes lefuttatandó parancsok pontos sorrendjét. Amennyiben az UPDATING ellentmondana az itt olvasottaknak, az UPDATING tartalma a mérvadó.

A korábban tárgyaltak szerint az UPDATING elolvasása nem helyettesíti a megfelelõ levelezési listák figyelemmel kísérését. Ez a két elvárás nem kizárja, hanem kiegészíti egymást.

24.7.3. Ellenõrizzük az /etc/make.conf állományt

Vizsgáljuk át a /usr/shared/examples/etc/make.conf és az /etc/make.conf állományokat. Az elõbbi tartalmaz néhány alapértelmezett beállítást - ezek javarészét megjegyzésbe rakták. Ha használni akarjuk a rendszer lefordítása során, tegyük bele ezeket az /etc/make.conf állományba. Ne felejtsük el azonban, hogy minden, amit megadunk az /etc/make.conf állományba, a `make` minden egyes elindításakor felhasználásra kerül. Éppen ezért olyanokat érdemes itt beállítani, amik az egész rendszerünket érintik.

A legtöbb felhasználó számára az /etc/make.conf állományhoz a /usr/shared/examples/etc/make.conf állományban található `CFLAGS` és `NO_PROFILE` sorokra lesz szüksége, melyeket kivehetünk a megjegyzésbõl.

A többi definíció (`COPTFLAGS`, `NOPORTDOCS` és így tovább) használatáról már mindenki maga dönt.

24.7.4. Frissítsük az /etc tartalmát

Az /etc könyvtár tartalmazza a rendszer beállításaival kapcsolatos információk jelentõs részét, valamint a rendszer indítása során lefutó szkripteket. Egyes szkriptek a FreeBSD verzióiról verzióira változnak.

Némely konfigurációs állományok a rendszer hétköznapi mûködésében is szerepet játszanak. Ilyen például az /etc/group.

Alkalmanként a `make installworld` parancs futása során igényt tart adott nevû felhasználókra és csoportokra. A frissítéskor azonban ezek a felhasználók vagy csoportok nem feltétlenül állnak rendelkezésre, ami gondokat okozhat. Ezért bizonyos esetekben a `make buildworld` elõzetesen ellenõrzi az igényelt felhasználók és csoportok meglétét.

Erre például szolgálhat a `smmsp` felhasználó esete. Nélküle a felhasználók nem tudták telepíteni az új rendszert, mert hiányában az mtree(8) nem volt képes létrehozni a /var/spool/clientmqueue könyvtárat.

Ezt úgy lehetett megoldani, hogy még az alaprendszer lefordítása (a `buildworld`) elõtt meg kellett hívni a mergemaster(8) parancsot a `-p` paraméterrel. Így csak azokat az állományokat fogja összehasonlítani, amelyek feltétlenül szükségesek a `buildworld` vagy az `installworld` sikeres mûködéséhez. Amennyiben a `mergemaster` egy olyan verziójával rendelkezünk, amely nem ismeri a `-p` paramétert, akkor az elsõ indításakor használjuk a forrásfában található újabb verzióját:

```
# cd /usr/src/usr.sbin/mergemaster
# ./mergemaster.sh -p
```

Ha különösen paranoiásak vagyunk, akkor a csoport törlése vagy átnevezése elõtt az alábbi paranccsal ellenõrizni tudjuk az általa birtokolt állományokat:

```
# find / -group GID -print
```

Ez megmutatja *GID* (mely megadható numerikus vagy név formájában is) jelzésû csoporthoz tartozó összes állományt a rendszerünkben.

24.7.5. Váltsunk egyfelhasználós módba

A rendszert egyfelhasználós módban érdemes lefordítani. A nyilvánvalóan érezhetõ gyorsaság elõnyei mellett azért is jobban járunk, mert az új rendszer telepítése során számos rendszerszintû állomány is módosításra kerül, beleértve a szabványos rendszerszintû binárisokat, függvénykönyvtárakat, include állományokat és így tovább. Ha üzemelõ rendszeren végezzük el

mindezen változtatásokat (különösen amikor rajtunk kívül még további felhasználók is tartózkodnak a rendszerben), az csak a bajt hozza ránk.

Másik lehetõség gyanánt a rendszert magát lefordíthatjuk többfelhasználós módban is, majd ezután csak a telepítést hajtjuk végre egyfelhasználós üzemmódban. Ha eszerint cselekszünk, egyszerûen várjunk addig, amíg az összes fordítás be nem fejezõdik, és az egyfelhasználósra váltást halasszuk a `installkernel` vagy `installworld` idejére.

Egy mûködõ rendszerben rendszeradminisztrátorként az alábbi parancs kiadásával válthatunk át egyfelhasználós módba:

```
# shutdown now
```

Ezt elérhetjük úgy is, ha újraindítjuk a rendszert és a rendszer indításakor a "single user" pontot választjuk a menübõl. Ekkor a rendszer egyfelhasználós módban indul el. Miután ez megtörtént, adjuk ki a következõ parancsokat:

```
# fsck -p
# mount -u /
# mount -a -t ufs
# swapon -a
```

Ezekkel a parancsokkal elõször ellenõrizzük az állományrendszereket, ezután újracsatlakoztatjuk a / állományrendszert írható módban, csatlakoztatjuk az /etc/fstab állományban megadott összes többi UFS típusú állományrendszert, majd bekapcsoljuk a lapozóállomány használatát.

Ha a gépünk óráját nem a greenwich-i, hanem a helyi idõ szerint állítottuk be (ez akkor áll fenn, ha a date(1) parancs nem a helyes idõt és idõzónát jelzi ki), akkor még erre is szükségünk lehet:

```
# adjkerntz -i
```

Ezzel a helyi idõzóna beállításait tudjuk jól beállítani - nélküle késõbb még gondjaink akadhatnak.

24.7.6. Töröljük a /usr/obj könyvtárat

A rendszer egyes részei fordításuk során a /usr/obj könyvtáron belülre kerülnek (alapértelmezés szerint). Az itt található könyvtárak a /usr/src könyvtárszerkezetét követik.

Ha mindenestõl töröljük ezt a könyvtárat, akkor növeli tudjuk a `make buildworld` folyamat sebességét és megmenekülünk néhány függõségekkel kapcsolatos fejfájástól is.

Egyes /usr/obj könyvtáron belüli állományoknál szerepelhet a "megváltoztathatatlan" (immutable) állományjelzõ (lásd chflags(1)), amelyet a mûvelet elvégzéséhez elõször el kell távolítanunk.

```
# cd /usr/obj
# chflags -R noschg *
# rm -rf *
```

24.7.7. Fordítsuk újra az alaprendszert

A kimenet elmentése

Jól járunk azzal, ha a make(1) futásának kimenetét elmentjük egy állományba, mivel így a hibák esetén lesz egy másolatunk a hibaüzenetrõl. Ha konkrétan nekünk nem is feltétlenül segít megtalálni a hiba tényleges okát, mások viszont többet tudnak róla mondani, ha beküldjük ezt a FreeBSD egyik levelezési listájára.

Ezt egyébként a legegyszerûbben a script(1) parancs segítségével oldhatjuk meg, amelynek paraméteréül azt az állományt kell megadni, ahova menteni akarjuk a kimenetet. Ezt közvetlenül a rendszer újrafordítása elõtt kell kiadnunk, majd miután megállt, a `exit` paranccsal kiléphetünk belõle.

```
# script /var/tmp/mw.out
Script started, output file is /var/tmp/mw.out
# make TARGET
... fordít, fordít, fordít ...
# exit
Script done, ...
```

Ilyenkor *soha ne* a /tmp könyvtárba mentsük a kimenetet, mert ennek a tartalma a következõ indítás során magától törlõdik. Sokkal jobban tesszük, ha a /var/tmp könyvtárba (ahogy tettük azt az elõbbi példában is) vagy a `root` felhasználó könyvtárába mentünk.

Az alaprendszer fordítása

A /usr/src könyvtárban kell állnunk:

```
# cd /usr/src
```

(kivéve természetesen, ha máshol van a forráskód, akkor abba a könyvtárba menjünk).

Az alaprendszert a make(1) paranccsal fordíthatjuk újra. Ez a Makefile nevû állományból olvassa be a FreeBSD programjainak újrafordítását leíró utasításokat, a fordításuk sorrendjét és így tovább.

A begépelendõ paranccsor általános alakja tehát a következõképpen néz ki:

```
# make -x -DVÁLTOZÓ target
```

A fenti példában a `-x` egy olyan a paraméter, amelyet a make(1) programnak adunk át. A make(1)

man oldalán megtalálhatjuk az összes neki átadható ilyen beállítást.

A `-D_VÁLTOZÓ_` alakú paraméterek közvetlenül a Makefile állománynak adnak át olyan változókat, amelyek segítségével vezérelhetõ a viselkedése. Ezek ugyanazok a változók, mint amelyek az /etc/make.conf állományban is szerepelnek, és itt a beállításuk egy másik módját kapjuk. Így a

```
# make -DNO_PROFILE target
```

paranccsal is megadhatjuk, hogy ne profilozott függkönyvtárak jöjjenek létre, ami pontosan megfelel a

```
NO_PROFILE=    true     #    Avoid compiling profiled libraries
```

sornak az /etc/make.conf állományban.

A *target* árulja el a make(1) programnak, hogy mi a teendõje. Minden egyes Makefile különbözõ "targeteket" definiál, és a kiválasztott target mondja meg, pontosan mi is fog történni.

Egyes targetek ugyan megjelennek a Makefile állományban, azonban nem feltétlenül hivatkozhatunk rájuk közvetlenül. Ehelyett csupán arra valók, hogy a fordítás folyamatának lépéseit felbontsák még kisebb allépésekre.

A legtöbb esetben azonban semmilyen paramétert nem kell átadnunk a make(1) parancsnak, ezért a teljes formája így fog kinézni:

```
# make target
```

ahol a *target* az egyik fordítási lehetõséget képviseli. Az elsõ ilyen targetnek mindig a `buildworld` -nek kell lennie.

Ahogy a neve is mutatja, a `buildworld` lefordítja az összes forrást a /usr/obj könyvtárba, majd a `installworld` mint másik target, telepíti az így létrehozott elemeket a számítógépre.

A targetek szétválasztása két okból is elõnyös. Elõször is lehetõvé teszi, hogy az új rendszert biztonságban lefordíthassuk, miközben az a jelenleg futó rendszert nem zavarja. A rendszer tehát képes "saját magát újrafordítani". Emiatt a `buildworld` target akár többfelhasználós módban is mindenféle nem kívánatos hatás nélkül használható. Ennek ellenére azonban továbbra is azt javasoljuk, hogy a `installworld` részt egyfelhasználós módban futtassuk le.

Másodrészt ezzel lehetõségünk nyílik NFS állományrendszer alkalmazásával több számítógépre is telepíteni hálózaton keresztül. Ha például három frissítendõ számítógépünk van, az `A`, `B` és `C`, akkor az `A` gépen elõször adjuk ki a `make buildworld`, majd a `make installworld` parancsot. A `B` és `C` gépek ezután NFS segítségével csatlakoztatják az `A`/usr/src és /usr/obj könyvtárait, amelyet követõen a `make installworld` paranccsal telepíteni tudjuk a fordítás eredményét a `B` és `C` gépekre.

Noha a `world` mint target még mindig létezik, használata határozottan ellenjavalt.

A

```
# make buildworld
```

parancs kiadásakor a `make` parancsnak megadható egy `-j` paraméter is, amellyel párhuzamosíthatjuk a folyamat egyes részeit. Ez általában többprocesszoros számítógépeken nyer értelmet, azonban mivel a fordítás folyamatának haladását inkább az állománymûveletek mintsem a processzor sebessége korlátozza, ezért alkalmazható akár egyprocesszoros gépeken is.

Tehát egy átlagos egyprocesszoros gépen így adható ki a parancs:

```
# make -j4 buildworld
```

Ennek hatására make(1) egyszerre 4 szálon igyekszik mûködni. A levelezési listákra beküldött tapasztalati jellegû bizonyítékok azt igazolják, hogy általában ez a beállítás adja a legjobb teljesítményt.

Ha többprocesszoros géppel rendelkezünk és rajta SMP támogatású rendszermagot indítottunk el, akkor érdemes 6 és 10 közötti értékekkel kísérleteznünk.

Idõigény

Számos tényezõ befolyásolja a fordítás tényleges idõbeli hosszát, de a FreeBSD-STABLE fa lefordítása mindenféle trükkök és rövidítések nélkül a legtöbb számítógépen olyan egy vagy két órára taksálható. A FreeBSD-CURRENT fához ennél valamivel több idõre lesz szükségünk.

24.7.8. Fordítsunk és telepítsünk egy új rendszermagot

Az újdonsült rendszerünket csak akkor tudjuk igazán kihasználni, ha egy új rendszermagot is készítünk hozzá. Ez gyakorlati szinten tulajdonképpen elvárás, mivel könnyen elõfordulhat, hogy bizonyos memóriabeli adatszerkezetek felépítése megváltozott, ezért némely programok, mint például a ps(1) és top(1), egészen addig nem lesznek képesek normálisan mûködni, amíg a rendszer és a rendszermag forráskódja nem illeszkedik egymáshoz.

Ennek legegyszerûbb és egyben legbiztonságosabb módja, ha a GENERIC beállításai alapján gyártunk és telepítünk egy rendszermagot. Még ha a GENERIC beállításai nem is tartalmazzák a rendszerünkben fellelhetõ összes eszközt, minden megtalálható bennük ahhoz, hogy a rendszert sikeresen elindíthassuk legalább egyfelhasználós módban. Ez mellesleg remek próbája az új rendszer életképességének. Miután elindítottuk a rendszert a GENERIC típusú rendszermaggal és meggyõzõdtünk róla, hogy a rendszer tényleg mûködõképes, a megszokott rendszermagunk konfigurációs állománya alapján nyugodtan elkészíthetjük ezután azt is.

FreeBSD alatt egy új rendszermag építése elõtt fontos újrafordítani az alaprendszert.

Ha saját beállításaink szerint akarunk rendszermagot létrehozni és már van is ehhez egy konfigurációs állományunk, akkor erre használhatjuk a `KERNCONF=SAJÁTMAG` paramétert is, valahogy így:

```
# cd /usr/src
# make buildkernel KERNCONF=SAJÁTMAG
# make installkernel KERNCONF=SAJÁTMAG
```

Hozzátennénk, hogy ha a `kern.securelevel` rendszerváltozó értékét 1 felé állítottuk *és* a rendszermag állományának beállítottunk `noschg` vagy hozzá hasonló állományjelzõt, akkor az `installkernel` lefuttatásához mindenképpen egyfelhasználós módba kell váltanunk. Minden más esetben további bonyodalmak nélkül ki tudjuk adni az említett parancsokat. A `kern.securelevel` részleteirõl az init(8) oldalán, a különbözõ állományjelzõkrõl pedig a chflags(1) oldalán olvashatunk.

24.7.9. Indítsuk újra a rendszert egyfelhasználós módban

Az új rendszermag mûködésének leteszteléséhez indítsuk újra a rendszert egyfelhasználós módban. Ennek pontos részleteit lásd Váltsunk egyfelhasználós módba.

24.7.10. Telepítsük az új rendszer binárisait

Ha a FreeBSD friss változatát nemrég fordítottuk le a `make buildworld` paranccsal, akkor utána az `installworld` segítségével tudjuk telepíteni a keletkezett programokat.

Tehát írjuk be ezeket:

```
# cd /usr/src
# make installworld
```

Amennyiben a paranccsorban a `make buildworld` használata során adtunk meg változókat, akkor ne felejtsük el ugyanazokat megadni a `make installworld` kiadása során sem. Ez viszont a többi paraméterre már nem feltétlenül érvényes. Például a `-j` beállítást szigorúan tilos az `installworld` targettel együtt használni.

Ennek megfelelõen tehát ha korábban ezt írtuk be:

```
# make -DNO_PROFILE buildworld
```

akkor így telepítsünk:

```
# make -DNO_PROFILE installworld
```

Máskülönben azokat a profilozott függvénykönyvtárakat próbáljuk meg telepíteni, amelyek a `make buildworld` futása során nem jöttek létre.

24.7.11. Frissítsük a `make installworld` által kihagyott állományokat

Az alaprendszer újrafordítása nem regisztrálja az új vagy megváltozott állományokat bizonyos könyvtárakban (különösen értendõ ez az /etc, /var és /usr esetén).

Az ilyen állományokat a legegyszerûbben a mergemaster(8) használatával tarthatjuk karban, de igény szerint akár kézzel is elvégezhetjük a szükséges aktualizálásokat. Függetlenül attól, hogy mit is választunk, mindenképpen készítsünk biztonsági mentést az /etc könyvtárról arra az esetre, ha bármilyen szörnyûség történne.

A `mergemaster`

A mergemaster(8) segédprogram valójában egy Bourne szkript, amely segít az /etc könyvtárunkban és a forrásfában levõ /usr/src/etc könyvtárban elhelyezkedõ konfigurációs állományok közti eltérések megállapításában. Ezt a módszert ajánljuk arra, hogy összevessük a konfigurációs állományainkat a forrásfában található változataikkal.

A használatának megkezdéséhez egyszerûen írjuk be, hogy `mergemaster`, majd várjunk egy kicsit, amíg a `mergemaster` létrehoz magának egy átmeneti környezetet a / könyvtárból elindulva és megtölti azt a különbözõ rendszerszintû beállításokat tartalmazó állományokkal. Ezeket az állományokat aztán összehasonlítja a jelenleg érvényben levõ változataikkal. Ilyenkor a köztük talált eltéréseket a diff(1) formátumának megfelelõen módon mutatja meg, ahol a `+` jelöli a hozzáadott vagy módosított sorokat, a `-` pedig a teljesen eltávolítandó vagy cserélendõ sorokat. Errõl a formátumról bõvebben a diff(1) man oldalán találhatunk felvilágosítást.

A mergemaster(8) ezt követõen megmutatja az összes olyan állományt, ahol eltérést tapasztalt, és ezen a ponton van lehetõségünk letörölni (delete) az új állományokat (amelyekre itt most ideiglenes állományként hivatkozik), telepíteni (install) a módosítatlan ideiglenes (új) állományt, valamint összefésülni (merge) az ideiglenes (új) és a jelenlegi állományokat, vagy ismét átnézni (view) a diff(1) által jelzett különbségeket.

Ha az ideiglenes állomány törlését választjuk, akkor a mergemaster(8) ezt úgy értelmezi, hogy változatlanul meg akarjuk tartani a jelenlegi változatot és törölni az újat. Ezt alapvetõen nem javasoljuk, hacsak tényleg nem látunk valamilyen okot erre. A mergemaster(8) parancssorában a `?` begépelésével bármikor kérhetünk segítséget. Ha az állomány kihagyását (skip) választjuk, akkor majd ismét felajánlja, amikor végeztünk az összes többivel.

A módosítatlan ideiglenes állomány telepítésének választásával lecseréljük a jelenleg verziót az újra. Ha az aktuális verziót sem változtattuk meg, akkor számunkra ez a legjobb megoldás.

Az állományok összefésülésének kiválasztásakor kapunk egy szövegszerkesztõt, benne a két állomány tartalmával. Ilyenkor tudjuk a képernyõn soronként egyeztetni a két állományt, majd a belõlük a megfelelõ részek összeválogatásával kialakítani az eredményt. Ebben a feldolgozási módban az `l` (mint left, vagyis bal) billentyû lenyomására a bal oldalon látható részt, az `r` (mint right, vagyis jobb) lenyomására pedig a jobb oldalon látható részt választjuk ki. Az így keletkezõ eredményt ezután egy állományba kerül, amelyet telepíteni tudunk. Ez a megoldás olyan állományok esetében használható, amikor a felhasználó módosított az alapértelmezett beállításokat.

Ha a diff(1) szerinti alakban akarjuk átnézni a különbségeket, akkor a mergemaster(8) ugyanúgy

megmutatja ezeket, mint a paranccsor megjelenítése elõtt.

Miután a mergemaster(8) végigment a rendszerszintû állományokon, további opciókat mutat. Megkérdezheti, hogy újra létre akarjuk-e hozni a jelszavakat tároló állományt (rebuild), illetve a folyamat végén a megmaradt ideiglenes állományok törlésére (remove) vár választ.

Az állományok aktualizálása kézzel

Ha inkább manuálisan szeretnénk frissíteni, akkor nem másolhatjuk csak egyszerûen át az állományokat a /usr/src/etc könyvtárból a /etc könyvtárba és nem hagyhatjuk ezeket sorsukra. Egyes állományokat elõször "telepíteni" kell. Ez azért van így, mert a /usr/src/etc könyvtár *nem pusztán* az /etc könyvtár egyszerû másolata. Ráadásul az /etc könyvtárban vannak olyan állományok, amelyek a /usr/src/etc könyvtárban nem is találhatóak meg.

Ha (az ajánlottak szerint) a mergemaster(8) segítségével dolgozunk, nyugodtan átléphetünk a következõ szakaszra.

Saját magunk a legegyszerûbben ezt úgy tudjuk megoldani, ha telepítjük az állományokat egy új könyvtárba és ezután nekiállunk változásokat keresni.

Az /etc meglevõ tartalmának mentése

Habár elméletileg magától semmi sem fogja bántani ezt a könyvtárat, azért ettõl függetlenül mindig érdemes biztosra menni. Ezért másoljuk az /etc könyvtár tartalmát egy megbízható helyre. Például:

```
# cp -Rp /etc /etc.old
```

Az `-R` itt a rekurzív másolást jelenti, a `-p` pedig a dátumok, az állományok és egyebek tulajdoni viszonyainak megõrzését.

Az /etc új változatának telepítéséhez szükségünk lesz még további könyvtárakra is. Erre a feladatra a /var/tmp/root tökéletesen megfelel, ahol még létre kell hoznunk néhány alkönyvtárat.

```
# mkdir /var/tmp/root
# cd /usr/src/etc
# make DESTDIR=/var/tmp/root distrib-dirs distribution
```

Ezzel létrejön a szükséges könyvtárszerkezet és települnek az állományok. Sok üres alkönyvtár is keletkezik a /var/tmp/root könyvtáron belül, ezeket töröljük. Ezt a legkönnyebben így tehetjük meg:

```
# cd /var/tmp/root
# find -d . -type d | xargs rmdir 2/dev/null
```

Ezzel törlõdnek az üres könyvtárak. (A szabvány hibakimenetet átirányítottuk a /dev/null eszközre, és ezzel elnyomtuk a nem üres könyvtárak esetén keletkezõ hibaüzeneteket.)

A /var/tmp/root most már tartalmazza az összes olyan állományt, amelyek normális esetben a / könyvtáron belül foglalnak helyet. Ezt követõen nincs más dolgunk, csak végigmenni az itt található állományokon és megállapítani, miben térnek a meglévõektõl.

Vegyük észre, hogy a /var/tmp/root könyvtárba telepített állományok némelyikének neve "."-tal kezdõdik. Az írás pillanatában ezek csak a /var/tmp/root/ és /var/tmp/root/root/ könyvtárakban található parancsértelmezõhöz tartozó indító állományok lehetnek, habár adódhatnak még ilyenek (attól függõen, mikor olvassuk ezt). Ezért a feldolgozásukhoz ne felejtsük el a `ls -a` parancsot használni.

A diff(1) alkalmazásával legegyszerûbben így tudunk összehasonlítani két állományt:

```
# diff /etc/shells /var/tmp/root/etc/shells
```

Ennek hatására megjelennek az /etc/shells és az új /var/tmp/root/etc/shells állományok közti különbségek. A segítségével gyorsan el tudjuk dönteni, hogy összefésüljük-e a két állományt, vagy csak egyszerûen írjuk felül a régebbi verziót az újjal.

Az új könyvtár (/var/tmp/root) nevébe írjuk bele a dátumot is, így könnyedén össze tudunk hasonlítani több verziót is

A rendszer gyakori újrafordítása az /etc szintén gyakori aktualizálását is maga után vonja, ami viszont fárasztó lehet.

Az iménti folyamatot fel tudjuk gyorsítani, hogy ha az /etc legutoljára összefésült változatát megtartjuk. A most következõ eljárás ennek mikéntjét vázolja fel.

1. A megszokottak szerint fordítsuk le a rendszert. Majd amikor az /etc könyvtárat és a többit is frissíteni akarjuk, a célként megadott könyvtár nevében adjuk meg a dátumot. Ha tehát például 1998. február 14. van, akkor írjuk ezt:

   ```
   # mkdir /var/tmp/root-19980214
   # cd /usr/src/etc
   # make DESTDIR=/var/tmp/root-19980214 \
       distrib-dirs distribution
   ```

2. Fésüljük össze a könyvtárban található az állományokat a fentiekben körvonalazottak szerint.

   Befejezés után *õrizzük meg* a /var/tmp/root-19980214 könyvtárat.

3. Mikor újra letöltjük a legfrissebb forrásokat és megismételjük az elõbbi lépéseket, haladjunk megint az elsõ lépés szerint. Ekkor tehát létrejön egy újabb könyvtár, amelynek a neve ezúttal már /var/tmp/root-19980221 lesz (ha például hetente frissítünk).
4. Most már meg tudjuk vizsgálni a közbeesõ héten született eltéréseket, ha a

két könyvtárra kiadunk egy rekurzív diff(1) hívást:

```
# cd /var/tmp
# diff -r root-19980214 root-19980221
```

Általában így kevesebb eltérést kapunk, mint amennyi például a /var/tmp/root-19980221/etc/ és az /etc összehasonlítása során elkerült volna. Mivel kisebb a keletkezett különbségek száma, ezért könnyebb lesz átvinnünk az /etc könyvtárunkba is a módosításokat.

5. Ezután törölhetjük a régebbi /var/tmp/root-* könyvtárat:

```
# rm -rf /var/tmp/root-19980214
```

6. Az /etc összefésülésekor mindig ismételjük meg ezeket a lépéseket.

A date(1) meghívásával akár automatikussá is tehetjük a könyvtárak névadását:

```
# mkdir /var/tmp/root-`date "+%Y%m%d"`
```

24.7.12. Újraindítás

Ezzel készen is vagyunk. Miután ellenõriztük, hogy minden a megfelelõ helyére került, indítsuk újra a rendszert. Ehhez egy egyszerû shutdown(8) is elegendõ:

```
# shutdown -r now
```

24.7.13. Befejeztük!

Gratulálunk, sikerült frissítenünk a FreeBSD rendszerünket.

Ha mégis valami balul ütne ki, könnyen újra tudjuk fordítani a rendszer egyes részeit. Például, ha véletlenül letöröltük az /etc/magic állományt az /etc frissítése vagy összefésülése során, a file(1) parancs nem fog tudni rendesen mûködni. Ilyenkor a következõket kell tennünk a hiba kijavításához:

```
# cd /usr/src/usr.bin/file
# make all install
```

24.7.14. Kérdések

Minden egyes változtatásnál újra kell fordítani a rendszert?

Nem könnyû választ adni erre a kérdésre, mivel ez alapvetõen a változtatás jellegétõl függ. Például, ha elindítjuk a CVSup programot és csak az alábbi állományok frissülnek:

```
src/games/cribbage/instr.c
src/games/sail/pl_main.c
src/release/sysinstall/config.c
src/release/sysinstall/media.c
src/shared/mk/bsd.port.mk
```

Ekkor valószínûleg nem éri meg újrafordítani a teljes rendszert. Elegendõ csupán belépni az érintett állományokat tartalmazó alkönyvtárakba és ott rendre kiadni a `make all install` parancsot. Ha viszont már valami komolyabb, például az src/lib/libc/stdlib változott meg, akkor vagy az egész rendszert, vagy legalább azon részeit fordítsuk újra, amely statikusan linkeltek (és minden más idõközben még hozzáadott statikusan linkelt dolgot).

Hogy melyik megoldást választjuk, teljesen rajtunk áll. Újrafordíthatjuk az egész rendszert kéthetente, mondván, hadd gyüljenek fel szépen a módosítások, vagy a függõségek pontos kielemzésével csak azokat az elemeket fordítjuk újra, amelyek tényleg meg is változtak.

Természetesen az egész attól függ, hogy milyen gyakran és melyik rendszert, a FreeBSD-STABLE-t vagy a FreeBSD-CURRENT-et frissítjük.

A fordító rengeteg 11-es jelzést (signal 11)signal 11 (vagy másfajta jelzéseket) dob hibával. Mi történhetett?

Ez általában hardveres meghibásodásra utal. A rendszer újrafordítása alapjaiban véve egy remek módszer számítógépünk alkatrészeinek terhelésére, ezért gyakorta elõhozza a memória már meglevõ hibáit. Ezek többnyire abban fogalmazódnak meg, hogy a fordító rejtélyes módon leáll mindenféle furcsa jelzések hatására.

Errõl biztosan úgy tudunk meggyõzõdni, ha újraindítjuk a make programot és az a folyamat egy teljesen másik pontján vérzik el.

Ilyenkor nem tudunk mást tenni, mint egymás után kicserélgetjük, kivesszük az alkatrészeket és így próbáljuk megállapítani, pontosan melyikük is okozza a gondokat.

A fordítása befejezése után törölhetem a /usr/obj könyvtárat?

Röviden: Igen.

A /usr/obj tartalmazza a fordítás folyamata során keletkezõ összes tárgykódot. Ennek törlése általában a `make buildworld` elsõ lépései között szerepel. Ezért tulajdonképpen a /usr/obj megtartásának nincs túlságosan sok értelme, viszont elég sok (jelenleg úgy kb. 340 MB) helyet fel tudunk így szabadítani.

Ha azonban értjük a dolgunkat, akkor megadhatjuk a `make buildworld` parancsnak, hogy hagyja ki ezt a lépést. Ennek hatására a fordítás sokkal hamarabb véget ér, mivel a legtöbb forrást így nem kell újrafordítani. Üröm az örömben, hogy ha netalán aprócska függõségi problémák merülnének

fel, akkor az egész fordítás megfeneklik mindenfelé különös módokon. Emiatt gyakran írnak feleslegesen leveleket a FreeBSD levelezési listáira, melyek a rendszer sikertelen újrafordításáról panaszkodnak, miközben kiderül, hogy az maguk az érintettek akarták lerövidíteni a folyamatot.

Lehetséges a megszakadt fordítás folytatása?

Ez attól függ, hogy a probléma bekövetkezése elõtt mennyire sikerült eljutni a fordításban.

Általában (tehát nem feltétlenül minden esetben) a `make buildworld` lefordítja a fordításhoz szükséges eszközök (például a gcc(1) és make(1)) újabb változatait és a rendszer függvénykönyvtárait, majd ezeket telepíti. Ezután ezekkel az új eszközökkel lefordíttatja saját magukat és ismét telepíti. Ezt követõen fordítja újra az új rendszerállományokkal az egész rendszert (így ezúttal már az olyan szokásos felhasználói programokat is, mint például az ls(1) és a grep(1)).

Ha tudjuk, hogy az utolsó fázisban álltunk le (mivel megnéztük a fordításhoz tartozó kimenetet), akkor (minden további nélkül) elég ennyi:

```
kijavítjuk a hibát ...
# cd /usr/src
# make -DNO_CLEAN all
```

Ezzel megmarad a korábbi `make buildworld` munkájának eredménye.

Ha ezt az üzenetet látjuk a `make buildworld` kimenetében:

```
------------------------------------------------------------
Building everything..
------------------------------------------------------------
```

akkor különösebb gond nélkül megcsinálhatjuk.

Amennyiben viszont nem látunk ilyen üzenetet, vagy nem vagyunk benne biztosak, akkor még mindig jobb elõvigyázatosnak lenni, ezért kénytelenek leszünk teljesen elölrõl kezdeni a fordítást.

Hogyan tudjuk felgyorsítani a fordítást?

- Futtassuk egyfelhasználós módban.
- Tegyük a /usr/src és /usr/obj könyvtárakat külön állományrendszerekre, külön lemezekre. Sõt, ha lehetséges, akkor ezeket a lemezeket tegyük külön lemezvezérlõkre.
- Még mindig jobb, ha ezeket az állományrendszereket a ccd(4) (lemezek összefûzését vezérlõ meghajtó) segítségével kiterjesztjük több lemezes eszközre.
- Kapcsoljuk ki a profilozást (az /etc/make.conf állományban a "NO_PROFILE=true" megadásával). Többnyire úgy sem lesz rá szükségünk.
- Az /etc/make.conf állományban a `CFLAGS` változót állítsuk az `-O -pipe` értékre. Az `-O2` gyakran sokkal lassabb, az `-O` és `-O2` alig tér el az optimalizálás mértékében. A `-pipe` paraméter hatására

pedig a fordítóprogram átmeneti állományok helyett csöveket használ a kommunikációra, és így megtakarít némi lemezhasználatot (a memóriahasználat terhére).

- Ha a make(1) parancsnak átadjuk a `-j_n_` paramétert, akkor képes több mindent párhuzamosan futtatni. Ez sok esetben segít attól függetlenül, hogy egy- vagy többprocesszoros gépünk van.
- A /usr/src könyvtárat tartalmazó állományrendszert csatlakoztathatjuk (vagy újracsatlakoztathatjuk) a `noatime` beállítással. Ilyenkor az állományrendszer nem rögzíti a hozzáférés idejét. Erre az információra sincs igazából szükségünk.

```
# mount -u -o noatime /usr/src
```

A fenti példa azt feltételezi, hogy a /usr/src könyvtárnak saját állományrendszere van. Ha ez nem így lenne (tehát például a /usr része), akkor itt azt kell megadnunk, nem pedig a /usr/src nevét.

- A /usr/obj könyvtárat tartalmazó állományrendszert csatlakoztathatjuk (vagy újracsatlakoztathatjuk) az `async` beállítással. Ennek hatására a lemez írása aszinkron módon történik. Magyarul az írási mûveletek azonnal befejezõdnek, miközben az adat ténylegesen csak pár másodperccel késõbb kerül ki a lemezre. Ezzel az írási kérelmek gyönyörûen összegyûjthetõek, ami nagymértékû növekedést eredményez a teljesítményben.

Ne felejtsük el azonban, hogy ezzel együtt az állományrendszerünk is sérülékenyebbé válik. Ezen beállítás használatával megnõ annak az esélye, hogy egy áramkimaradást követõ indításnál az állományrendszer helyreállíthatatlan állapotba kerül.

Ha egyedül csak a /usr/obj található ezen az állományrendszeren, akkor ez nem jelent akkora veszélyt. Amikor viszont rajta kívül még értékes adat is található az állományrendszeren, a beállítás érvényesítése elõtt mindenképpen készítsünk róla friss mentéseket.

```
# mount -u -o async /usr/obj
```

Ahogy arról az elõbb is szó esett, ha a /usr/obj nem egy különálló állományrendszeren található, akkor a példában szereplõ csatlakozási pontot cseréljük ki a megfelelõre.

Mi tegyünk, ha valami nem megy rendesen?

Egyértelmûen bizonyosodjunk meg róla, hogy a korábbi fordításokból nem maradtak vissza semmiféle kóbor állományok. Ennyi sokszor pontosan elég.

```
# chflags -R noschg /usr/obj/usr
# rm -rf /usr/obj/usr
# cd /usr/src
```

```
# make cleandir
# make cleandir
```

Igen, a `make cleandir` parancsot tényleg kétszer kell kiadni.

Ezután a `make buildworld` parancstól indulva kezdjük újra a fordítást.

Ha még ezek után is fennáll a probléma, küldjük el a hibát tartalmazó kimenetet és a `uname -a` parancs eredményét a FreeBSD general questions levelezési lista címére. Ne lepõdjünk meg, ha a beállításainkra vonatkozóan még kapunk további kérdéseket is!

24.8. A források követése több géppel

Ha egyszerre több számítógéppel is szeretnénk követni ugyanannak a forrásfának a változásait és ezért mindegyikre letöltjük a forrásokat majd újrafordítjuk ezeket, akkor sok erõforrást, de leginkább lemezterületet, hálózati sávszélességet és processzoridõt, feleslegesen használunk. Ezekkel úgy tudunk spórolni, ha valójában csak egyetlen géppel végeztetjük el a munka legtöbb részét, miközben a többi NFS használatával dolgozik. Ez a szakasz ezt a módszert foglalja össze.

24.8.1. Elõkészületek

Elõször is szedjük össze az egyezõ binárisokat futtató gépeket, melyekre a továbbiakban csak *fordítási csoport* néven hivatkozunk. Minden gépnek lehet saját rendszermagja, viszont a felhasználói programok mindegyikõjük esetében ugyanazok. Ebbõl a csoportból válasszuk ki egy *fordító gépet*. Ez lesz az a gép, amelyen a rendszer és a rendszermag lefordításra kerül. Ideális esetben ez a leggyorsabb gép, amelynek elegendõ a processzorkapacitása arra, hogy lefuttassa a `make buildworld` és `make buildkernel` parancsokat. Érdemes még rajta kívül kiválasztanunk egy *tesztelõ gépet* is, ahol a véglegesítés elõtt kipróbálhatjuk a szoftverfrissítéseket. Ennek egy olyan gépnek *kell* lennie, amely akár hosszabb ideig is nélkülözhetõ a csoportból. Lehet akár maga a fordítást végzõ gép is, de nem elvárás.

A fordítási csoportban levõ összes gépnek ugyanarról a géprõl és ugyanarra a pontra kell csatlakoztatnia a /usr/obj és /usr/src könyvtárakat. Ezek optimális esetben a fordítással foglalkozó gép két külön lemezmeghajtóján vannak, melyek egyaránt elérhetõek NFS-en keresztül. Ha több fordítási csoportunk is van, akkor az /usr/src könyvtárnak elegendõ csak egyetlen fordító gépen meglennie, a többi pedig csatlakoztassa NFS-en keresztül.

Végül gyõzõdjünk meg róla, hogy az /etc/make.conf és a /etc/src.conf állományok tartalma a fordítási csoport mindegyik gépénél megegyezik a fordító gépével. Ez azt jelenti, hogy a fordító gépnek az alaprendszer ugyanazon részeit és ugyanúgy kell létrehozni, mint amelyet a fordítási csoport akármelyik gépére telepíteni is akarunk. Ezenkívül még a fordítási csoportban levõ minden egyes gép /etc/make.conf állományában a `KERNCONF` értékének a saját rendszermagjára vonatkozó konfigurációt kell megadni, illetve a fordítással foglakozó gép `KERNCONF` változójánál pedig az együtt összeset, a sajátjával kezdve. Ennek megfelelõen a fordító gépnek a rendszermagok lefordításához rendelkeznie kell az egyes gépek /usr/src/sys/arch/conf könyvtárában meglevõ állományaival.

24.8.2. Az alaprendszer

Most, miután mindent megfelelõen elõkészítettünk, készen állunk a munkára. A Az alaprendszer fordításaban leírtak szerint fordítsuk le a rendszermagokat és az alaprendszert a fordító gépen, de utána még nem telepítsünk semmit se. Ha befejezõdött a fordítás, lépjünk be a tesztelõ gépre és telepítsük a frissen fordított rendszermagot. Ha ez a gép NFS-en keresztül éri a /usr/src és /usr/obj könyvtárakat, akkor az egyfelhasználós módban aktiválni kell a hálózatot, majd csatlakoztatni ezeket. Ezt legkönnyebben úgy tudjuk megcsinálni, ha a gépet elõször elindítjuk többfelhasználós módban, majd a `shutdown now` paranccsal egyfelhasználós módba váltunk. Ha eljuttunk ide, telepítsünk az új rendszermagot és rendszert, illetve a megszokott módon futtassuk a `mergemaster` parancsot. Amikor ezt befejeztük, ezen a gépen térjünk vissza a hétköznapi többfelhasználós mûködési módba.

Miután a tesztelésre szánt gépen ellenõriztük, hogy minden a megfelelõ módon mûködik, az elõbb tárgyalt eljárással telepítsük fel a fordítási csoportban levõ összes többi gépre is az új szoftvereket.

24.8.3. Portok

Ugyanezt a gondolatmenet alkalmazható a portfa esetében is. Az elsõ és egyben legfontosabb lépés a /usr/ports csatlakoztatása ugyanarról a géprõl a fordítási csoport minden gépére. Az /etc/make.conf megfelelõ beállításával még a terjesztési állományokat is meg tudjuk osztani. A `DISTDIR` értékét egy olyan közösen használt könyvtárra állítsuk, amely írható az NFS-en keresztül megosztott állományrendszerünkben a `root` felhasználóként tevékenykedõk számára. A `WRKDIRPREFIX` változót minden gépen egy helyi fordítási könyvtárra állítsuk. Zárásképpen még hozzátesszük, hogy ha csomagokat akarunk készíteni és mások számára is elérhetõvé tenni, akkor ne felejtsük el a `PACKAGES` változót a `DISTDIR` változóhoz hasonlóan beállítani.

Chapter 25. DTrace

25.1. Áttekintés

A DTrace, vagy más néven Dynamic Tracing technológiát a Sun™ dolgozta ki szerverek teljesítményében jelentkezõ szûk keresztmetszetek felderítésének megkönnyítésére. Ez nem egy nyomkövetésre szolgáló megoldást takar, hanem inkább a rendszer valós idejû elemzését és teljesítményének vizsgálatát elõsegítõ eszközt.

A DTrace figyelemre méltó elemzõeszköz, rengeteg rendkívül hasznos képességgel rendelkezik a rendszerben felbukkanó problémák diagnosztizálására. Elõre programozott szkriptek segítségével pedig ezen képességek további elõnyeit tudjuk kihasználni, ugyanis a DTrace programozható egy ún. D nyelven, amelynek révén a különbözõ vizsgálatokat könnyen a saját igényeink szerint tudjuk alakítani.

A fejezet elolvasása során megismerjük:

- mi is az a DTrace és milyen lehetõségei vannak;
- a Solaris™ és FreeBSD operációs rendszereken megtalálható DTrace implementációk közti eltéréseket;
- a DTrace FreeBSD alatt hogyan engedélyezhetõ és használható.

A fejezet elolvasásához ajánlott:

- a UNIX® és FreeBSD alapvetõ ismerete (A UNIX alapjai);
- a rendszermag konfigurációjának és fordításának alapvetõ ismerete (A FreeBSD rendszermag testreszabása);
- az operációs rendszerek és azon belül a FreeBSD biztonsági fogalmainak minimális ismerete (Biztonság);
- a FreeBSD forrásainak megszerzésének és azok lefordításának ismerete (A FreeBSD frissítése és frissen tartása).

Ez a funkció még folyamatos tesztelés alatt áll. Bizonyos részei még egyáltalán nem, vagy csak korlátozottan érhetõek el. A dokumentáció annak megfelelõen fog majd változni, hogy ezek az elemek fokozatosan elérik az éles felhasználáshoz szükséges szintet.

25.2. Eltérések az implementációban

Noha a FreeBSD alatt megtalálható DTrace implementáció nagyon hasonló az eredeti, Solaris™ alatt futó változathoz, tartalmaz bizonyos különbségeket, amelyeket a továbblépés elõtt mindenképpen érdemes megemlítenünk. Az egyik legfontosabb ilyen szembetûnõ különbség, hogy a FreeBSD esetén a DTrace használatát külön engedélyezni kell. A DTrace megfelelõ mûködéséhez tehát a rendszermag konfigurációs állományában meg kell adnunk bizonyos beállításokat és modulokat kell betöltenünk. Ezekrõl hamarosan szó lesz.

A rendszermag konfigurációs állományában a DDB_CTF opció segítségével tudjuk engedélyezni ún. CTF adatok betöltését mind a rendszermag moduljaiból, mind pedig magából a rendszermagból egyaránt. A CTF a Solaris™ "Compact Type Format" elnevezésû formátumára utal, amellyel például a DWARF megoldásához hasonló módon tárolhatunk tömörített alakban különbözõ típusú nyomkövetési információkat. Ilyen CTF adatok többek közt a ctfconvert és a ctfmerge használatával rendelhetõek hozzá bináris állományokhoz. A ctfconvert segédprogram a fordítóprogram által az ELF állományokban szereplõ DWARF típusú szakaszokban tárolt információkat képes beolvasni, és a ctfmerge a tárgykódban található CTF típusú ELF szakaszokat tudja végrehajtható állományokká vagy osztott könyvtárakka összefûzni. Röviden beszélni fogunk arról, hogyan lehet mindezeket a FreeBSD alaprendszerébe és rendszermagjába is beépíteni.

FreeBSD és Solaris™ esetén elõfordulhat, hogy más fajta providerek állnak rendelkezésünkre. Ezek közül talán a legfontosabb a dtmalloc, amely a FreeBSD rendszermagjában típus szerint teszi lehetõvé a malloc() függvény követését.

FreeBSD alatt kizárólag csak a root tudja használni a DTrace-t. Ennek oka a két operációs rendszer biztonsági megoldásai közti különbségekben keresendõ, mivel a Solaris™ esetén létezik néhány olyan alacsonyszintû ellenõrzés, amely a FreeBSD-nél még nincs. Ezért például a /dev/dtrace/dtrace eszköz szigorúan csak a root számára érhetõ el.

Végezetül megemlítjük, hogy a DTrace felhasználására a Sun™ CDDL licence vonatkozik. A Common Development and Distribution License FreeBSD a /usr/src/cddl/contrib/opensolaris/OPENSOLARIS.LICENSE állományban található, vagy interneten keresztül a http://www.opensolaris.org/os/licensing címen.

Ezen licenc értelmében a DTrace támogatással készített FreeBSD rendszermagok továbbra is BSD licencûek maradnak, azonban a rendszerrel terjesztett binárisok futtatásakor vagy a modulok betöltésekor már a CDDL érvényesül.

25.3. A DTrace támogatásának engedélyezése

A DTrace által felkínált lehetõségeket a következõ sorok hozzáadásával tudjuk engedélyezni a rendszermag konfigurációs állományában:

```
options         KDTRACE_HOOKS
options         DDB_CTF
```

AMD64 architektúrán ezeken kívül még az alábbi sor is kelleni fog:

```
options         KDTRACE_FRAME
```

Ezzel a beállítással az FBT ("function boundary tracing") részére nyújtunk támogatást. A DTrace ugyan enélkül is képes lesz mûködni, de akkor csak korlátozott mértékben tudunk ilyen típusú vizsgálatokat végezni.

Az egész rendszert újra kell fordítanunk a CTF használatával. Ennek elvégzéséhez a következõ

parancsokat kell kiadnunk:

```
# cd /usr/src
# make WITH_CTF=1 kernel
```

A fordítás befejezõdése után indítsuk újra a rendszerünket.

A rendszer újraindulása és az új rendszermag betöltõdése után szükségünk lesz egy Korn-féle parancsértelmezõre is, mivel a DTrace eszköztárában rengeteg, a `ksh` programra épülõ eszközt fogunk találni. Ezért tehát telepítsük a shells/ksh93 csomagot, de megjegyezzük, hogy ugyanezen eszközök számára a shells/pdksh vagy shells/mksh csomagok is megfelelnek.

Végül töltsük le a DTrace eszköztárának legfrissebb változatát. Az aktuális verzió a http://www.opensolaris.org/os/community/dtrace/dtracetoolkit/ címen érhetõ el. Képes önmagát telepíteni, de a benne található eszközök használatához nem kötelezõ ezt elvégezni.

25.4. A DTrace használata

A DTrace funkcióinak alkalmazásához léteznie kell egy DTrace eszköznek. Ennek létrehozásához be kell töltenünk a megfelelõ modult:

```
# kldload dtraceall
```

Innentõl már mûködésre kész a DTrace. Rendszeradminisztrátorként a következõ módon kérdezhetjük le a rendelkezésre álló vizsgálatokat:

```
# dtrace -l | more
```

Mivel lekérdezés eredménye pillanatok alatt betöltené az egész képernyõt, ezért az egészet még átirányítjuk a `more` parancshoz. Ha ez rendesen lefut, akkor a DTrace ténylegesen használhatónak tekinthetõ. Ezt követõen tekintsük át a hozzá tartozó eszközkészletet.

Ez a mellékelt eszközkészlet lényegében a rendszerrel kapcsolatos információk összegyûjtésére alkalmas szkripteket tartalmaz. Vannak szkriptek, amelyekkel a megnyitott állományokat, a memóriát, a processzorhasználatot és még sok minden mást kérdezhetünk le. A szkriptek a következõ parancs segítségével tömöríthetõek ki:

```
# gunzip -c DTraceToolkit* | tar xvf -
```

A `cd` parancs segítségével lépjünk be az így keletkezõ könyvtárba, és a kisbetûs névvel rendelkezõ állományok engedélyeit állítsuk be a `755` módra.

Mindegyik szkriptben el kell végeznünk némi módosítást: a /usr/bin/ksh hivatkozásokat írjuk át mindenhol a /usr/local/bin/ksh névre, illetve a /usr/bin/sh hivatkozásokat /bin/sh névre, majd végezetül pedig a /usr/bin/perl hivatkozásokat a /usr/local/bin/perl névre.

Itt még egyszer kiemelnénk, hogy a FreeBSD-ben jelenleg megtalálható DTrace támogatás *még nem teljes* és *kísérleti jelleggel* szerepel. Ezért bizonyos szkriptek nem fognak mûködni, vagy azért, mert túlságosan Solaris™ lehetõségeihez igazodnak, vagy pedig azért, mert a jelenlegi implementáció által még nem ismert vizsgálatokra támaszkodnak.

Jelenlegi ismereteink szerint a FreeBSD egyelõre csak két szkriptet támogat teljes mértékben, ezek a hotkernel és a procsystime. A szakasz további részében ezzel a kettõvel fogunk részletesebben foglalkozni.

A hotkernel feladata segíteni beazonosítani azokat a függvényeket, amelyek a legtöbb idõt veszik igénybe a rendszermagon belül. A szkript futtatásakor nagyjából a következõt csinálja:

```
# ./hotkernel
Sampling... Hit Ctrl-C to end.
```

A folyamat `Ctrl` + `C` billentyûkombináció hatására állítható meg. A szkript futásának befejezõdésekor különbözõ rendszermagbeli függvények és a hozzájuk tartozó idõk jelennek meg, az utóbbi szerint növekvõ sorrendben:

```
kernel`_thread_lock_flags                           2   0.0%
0xc1097063                                          2   0.0%
kernel`sched_userret                                2   0.0%
kernel`kern_select                                  2   0.0%
kernel`generic_copyin                               3   0.0%
kernel`_mtx_assert                                  3   0.0%
kernel`vm_fault                                     3   0.0%
kernel`sopoll_generic                               3   0.0%
kernel`fixup_filename                               4   0.0%
kernel`_isitmyx                                     4   0.0%
kernel`find_instance                                4   0.0%
kernel`_mtx_unlock_flags                            5   0.0%
kernel`syscall                                      5   0.0%
kernel`DELAY                                        5   0.0%
0xc108a253                                          6   0.0%
kernel`witness_lock                                 7   0.0%
kernel`read_aux_data_no_wait                        7   0.0%
kernel`Xint0x80_syscall                             7   0.0%
kernel`witness_checkorder                           7   0.0%
kernel`sse2_pagezero                                8   0.0%
kernel`strncmp                                      9   0.0%
kernel`spinlock_exit                               10   0.0%
kernel`_mtx_lock_flags                             11   0.0%
kernel`witness_unlock                              15   0.0%
kernel`sched_idletd                               137   0.3%
0xc10981a5                                      42139  99.3%
```

Ez a szkript modulok esetén is alkalmazható. Ezt a módját a `-m` kapcsoló megadásával aktiválhatjuk:

```
# ./hotkernel -m
Sampling... Hit Ctrl-C to end.
^C
MODULE                                                  COUNT   PCNT
0xc107882e                                                  1   0.0%
0xc10e6aa4                                                  1   0.0%
0xc1076983                                                  1   0.0%
0xc109708a                                                  1   0.0%
0xc1075a5d                                                  1   0.0%
0xc1077325                                                  1   0.0%
0xc108a245                                                  1   0.0%
0xc107730d                                                  1   0.0%
0xc1097063                                                  2   0.0%
0xc108a253                                                 73   0.0%
kernel                                                    874   0.4%
0xc10981a5                                             213781  99.6%
```

A procsystime szkript egy adott azonosítóval vagy névvel rendelkezõ programhoz tudja megadni az általa kezdeményezett rendszerhívások által felhasznált idõt. A most következõ példában elindítjuk a /bin/csh egy újabb példányát. A procsystime elindul, majd megvárja, amíg kiadunk néhány parancsot a csh frissen indított másolatában. A teszt eredményei tehát a következõk lesznek:

```
# ./procsystime -n csh
Tracing... Hit Ctrl-C to end...
^C

Elapsed Times for processes csh,

         SYSCALL          TIME (ns)
          getpid               6131
       sigreturn               8121
           close              19127
           fcntl              19959
             dup              26955
         setpgid              28070
            stat              31899
       setitimer              40938
           wait4              62717
       sigaction              67372
     sigprocmask             119091
    gettimeofday             183710
           write             263242
          execve             492547
           ioctl             770073
           vfork            3258923
      sigsuspend            6985124
            read         3988049784
```

Jól megfigyelhetõ, hogy (nanomásodpercekben mérve) a legtöbb idõt a `read()`, a legkevesebb idõt pedig a `getpid()` rendszerhívás vette igénybe.

25.5. A D nyelv

A DTrace eszköztárában megtalálható számos szkript a DTrace saját programozási nyelvén íródott. Ezt a nyelvet nevezik a Sun™ implementációjában "a D nyelvnek". Ennek ismertetésére itt most külön nem térünk ki, azonban a http://wikis.sun.com/display/DTrace/Documentation címen igen részletesen olvashatunk róla.

Part IV: Hálózati kommunikáció

A FreeBSD az egyik legelterjedtebb operációs rendszer a legnagyobb hálózati teljesítményt nyújtó kiszolgálók körében. Az itt található fejezetek témái:

- Soros kommunikáció
- PPP és PPP Etherneten keresztül (PPPoE)
- Elektronikus levelezés
- Hálózati kiszolgálók futattása
- Tûzfalak
- Egyéb haladó hálózati témák

Ezek a fejezetek nem állnak egymással szoros kapcsolatban, csupán egy adott témáról adnak ismereteket. Ennélfogva nem kötelezõ ezeket sorrendben elolvasni, valamint egyáltalán nem is kell mindegyikõjüket átolvasni ahhoz, hogy a FreeBSD-t hálózati környezetben is használni tudjuk.

Chapter 26. Soros vonali kommunikáció

26.1. Áttekintés

A UNIX® mindig is támogatta a soros vonali kommunikációt. Tulajdonképpen az elsõ UNIX®-os gépek is soros vonalon kapták a felhasználóktól a bemenetet és ugyanígy küldték vissza a kimenetet. Az idõk azóta már sokat változtak, hogy egy átlagos "terminál" mindössze egy 10 karakter per másodperc sebességû soros nyomtatóból és egy billentyûzetbõl állt. Ebben a fejezetben ismertetünk néhány olyan megoldást, amellyel a FreeBSD képes soros vonalon keresztül kommunikálni.

A fejezet elolvasása során megismerjük:

- hogyan kapcsoljunk terminálokat a FreeBSD rendszerünkre;
- hogyan tárcsázzunk modem segítségével távoli számítógépeket;
- hogyan tegyük lehetõvé gépünkre a bejelentkezést távoli felhasználók számára;
- hogyan indítsuk a rendszerünket soros konzolról.

A fejezet elolvasásához ajánlott:

- egy új rendszermag beállításának és telepítésének ismerete (A FreeBSD rendszermag testreszabása);
- a UNIX®-os engedélyek és a UNIX® alatt futtatott programok mûködtetésének megértése (A UNIX alapjai);
- annak a soros vonali hardvernek (modemnek vagy többportos kártyának a) kézikönyve, amelyet a FreeBSD-vel használni szeretnénk

26.2. Bevezetés

26.2.1. Alapfogalmak

bps
: Bit per másodperc - az adatátvitel sebessége

DTE
: Adatterminál eszköz (Data Terminal Equipment) - ez például a számítógépünk

DCE
: Adatkommunikációs eszköz (Data Communications Equipment) - ez a modem

RS-232
: a hardveres soros vonali kommunikációhoz szükséges EIA szabványú kábel

Amikor ebben a fejezetben az adatátvitel sebességérõl beszélünk, akkor szándékosan nem használjuk a "baud" fogalmát. A baud ugyanis a kommunikációs eszközben adott idõ alatt lezajló

jelváltások mennyiségét jelöli, miközben itt a "bps" (bit per másodperc) kifejezés használata a *helyes* (vagy legalább is a szõrszálhasogatók egyelõre megnyugodhatnak).

26.2.2. Kábelek és portok

Ha a FreeBSD rendszerünkhöz egy modemet vagy egy terminált akarunk csatlakoztatni, akkor ahhoz a számítógépünkben szükség lesz egy szabad soros portra és egy megfelelõ típusú kábelre. Ha már tisztában vagyunk a rendelkezésre álló hardverrel és a hozzá tartozó kábellel, akkor nyugodtan átléphetjük ezt a részt.

A kábelek fajtái

A soros kábeleknek több különbözõ típusa van. Közülük a céljainknak leginkább megfelelõ két legismertebb változatuk az ún. null-modem és a szabványos ("egyenes") RS-232-es soros kábelek. A hardverhez tartozó dokumentációban megtaláljuk, hogy pontosan melyik típus tartozik hozzá.

A null-modem kábelek

Egy null-modem kábel bizonyos jeleket, többek közt a "földet" (Signal Ground, SG), egyenesen küldi, másokat viszont felcserélten. Például az "átküldött adat" (Transmitted Data, TD) jelzésû tû a kábel másik végén a "fogadott adat" (Received Data, RD) tûhöz fut be.

A terminálokhoz akár saját magunk is le tudunk gyártani egy null-modem kábelt (például ha a boltiakkal nem lennénk megelégedve). A következõ táblázatban az RS-232C jeleit és érintkezõinek számozását láthatjuk egy DB-25-ös csatlakozó esetében. A szabvány a kábel két 1-es tûjét összekapcsoló vonalat *védõföldnek* (Protective Ground, PD) nevezi, de ezt gyakran el is hagyják. Némely terminál remekül mûködik mindössze a 2-es, 3-as és 7-es tûk használatával, miközben mások az iménti példától eltérõ kiosztást igényelnek.

Táblázat 8. A DB-25 DB-25 közti null-modem kábel

Jel	Tû		Tû	Jel
SG	7	párja:	7	SG
TD	2	párja:	3	RD
RD	3	párja:	2	TD
RTS	4	párja:	5	CTS
CTS	5	párja:	4	RTS
DTR	20	párja:	6	DSR
DTR	20	párja:	8	DCD
DSR	6	párja:	20	DTR
DCD	8	párja:	20	DTR

Íme a mostanság elterjedt másik két séma.

Táblázat 9. A DB-9 DB-9 közti null-modem kábel

Jel	Tû		Tû	Jel
RD	2	párja:	3	TD
TD	3	párja:	2	RD
DTR	4	párja:	6	DSR
DTR	4	párja:	1	DCD
SG	5	párja:	5	SG
DSR	6	párja:	4	DTR
DCD	1	párja:	4	DTR
RTS	7	párja:	8	CTS
CTS	8	párja:	7	RTS

Táblázat 10. DB-9 DB-25 közti null-modem kábel

Jel	Tû		Tû	Jel
RD	2	párja:	2	TD
TD	3	párja:	3	RD
DTR	4	párja:	6	DSR
DTR	4	párja:	8	DCD
SG	5	párja:	7	SG
DSR	6	párja:	20	DTR
DCD	1	párja:	20	DTR
RTS	7	párja:	5	CTS
CTS	8	párja:	4	RTS

Amikor egy tû az átellenes oldalon két másik tûhöz csatlakozik, akkor azt általában úgy valósítják meg, hogy a két tût a saját oldalukon összekötik, majd ezt kapcsolják hozzá a harmadik tûhöz.

Ezek a megoldások a legnépszerûbbek. Természetesen a tûk összekötésének több más variációja is létezik (ezekrõl az *RS-232 Made Easy* c. könyvben olvashatunk bõvebben), ahol az SG párja az SG, a TD párja az RD, az RTS és a CTS párja az DCD, a DTR párja a DSR és ugyanezek fordítva.

Szabványos RS-232C kábelek

A szabványos soros kábel az összes RS-232C jelet közvetlenül átküldi. Vagyis a kábel egyik végén levõ "átküldött adat" tû a másik végén is az "átküldött adat" tûhöz csatlakozik. Az ilyen típusú kábeleket többnyire a számítógépek és a modemek között alkalmazzák, de egyes terminál típusok esetében is szükségünk lehet rá.

A portok

A soros port olyan eszköz, amelyen keresztül a FreeBSD-s gép és a terminál között adatokat tudunk

közvetíteni. Ebben a szakaszban az ilyen portok különféle típusait és ezek használatát ismertetjük FreeBSD alatt.

A portok típusai

A soros portoknak több típusa létezik. Mielõtt vásárolnánk egy készítenénk egy soros kábelt, mindenképpen gyõzödjünk meg róla, hogy csatlakoztatni tudjuk majd a FreeBSD-s rendszerünkhöz és a terminálhoz egyaránt.

A legtöbb terminálon DB-25-ös portot találunk. A személyi számítógépek, köztük azok, amelyeken FreeBSD fut, DB-25-ös és DB-9es portokkal rendelkeznek. Ha a gépünkben egy többportos soros kártya van, akkor ezeken kívül még RJ-12-es és RJ-45-ös portjaink is lehetnek.

A hardverhez tartozó dokumentációból tudjuk kideríteni az adott port konkrét fajtáját, de gyakran a port vizuális vizsgálata is segíthet eldönteni a kérdést.

A portok nevei

FreeBSD alatt az egyes soros portokat a /dev könyvtárban található eszközleírókon keresztül tudjuk elérni. Ezeknek két típusa van:

- A behíváshoz használt portok nevei /dev/ttydN alakúak, ahol az *N* a port sorszáma, ami nullától indul. A behívó portok alapvetõen a terminál esetében használatosak. A behívó portok használatához a soros vonalon az "vonal észlelése" (Data Carrier Detect, DCD) jelnek kell megbízhatóan mûködnie.
- A híváshoz használt portok nevei /dev/cuadN alakúak. A hívó portokat terminálok esetében ritkán alkalmazzák, helyettük inkább csak modemekhez használják. A hívó portokat akkor érdemes használni, ha a soros kábel vagy a terminál nem ismeri a DCD jelet.

Ha a terminált az elsõ soros portra (ami MS-DOS®-ban a COM1) csatlakoztattuk, akkor a /dev/ttyd0 segítségével fogunk rá hivatkozni. Ha viszont a második soros porton (más néven COM2) található, akkor a /dev/ttyd1 eszközt használjuk, és így tovább.

26.2.3. A rendszermag beállítása

A FreeBSD alapból négy soros portot támogat. Az MS-DOS® világban ezeket rendre COM1, COM2, COM3 és COM4 portoknak nevezik. A FreeBSD jelen pillanatban ismeri még a "butább" többportos soros csatolókártyákat is, például a BocaBoard 1008 és 2016 típusokat, valamint több intelligensebb többportos kártyát, például a Digiboard és a Stallion Technologies gyártmányait. Az alap rendszermag azonban csak a szabványos COM portokat keresi.

Ha ellenõrizni akarjuk, hogy a rendszermag rendben megtalálta a soros portokat, akkor figyelmesen olvassuk el a rendszerindítás során megjelenõ üzeneteket, vagy az `/sbin/dmesg` parancs kiadásával kérdezzük vissza a rendszermag üzeneteit. Különösen a `sio` kezdetû sorokra kell figyelnünk.

Az alábbi paranccsal tudjuk leszûrni a `sio` szövegrészt tartalmazó sorokat:

```
# /sbin/dmesg | grep 'sio'
```

Például, ha négy soros port található a rendszerünkben, akkor a rájuk vonatkozó rendszerüzenetek a következõk lesznek:

```
sio0 at 0x3f8-0x3ff irq 4 on isa
sio0: type 16550A
sio1 at 0x2f8-0x2ff irq 3 on isa
sio1: type 16550A
sio2 at 0x3e8-0x3ef irq 5 on isa
sio2: type 16550A
sio3 at 0x2e8-0x2ef irq 9 on isa
sio3: type 16550A
```

Ha a rendszermagunk nem ismerte volna fel az összes soros portot, akkor valószínûleg a /boot/device.hints állományt kell módosítanunk. Tegyük megjegyzésbe vagy akár teljesen távolítsuk is el azokat az eszközöket, amelyekkel nem rendelkezünk.

A soros portok és a többportos kártyák beállításával kapcsolatban a sio(4) man oldalát olvassuk el. Óvatosan bánjunk a FreeBSD megelõzõ változataiból származó konfigurációs állományokkal, mert az eszközök vonatkozó beállításokat és azok formátuma megváltozhatott azóta.

Az `port IO_COM1` a `port 0x3f8`, az `IO_COM2` a `0x2f8`, az `IO_COM3` a `0x3e8` és az `IO_COM4` a `0x2e8` beállítást helyettesíti. Ezek az adott porthoz tartozó gyakori címeket képviselik. A 4-es, 3-as, 5-ös és 9 megszakítások is igen általánosak ezeknél. A hagyományos soros portok viszont az ISA buszos PC-k esetében *nem képesek* a megszakításokon osztozni. (A többportos kártyák azonban lehetõvé teszik az 16550A számára, hogy mindössze egy vagy két megszakítást használjon.)

26.2.4. Speciális eszközállományok

A rendszermagban található legtöbb eszköz az ún. "speciális eszközállományokon" keresztül érhetõ el, melyek a /dev könyvtárban találhatóak. A sio eszközök a /dev/ttydN (behívó portok) és /dev/cuadN (hívó portok) állományok használatával érhetõek el. A FreeBSD ezenkívül még külön eszközállományokat biztosít az inicializációhoz (/dev/cuadN.init) és a zároláshoz (/dev/cuadN.lock). Az inicializációs állományok a port megnyitásakor használhatóak a hozzá tartozó paraméterek beállítására, például így tudjuk elküldeni a `crtscts` utasítást az olyan modemeknek, amelyek a forgalom irányítását `RTS/CTS` jelzéseken keresztül valósítják meg. A zároló állományokkal a portokra vonatkozó zárolásokat állíthatjuk be, így a felhasználók vagy a programok nem lesznek képesek bizonyos paramétereket megváltoztatni. A termios(4), sio(4) és stty(1) man oldalakon olvashatunk részletesebben a terminálok beállításairól, valamint az eszközök zárolásáról és inicializálásáról.

26.2.5. A soros port beállítása

A ttydN (vagy cuadN) lesz az az eszköz, amit majd az alkalmazásainkból el akarunk érni. Amikor egy futó program megnyit egy ilyen eszközt, mindig tartoznak hozzá alapértelmezett terminál I/O

beállítások. Ezeket a következõ paranccsal tudjuk lekérdezni:

```
# stty -a -f /dev/ttyd1
```

Ha megváltoztatjuk az eszköz beállításait, akkor azok egészen addig érvényben is maradnak, amíg le nem zárjuk. Ha tehát ezután újra megnyitjuk, akkor minden visszaáll az alapértelmezett állapotra. Az alapértelmezett beállítások megváltoztatásához a "kezdeti állapotot" szimbolizáló eszközt kell megnyitnunk és átállítanunk. Például, ha alapból engedélyezni akarjuk a `CLOCAL` módot, a 8 bites kommunikációt és a `XON/XOFF` típusú forgalomirányítást a ttyd5 eszközön, akkor a következõt gépeljük be:

```
# stty -f /dev/ttyd5.init clocal cs8 ixon ixoff
```

A soros eszközök rendszerszintû inicializálását az /etc/rc.d/serial állomány vezérli. Lényegében ez határozza meg az összes soros eszköz alapértelmezett beállítását.

Ha bizonyos beállítások megváltoztatását tiltani szeretnénk az alkalmazások felé, akkor azt a "zárolt állapotot" tartalmazó eszközben kell rögzítenünk. Például, ha a ttyd5 eszköz sebességét fixen 57600 bps-ra akarjuk beállítani, akkor írjuk be ezt:

```
# stty -f /dev/ttyd5.lock 57600
```

Ezután ha egy alkalmazás megnyitja a ttyd5 eszközt és megpróbálja a port sebességét átállítani, akkor az továbbra is 57600 bps marad.

A kezdeti és a zárolt állapotot képezõ eszközöket általában csak a `root` felhasználó számára szabad írhatóvá tenni.

26.3. Terminálok

A terminálok olyankor kínálnak kényelmes és költséghatékony hozzáférést a FreeBSD rendszerünkhöz, amikor sem a gép konzolját, sem pedig a hozzá tartozó hálózatot nem érjük el. Ebben a szakaszban olvashatjuk, miként kell terminálokat használni FreeBSD alatt.

26.3.1. A terminálok alkalmazásai és típusai

Az eredeti UNIX® rendszereknek nem voltak konzoljaik. Ehelyett az emberek a soros portokra csatlakoztatott terminálokon keresztül jelentkeztek be és így futtattak rajtuk programokat. Ez nagyon hasonlít ahhoz, mint amikor egy modem és egy terminálprogram felhasználásával betárcsázunk egy távoli gépre és vele szöveges módban dolgozunk.

Napjaink személyi számítógépein azonban találhatunk már akár nagy felbontású megjelenítéssel megáldott konzolokat is, habár a soros porton keresztüli bejelentkezés lehetõsége még mind a mai napig elérhetõ a legtöbb UNIX®-alapú rendszerben. Ez alól a FreeBSD sem kivétel. Ha rákötünk egy terminált a gépünk egyik üres soros portjára, akkor a megszokott módon képesek vagyunk bejelentkezni a rendszerbe és futtatni bármilyen szöveges programot, hasonlóan ahhoz, ahogy azt a

konzolban vagy az X Window Systemben egy `xterm` ablakban megtehetjük.

Ha egy irodában vagyunk, akkor egy FreeBSD rendszerre több terminált is kapcsolhatunk, melyek az alkalmazottak asztalain foglalnak helyet. Otthoni használat esetén egy kiöregedett számítógép, például egy régi IBM PC vagy egy Macintosh® is ráköthetõ egy gyorsabb FreeBSD rendszerre. Ennek segítségével az egyébként egyfelhasználós számítógépünket egy valódi többfelhasználós rendszerré alakíthatjuk.

A FreeBSD esetén háromféle terminálról beszélhetünk:

- A buta (dumb) terminálok
- A terminálként funkcionáló személyi számítógépek
- Az X terminálok

A most következõ alszakaszokban ezeket fejtjük ki részletesebben.

A buta terminálok

A buta terminál alatt olyan speciálizált eszközt értünk, amellyel soros vonalon keresztül csatlakozunk számítógépekhez. Azért nevezik ezeket "butának", mert csupán annyi számítási teljesítményt zsúfoltak beléjük, hogy szöveget legyenek képesek küldeni, fogadni és megjeleníteni. Semmilyen program nem képes rajtuk futni. Helyette az a számítógép fogja a szövegszerkesztõt, fordítóprogramot, levelezõ klienst, játékot és a többit futtatni, amelyre vele kapcsolódtunk.

A buta terminálóknak többszáz, különbözõ gyártmányú fajtája létezik. Ilyenek például a Digital Equipment VT-100 vagy a Wyse WY-75 típusú termináljai. A FreeBSD szinte mindegyiküket ismeri. Egyes drágább terminálok még grafikus megjelenítésre is képesek, de ezeket a lehetõségeket csak bizonyos szoftverek tudják ténylegesen kihasználni.

A buta terminálok leginkább olyan munkahelyeken terjedtek el, ahol az alkalmazottaknak nincs szükségük grafikus alkalmazások, tehát például az X Window System használatára.

Személyi számítógépek mint terminálok

Ha egy buta terminál csupán szöveg küldésére, fogadására és megjelenítésére képes, akkor bármelyik személyi számítógép utána tudja mindezt csinálni. Ehhez mindössze egy megfelelõ kábelre és az adott gépen futó *terminál emulációs* szoftverre van szükségünk.

Az ilyen fajta megoldás nagyon elterjedt az otthoni használat esetén. Például, ha valamelyik családtagunk éppen szorgalmasan dolgozik a FreeBSD rendszerkonzolján, akkor a rákapcsolt terminálon keresztül még mi magunk is el tudunk végezni valamennyi szöveges felületet igénylõ munkát.

Az alap FreeBSD rendszerben legalább két segédprogram használható a soros vonali kapcsolaton keresztüli munkára: a cu(1) és a tip(1).

Egy FreeBSD rendszerû kliensrõl így tudunk csatlakozni egy másik rendszerre:

```
# cu -l soros-vonali-eszköz
```

Ahol a "soros-vonali-eszköz" a rendszerünkben a soros portot jelölõ speciális eszköz neve. Az ilyen eszközök neve /dev/cuadN.

Az eszköz nevében az "N"-es rész a soros port sorszámát adja meg.

A FreeBSD-ben az eszközök sorszámozása nullától kezdõdik, nem pedig egytõl (ellentétben tehát azzal, ahogy azt az MS-DOS® rendszerekben és leszármazottaikban már megszokhattuk). Ez azt jelenti, hogy amit az MS-DOS® alapú rendszerekben COM1-nek hívnak, az a FreeBSD-ben általában a /dev/cuad0.

Egyes emberek más, többnyire a Portgyûjteménybõl is elérhetõ programokat szeretnek inkább használni. A portok között találhatunk elég sok olyan szoftvert, amely a cu(1) és a tip(1) programokhoz hasonlóan mûködik. Ilyen például a comms/minicom.

Az X terminálok

Az X terminálok a terminálok közül a legfejlettebbek. Általában nem is soros porton, hanem hálózaton, például Etherneten keresztül csatlakoznak. Természetesen nem csak szöveges alkalmazásokat, hanem lényegében bármilyen X alkalmazást képesek megjeleníteni.

Az X terminálokról itt most csak a teljesség kedvéért szólunk, de ebben a fejezetben *nem* szándékozunk tárgyalni az X terminálok csatlakoztatását, beállítását és használatát.

26.3.2. Beállítás

Ebben a fejezetben ismertetjük mindazt, ami ahhoz kell, hogy a FreeBSD rendszerünkön engedélyezni tudjuk a terminálon keresztüli bejelentkezéseket. Feltételezzük, hogy a rendszermagunk támogatja a terminálok által használt soros portokat, illetve, hogy ezeket már csatlakoztattuk is.

Ha visszagondolunk a A FreeBSD rendszerindítási folyamatare, akkor eszünkbe juthat, hogy a rendszer indításakor az `init` nevû program felelõs az összes futó program irányításáért és inicializálódásáért. Az `init` egyik feladata, hogy beolvassa az /etc/ttys állományt és neki megfelelõen az elérhetõ terminálokon elindítsa a `getty` programot. A `getty` felelõs a bejelentkezéshez szükséges azonosító beolvasásáért és a `login` program elindításáért.

Ennek megfelelõen tehát, ha a FreeBSD rendszerünkön terminálokat akarunk beállítani, akkor ehhez a következõ lépéseket kell megtennünk `root` felhasználóként:

1. Az /etc/ttys állományba vegyünk fel egy bejegyzést a soros porthoz tartozó /dev könyvtárbeli eszközhöz, ha még nem szerepelne benne.
2. A porthoz adjuk meg a `/usr/libexec/getty` programot, majd hozzá az /etc/gettytab állományból válasszuk ki a megfelelõ *getty* típust.
3. Adjuk meg a terminál alapértelmezett típusát.
4. Állítsuk a portot "on" (bekapcsolt) állapotúra.

5. Adjuk meg, hogy a port "secure" (biztonságos) legyen-e.
6. Mondjuk meg az `init` programnak, hogy olvassa újra az /etc/ttys állományt.

A másik lépés kiegészítõ lépéseként az /etc/gettytab állományban mi magunk is létrehozhatunk egy saját *getty* típust. A fejezetben ehhez ugyan nem adunk segítséget, de ha érdekel minket a téma, akkor ezzel kapcsolatban a gettytab(5) és getty(8) man oldalakat érdemes elolvasni.

Egy bejegyzés felvétele az /etc/ttys állományba

Az /etc/ttys állományban találhatjuk meg az összes portot, ahonnan a FreeBSD rendszerünk engedélyezi a bejelentkezést. Például a ttyv0, az elsõ virtuális konzol is szerepel benne. Ezen a bejegyzésen keresztül tudunk bejelentkezni a konzolra. Ebben az állományban találjuk meg még a többi virtuális konzol, soros port és pszeudoterminál bejegyzéseit is. A rögzített terminálok esetén egyszerûen csak adjuk meg a soros porthoz tartozó /dev könyvtárbeli eszközt a /dev elõtag nélkül (így például a /dev/ttyv0 ttyv0 néven fog megjelenni).

Az alap FreeBSD telepítésben egy olyan /etc/ttys állomány található, amely tartalmazza az elsõ négy soros portot, a ttyd0 eszköztõl kezdve a ttyd3 eszközig. Ha tehát ezekre a portokra csatlakoztatnunk egy terminált, akkor már nem kell egy újabb bejegyzést felvennünk hozzájuk.

Példa 29. Terminálok felvétele az /etc/ttys állományba

Tegyük fel, hogy két eszközt szeretnénk a rendszerünkhöz csatlakoztatni: egy Wyse-50-es terminált és egy régi 286-os IBM PC-t, amelyen a Procomm terminálszoftverrel emulálunk egy VT-100-as terminált. A Wyse terminált a második soros portunkra kötjük, míg a 286-ost a hatodik soros portra (például egy többportos soros vonali kártyán). A nekik megfelelõ /etc/ttys állománybeli bejegyzések így fognak kinézni:

```
ttyd1  "/usr/libexec/getty std.38400"  wy50  on  insecure
ttyd5   "/usr/libexec/getty std.19200"  vt100  on  insecure
```

- Az elsõ mezõben általában a terminálhoz tartozó eszközt nevezzük meg, amely a /dev könyvtárban található.
- A második mezõ a vonalhoz tartozó végrehajtandó parancs, ami általában a getty(8). A `getty` mûködésbe helyezi és megnyitja a vonalat, beállítja a sebességét, bekéri a felhasználó nevét, majd elindítja a login(1) programot.A `getty` program egy (opcionális) paramétert fogad el a parancssorában, ami a *getty* típusa. Egy ilyen *getty* típus szabja meg a terminálhoz tartozó vonal jellemzõit, például az adatátviteli sebességet és a paritást. A `getty` ezeket a jellemzõket az /etc/gettytab állományból olvassa be.A /etc/gettytab egyaránt tartalmaz bejegyzéseket a régi és új típusú terminálokhoz. Az `std` szöveggel kezdõdõ bejegyzések szinte majdnem minden esetben mûködnek a hardveres terminálokkal. Az ilyen bejegyzések figyelmen kívül hagyják a paritást. 110 és 115 200 bps között minden adatátviteli sebességhez tartozik egy-egy `std` bejegyzés. Természetesen ebbe az állományba akár a saját bejegyzéseinket is elkészíthetjük. A gettytab(5) man oldal nyújt ehhez átfogó segítséget.Amikor az/etc/ttys állományban megadjuk a *getty* típusát, akkor ellenõrizzük, hogy a beállításai megfelelnek a terminálénak.A példánknál maradva: a Wyse-50 nem

használ paritást és 38 400 bps-en üzemel. A 286-os gép szintén nem dolgozik paritással és 19200 bps-sel kapcsolódik.

- A harmadik mezõben adjuk meg általában a vonalra csatlakozó terminál típusát. Ez a betárcsázós portok esetében többnyire az `unknown` vagy a `dialup`, mivel ezeken keresztül a felhasználók gyakorlatilag szinte bármilyen típusú terminállal vagy szoftverrel be tudnak jelentkezni. A hardveres terminálóknál a terminál típusa azonban nem változik, ezért a termcap(5) adatbázisban keressük ki a nekik megfelelõt és adjuk meg ebben a mezõben.A példánkban a Wyse-50 egy valós terminháltípust használ, miközben a 286-oson futó Procomm egy VT-100-as típusú terminált emulál.
- A negyedik mezõ azt mondja meg, hogy a port engedélyezett-e vagy sem. Ha itt a `on` értéket adjuk meg, akkor az `init` elindítja a második mezõben szereplõ `getty` programot. Ha viszont itt az `off` szerepel, akkor a `getty` nem fog elindulni, így ezen a porton be sem fogunk tudni jelentkezni.
- Az utolsó mezõben a port megbízhatóságát kell megjelölnünk. Ha biztonságosnak (`secure`) állítjuk be a portot, akkor rajta keresztül a `root` (vagy bármelyik nullás felhasználói azonosítóval rendelkezõ) felhasználó be tud jelentkezni. Amikor viszont nem biztonságos (`insecure`), akkor elõször egy normál felhasználóval kell bejelentkeznünk, majd a su(1) programmal vagy egy hozzá hasonló megoldással kell rendszeradminisztrátorrá válnunk.Leginkább az `insecure` beállítást javasoljuk, még hét lakat alatt õrzött terminálok esetében is. Valójában sokkal egyszerûbb bejelentkezni, majd kiadni egy `su` parancsot, ha netalán rendszeradminisztrátori jogosultságokra lenne szükségünk.

A `init` utasítása az /etc/ttys újraolvasására

Miután az /etc/ttys állományban elvégeztük a megfelelõ módosításokat, a konfigurációs állomány újraolvasásához küldjünk egy SIGHUP (bontás) jelzést az `init` programnak. Mint például:

```
# kill -HUP 1
```

Mivel mindig az `init` indul el elsõként a rendszerben, ezért a hozzá tartozó azonosító az 1 lesz.

Ha mindent jól állítottunk be, a kábelek is a helyükön vannak és a terminálokat is bekapcsoltuk, akkor minden terminálhoz elindul egy `getty` program, és mindegyikõjükön megjelenik a bejelentkezõ képernyõ.

26.3.3. A terminálokkal kapcsolatos hibajelenségek

Olykor hiába igyekszünk a lehetõ legaprólékosabban ügyelni minden apró részletre, könnyen elõfordulhat, hogy valamiért a terminál mégsem mûködik rendesen. Következzen most egy lista néhány ismert tünetrõl és azok javasolt gyógymódjairól.

Nem jelenik meg a bejelentkezõ képernyõ

Ellenõrizzük, hogy a terminált rendesen csatlakoztattuk és áram alá helyeztük. Amikor egy

személyi számítógépet használunk terminálnak, akkor nézzük meg, hogy a terminál emulációs program a megfelelõ soros porton fut.

Vizsgáljuk meg, hogy a kábel mind a két vége pontosan illeszkedik a portokba. Gyõzõdjünk meg róla, hogy valóban a megfelelõ típusú kábelt használjuk.

Nézzük meg, hogy a terminál és a FreeBSD is ugyanazon az adatátviteli sebességen és paritási beállítással megy. Ha képernyõvel rendelkezõ terminálunk van, akkor a kontrasztot és fényerõsséget is ellenõrizzük. Ha nyomtatós terminálunk van, akkor vizsgáljuk meg a papír és a tinta állapotát.

Gyõzõdjünk meg róla, hogy a `getty` valóban fut és rendesen kiszolgálja a terminált. Például a `ps` paranccsal listázzuk ki az összes jelenleg futó programot és keressük meg köztük a `getty` programot:

```
# ps -axww|grep getty
```

Ekkor látnunk kell a terminálhoz tartozó bejegyzést. Például, ha a `getty` második soros portot jelképezõ ttyd1 eszközön fut, és az /etc/gettytab állományból az `std.38400` nevû bejegyzést használja, akkor ez jelenik meg:

```
22189  d1  Is+    0:00.03 /usr/libexec/getty std.38400 ttyd1
```

Amennyiben semmilyen `getty` nem fut, akkor ellenõrizzük, hogy valóban engedélyeztük-e a portot az /etc/ttys állományban. A ttys állomány átírása után ne felejtsük el kiadni a `kill -HUP 1` parancsot sem.

Ha a `getty` fut, de a terminálon továbbra sem látjuk a bejelentkezõ képernyõt, vagy megjelenik, de nem tudunk gépelni, akkor elõfordulhat, hogy a terminál vagy kábel nem támogatja a hardveres kézfogást (handshaking). Próbáljuk meg az /etc/ttys állományban levõ `std.38400` bejegyzést az `3wire.38400` bejegyzésre kicserélni (de utána ne felejtsük el kiadni a `kill -HUP 1` parancsot). A `3wire` nagyon hasonlít az `std` bejegyzéshez, de elhagyja a hardveres kézfogást. A `3wire` alkalmazásakor viszont a puffer telítõdésének megelõzése érdekében próbálkozzunk az adatátviteli sebesség csökkentésével vagy engedélyezzük a szoftveres forgalomirányítást.

Amikor mindenféle szemét jelenik meg a képernyõn

Ellenõrizzük, hogy a FreeBSD és a terminál ugyanazt az adatátviteli sebességet és paritási beállítást használja. Nézzük meg a futó `getty` programokat, és hogy a megfelelõ *getty* típussal mennek-e. Ha nem, módosítsuk az /etc/ttys állományt és adjuk ki a `kill -HUP 1` parancsot.

A karakterek duplán jelennek meg, a jelszó begépelésekor látható

Állítsuk át a terminált (vagy a terminál emulációs szofvert) "half duplex" vagy "local echo" módról "full duplex" módra.

26.4. Betárcsázós szolgáltatások

Amikor egy FreeBSD rendszert akarunk betárcsázós szolgáltatásokhoz beállítani, akkor az nagyon hasonlít a terminálok csatlakoztatásához, azzal a eltéréssel, hogy ilyenkor a terminálok helyett modemekkel kell dolgoznunk.

26.4.1. Külsõ kontra belsõ modemek

A külsõ modemek sokkal kényelmesebbnek tûnnek betárcsázás szempontjából, mivel az ilyenek gyakran a statikus memóriájukban tárolt paraméterek révén tulajdonképpen félig elõre be vannak állítva és sok esetben a fontosabb RS-232 jeleket külön lámpácskákkal mutatják. A villogó lámpák könnyen elkápráztatják a laikusokat, de emellett igen fontosak a modem mûködõképességének megállapításában is.

Ezzel szemben a belsõ modemeken nem található statikus memória, ezért a paramétereik csak DIP kapcsolókkal módosíthatóak. Még ha egy belsõ modemem látunk is lámpákat, akkor sem könnyû figyelni rájuk, mert a gépünk burkolata úgyis eltakarja ezeket.

Modemek és kábelek

Ha külsõ modemet használunk, akkor mindenképpen szükségünk lesz hozzá még egy megfelelõ kábelre is. Egy szabványos RS-232-es soros kábel erre tökéletesen megfelel egészen addig, amíg a normál jeleket így kötötték be rajta:

Táblázat 11. A jelek neve

Rövidítés	Elnevezés
RD	Received Data (fogadott adat)
TD	Transmitted Data (küldött adat)
DTR	Data Terminal Ready (adatterminál kész)
DSR	Data Set Ready (adatbeállítás kész)
DCD	Data Carrier Detect (vonal észlése - az RS-232 fogadást érzékelõ vonala)
SG	Signal Ground (föld)
RTS	Request to Send (küldés kérése)
CTS	Clear to Send (küldés engedélyezése)

A FreeBSD-nek 2400 bps felett a forgalom irányításához az RTS és CTS jelekre van szüksége. A CD jellel állapítja meg, hogy a hívás létrejött vagy a bontották a vonalat, és a DTR jel hozza alapállapotba a modemet a munkamenet befejezése után. Egyes kábelekben nem mindegyik jelet vezették át, így ha például gondjaink akadnak a bejelentkezõ képernyõvel amikor a vonalat bontjuk, akkor érdemes átnéznünk a kábelt.

A többi UNIX®-szerû operációs rendszerhez hasonlóan a FreeBSD is hardveres jelek segítségével igyekszik kideríteni, hogy a hívás megvalósult vagy bontották a vonalat, valamint a hívás befejezése után így bontja a vonalat és állítja vissza a modemet. A FreeBSD igyekszik elkerülni a parancsok

küldését a modem felé, vagy a modem állapotának folyamatos ellenõrzését. Ha már van némi tapasztalatunk a PC-alapú BBS-ek modemes elérését illetõen, akkor valószínûleg értjük ezek okait.

26.4.2. A soros vonali felülettel kapcsolatos megfontolások

A FreeBSD ismeri az NS8250-, NS16450-, NS16550- és NS16550A alapú EIA RS-232C (CCITT V.24) szabványú kommunikációs felületeket. A 8250-es és a 16450-es eszközök egykarakteres pufferrel rendelkeznek. A 16550-es eszközök 16 karakteres puffert tartalmaznak, amellyel jobb teljesítmény érhetõ el. (A sima 16550-esben levõ hibák miatt azonban ez a 16 karakteres puffer nem használható ki rendesen, ezért lehetõleg a 16550A verziót használjuk). Mivel az operációs rendszer részérõl az egykarakteres eszközök jóval több törõdést igényelnek, mint a 16 karakteres eszközök, ezért inkább a 16550A alapú soros felületi kártyákat ajánljuk. Amikor a rendszer egyszerre több soros portot is kezel, vagy erõs terhelés alatt áll, akkor a 16550A alapú kártyákról általában az is elmondható, hogy kisebb hibával dolgoznak.

26.4.3. Egy gyors áttekintés

Ahogy arról már a terminálok esetében szó esett, az `init` az összes betárcsázós kapcsolathoz tartozó soros porthoz elindít egy `getty` programot. Például, ha a modemet a /dev/ttyd0 eszközre kapcsoltuk, akkor a `ps ax` parancs kimenetében ezt láthatjuk:

```
4850 ??  I      0:00.09 /usr/libexec/getty V19200 ttyd0
```

Amikor egy felhasználó felhívja a modemet és az kapcsolódik, akkor a modem egy CD (Carrier Detect) jelet küld. A rendszermag ekkor tudomásul veszi a vonal észlelését és a `getty` segítségével megindítja a kommunikációt. A `getty` egy `login:` szöveget küld át a vonalhoz megadott sebességgel. A `getty` elkezdi figyelni, hogy a értelmes karakterek érkeznek-e vissza, és egy átlagos konfigurációban, ha ezt szemétnek találja (mert például a modem nem a `getty` számára beállított sebességgel csatlakozott), akkor megpróbálja egészen addig hangolni a vonal sebességét, amíg feldolgozásra alkalmas karaktereket nem kap.

Miután a felhasználó megadta a felhasználói nevét, a `getty` elindítja a /usr/bin/login programot, amely befejezi a beléptetést a felhasználó jelszavának bekérésével és annak elfogadása esetén a hozzá tartozó parancsértelmezõ elindításával.

26.4.4. A konfigurációs állományok

FreeBSD rendszerünkben a betárcsázós kapcsolatok engedélyezéséhez az /etc könyvtárban három állomány módosítására lesz szükségünk. Közülük az elsõ, az /etc/gettytab a /usr/libexec/getty démon beállításait tartalmazza. A második, az /etc/ttys az /sbin/init számára mondja meg, hogy melyik tty eszközökhöz tartozik `getty`. Végezetül a portok inicializálásához kötõdõ beállításokat az /etc/rc.d/serial szkriptben kell megadnunk.

Két "iskola" jött létre aszerint, hogy UNIX® alatt hogyan használják a betárcsázós modemeket. Az egyik csoport úgy szereti beállítani a modemeit és rendszerit, hogy a távoli felhasználó által választott sebességtõl függetlenül a számítógép és a modem közti RS-232 felület egy fix sebességen fut. Ennek a beállításnak megvan az az elõnye, hogy a távoli felhasználó ilyenkor szinte azonnal megkapja a bejelentkezõ képernyõt. A hátránya viszont, hogy ebben az esetben a rendszer nem

ismeri a felhasználó valódi adatátviteli sebességét, ezért az olyan teljes képernyõs alkalmazások, mint például az Emacs, nem lesznek képesek a lassabb kapcsolatokhoz szabni a megjelenítésüket.

A másik csoport a modemek RS-232-es felületét a távoli felhasználó kapcsolódási sebessége szerint állítja be. Így például egy V.32bis (14,4 Kbps) kapcsolat esetén a modemhez tartozó RS-232 felület 19,2 Kbps-on fog menni, miközben a 2400 bps sebességû kapcsolatokhoz egy vele azonos sebességû RS-232-es felület fog tartozni. Mivel a `getty` nem képes kommunikálni a modemek által lejelentett csatlakozási sebességen, ezért úgy próbálja azt megállapítani, hogy elküldi a `login:` szöveget az alap sebességgel, majd figyeli a válaszul érkezõ karaktereket. Ha a felhasználó ilyenkor szemetet lát, akkor feltételezik, hogy addig fogja nyomkodni az `Enter` billentyût, amíg valami értelmes szöveget meg nem lát. Amikor az adatátviteli sebesség eltér, akkor a `getty` ebbõl csupán csak annyit vesz észre, hogy a felhasználó "szemetet" küld, ezért egy újabb sebességgel megpróbálja megint elküldeni a `login:` szöveget. Hivatalosan ez a folyamat ismétlõdik orrvérzésig, de általában csak egy-két billentyût kell leütni a megfelelõ beállításokhoz. Nyilvánvaló, hogy ilyenkor a bejelentkezés messze nem olyan zavartalan, mint a "rögzített sebességû" esetben, de így a lassabb kapcsolattal rendelkezõ felhasználók is jobb használatóságot kapnak a teljes képernyõs programokkal.

Ebben a szakaszban egy valamennyire kiegyensúlyozott beállítást igyekszünk bemutatni, de részben elfogunk hajlani abban az irányba, amikor a modem a kapcsolat sebességét követi.

/etc/gettytab

A /etc/gettytab egy termcap(5)-szerû állomány, amely a getty(8) beállításait tartalmazza. A gettytab(5) man oldalon olvashatunk az állomány pontos felépítésérõl és benne felsorolt beállításokról.

A rögzített sebességû beállítás

Ha a modem kommunikációs sebességét rögzíteni akarjuk, akkor ehhez többnyire semmit sem kell megváltoztatnunk az /etc/gettytab állományban.

Az alkalmazkodó sebességû beállítás

Az /etc/gettytab állományban létre kell hoznunk egy olyan bejegyzést, amelyen keresztül a `getty` tudni fogja, hogy milyen sebességeken akarjuk használni a modemet. Ha egy 2400 bps sebességû modemünk van, akkor hozzá a már meglevõ `D2400`-as bejegyzést kell használnunk.

```
#
# A gyors betárcsázós terminálokhoz íme egy 2400/1200/300-as váltás
# (bárhonnan kezdõdhet):
#
D2400|d2400|Fast-Dial-2400:\
        :nx=D1200:tc=2400-baud:
3|D1200|Fast-Dial-1200:\
        :nx=D300:tc=1200-baud:
5|D300|Fast-Dial-300:\
        :nx=D2400:tc=300-baud:
```

Ha ennél gyorsabb modemünk van, akkor már mindenképpen fel kell vennünk hozzá egy új

bejegyzést az /etc/gettytab állományba. Ezzel a beállítással egy 14,4 Kbps sebességû modemet tudunk legfeljebb 19,2 Kbps-en használni:

```
#
# Kiegészítések egy V.32bis modemhez:
#
um|V300|High Speed Modem at 300,8-bit:\
        :nx=V19200:tc=std.300:
un|V1200|High Speed Modem at 1200,8-bit:\
        :nx=V300:tc=std.1200:
uo|V2400|High Speed Modem at 2400,8-bit:\
        :nx=V1200:tc=std.2400:
up|V9600|High Speed Modem at 9600,8-bit:\
        :nx=V2400:tc=std.9600:
uq|V19200|High Speed Modem at 19200,8-bit:\
        :nx=V9600:tc=std.19200:
```

Ennek eredménye egy 8 bites, paritásmentes kapcsolat lesz.

A fenti példában a kommunikációt 19,2 Kbps-en (V.32bis kapcsolaton) kezdjük, majd utána haladunk végig a 9600 bps (V.32), 2400 , 1200 bps és 300 bps sebességû kapcsolatokon, majd vissza ismét a 19,2 Kbps-re. Az adatátviteli sebesség ilyen típusú váltogatását az `nx=` ("next table", azaz "következõ táblázat") tulajdonság segítségével valósítják meg. Minden sorban látható még egy `tc=` ("table continuation", vagyis "a táblázat folytatása") bejegyzés is, amivel az adott adatátviteli sebesség "szabványos" beállításait adjuk meg.

Ha egy 28,8 Kbps sebességû modemünk van és/vagy egy 14,4 Kbps sebességû modemen akarunk tömörítést használni, akkor a 19,2 Kbps-nél nagyobb kommunikációs sebességet kell használnunk. Íme egy olyan gettytab. ami 57,6 Kbps-rõl indít:

```
#
# A V.32bis vagy V.34 modemekhez kiegészítés,
# 57,6 Kbps-rõl indulunk:
#
vm|VH300|Very High Speed Modem at 300,8-bit:\
        :nx=VH57600:tc=std.300:
vn|VH1200|Very High Speed Modem at 1200,8-bit:\
        :nx=VH300:tc=std.1200:
vo|VH2400|Very High Speed Modem at 2400,8-bit:\
        :nx=VH1200:tc=std.2400:
vp|VH9600|Very High Speed Modem at 9600,8-bit:\
        :nx=VH2400:tc=std.9600:
vq|VH57600|Very High Speed Modem at 57600,8-bit:\
        :nx=VH9600:tc=std.57600:
```

Ha lassú a processzorunk, vagy a rendszerünk túlságosan terhelt és nincs 16550A típusú soros portunk, akkor 57,6 Kbps-en `sio`"silo" hibák keletkezhetnek.

/etc/ttys

Az /etc/ttys állomány beállításáról már a Terminálok felvétele az /etc/ttys állományba adott képet. Ez a modemek esetében sem tér el különösebben, habár a `getty` programnak más terminéltípust és -beállításokat kell átadnunk. Akár rögzített, akár alkalmazkodó sebességet akarunk beállítani, ennek általános alakja az alábbi:

```
ttyd0   "/usr/libexec/getty xxx"   dialup on
```

A sorban látható elsõ elem a megfelelõ speciális eszköz neve - jelen esetben ez a ttyd0, amely a /dev/ttyd0 eszközre vonatkozik és ezt fogja a `getty` figyelni. A második elem, vagyis a `"/usr/libexec/getty xxx"` (ahol a *xxx* helyére kell beírni a megfelelõ gettytab állománybeli bejegyzést nevét) lesz az a parancs, amelyet az `init` meghív. A harmadik elem, a `dialup` a terminálok alapértelmezett típusa. A negyedik paraméter, az `on` jelzi az `init` programnak, hogy aktiválja a vonalat. A sorban megjelenhetne továbbá még egy ötödik paraméter is, a `secure`, de ezt csak olyan terminálok esetében érdemes megadni, amelyek fizikailag megbízhatóak (például a rendszerkonzol).

Az alapértelmezett terminéltípus (vagyis a fenti példában a `dialup`) a helyi beállításoktól függ. A betárcsázós vonalak esetében hagyományosan a `dialup` a terminál alapértelmezett típusa, amit aztán a felhasználók a bejelentkezéskor lefutó szkriptjeiken keresztül a automatikusan át tudnak állítani a nekik megfelelõ terminálra. A szerzõ saját rendszerében azonban inkább a `vt102` terminéltípust volt érdemes megadni alapértelmezettként, mivel ott a felhasználók csak ilyen típusú terminálokat használnak.

Miután az /etc/ttys állományban elvégeztük a szükséges módosításokat, egy HUP jelzéssel figyelmeztessük az `init` programot az újbóli beolvasására. Ehhez a következõ parancs ajánlott:

```
# kill -HUP 1
```

Ha még csak állítjuk be elõször a rendszerünket, akkor az `init` figyelmeztetése elõtt legyünk türelmesek, és várjuk meg, amíg a modemek befejezik az inicializálást és kapcsolódnak a vonalakra.

A rögzített sebességû beállítás

A rögzített sebesség beállításánál a ttys állományban a `getty` paramétereként egy szintén rögzített sebességû bejegyzést kell megadnunk. Például az olyan modemeknél, ahol a sebességet 19,2 Kbps-re rögzítjük, a ttys így fog kinézni:

```
ttyd0   "/usr/libexec/getty std.19200"   dialup on
```

Amennyiben a modemünk nem ezen a sebességen üzemelne, akkor az `std.sebesség` paramétert használjuk az `std.19200` helyett. Elõtte azonban ne felejtsük el ellenõrizni, hogy a megadott típus szerepel-e az /etc/gettytab állományban.

Az alkalmazkodó sebességû beállítás

Az alkalmazkodó sebességû beállításnál a ttys állományban az /etc/gettytab állományból a megfelelõ "auto-baud" (sic) kell megadnunk. Például, ha modemünk kezdõsebessége 19,2 Kbps (és a gettytab ehhez tartalmaz egy `V19200` nevû bejegyzést), akkor a ttys így fog kinézni:

```
ttyd0   "/usr/libexec/getty V19200"   dialup on
```

/etc/rc.d/serial

A gyorsabb, mint például a V.32, V.32bis és V.34 modemeknél meg kell adnunk a hardveres forgalomirányítás (`RTS/CTS`) használatát is. Az /etc/rc.d/serial állományban tudjuk megadni a FreeBSD rendszermagban a vonal használatához szükséges vezérlési beállításokra vonatkozó `stty` parancsokat.

Például állítsuk be az 1-es sorszámú (vagyis a COM2) soros porton a `crtscts termios` beállítást a behíváshoz és a híváshoz használt eszközök inicializálásakor. Ehhez a következõ sorokat kell felvennünk az /etc/rc.d/serial állományba:

```
# A soros portok kezdeti beállításai:
stty -f /dev/ttyd1.init crtscts
stty -f /dev/cuad1.init crtscts
```

26.4.5. A modemek beállításai

Ha olyan modemeink vannak, amelyek paramétereit egy statikus memóriában tárolták le, akkor ezek beállításához egy terminálprogramot kell használnunk (amilyen például MS-DOS® alatt a Telix vagy FreeBSD alatt a `tip`). A modemet a `getty` programnak megadott kezdeti sebességen csatlakoztassuk és az alábbi elvárások alapján állítsuk be a paramétereit:

- Kapcsolódáskor CD jelzése.
- Mûködéskor DTR jelzése. A DTR küldésekor bontsa a vonalat és hozza alapállapotba a modemet.
- CTS vezérlésû kimenõ adatforgalom.
- A XON/XOFF forgalomvezérlés tiltása.
- RTS vezérlésû bejövõ adatforgalom.
- Csendes mód (ne adjon értesítést az eredményekrõl).
- A parancsokat ne írja vissza.

A modemhez tartozó dokumentációban kell utánajárnunk, hogy milyen parancsok és/vagy DIP kapcsolók átállításával lehet mindezeket elérni.

Például, ha a fenti paramétereket egy U.S. Robotics® Sportster® 14400-as külsõ modem esetében a következõ neki kiküldött paranccsal lehet beállítani:

```
ATZ
```

```
AT&C1&D2&H1&I0&R2&W
```

Ilyenkor még akár más egyéb paramétereket is beállíthatunk, például a V.42bis és/vagy az MNP5 tömörítést.

Az U.S. Robotics® Sportster® 14400 külsõ modemen ezenkívül még találunk néhány DIP kapcsolót is. Az ilyen modemek esetében például ezeket a beállításokat tudjuk használni:

- 1. kapcsoló: FEL - normális DTR
- 2. kapcsoló: N/A (verbális/numerikus eredményjelzõ kódok)
- 3. kapcsoló: FEL - az eredményjelzõ kódok küldésének tiltása
- 4. kapcsoló: LE - nem küldi vissza a parancsokat
- 5. kapcsoló: FEL - automatikus válasz
- 6. kapcsoló: FEL - normális Carrier Detect
- 7. kapcsoló: FEL - a memóriában tárolt alapértelmezések betöltése
- 8. kapcsoló: N/A (intelligens/buta mód)

A modemeknél az eredményjelzõ kódok kikapcsolása/letiltása ezért fontos, mert így el tudunk kerülni az olyan problémákat, hogy a `getty` tévesen egy `login:` promptot küld a parancs módban levõ modemnek, amikor az visszaküldi a parancsot és az eredmény kódját. Ennek eredménye egy hosszúra nyúló, zavaros társalgás lesz a `getty` és a modem között.

A rögzített sebességû beállítás

A rögzített sebességû konfiguráció használata esetén úgy kell beállítanunk a modemet, hogy a konkrét adatátviteli sebsségtõl függetlenül is egy állandó sebességû kapcsolat álljon fenn a számítógép és a modem között. A U.S. Robotics® Sportster® 14400-as külsõ modem esetében a most következõ parancsokkal tudjuk rögzíteni a kapcsolat sebességét:

```
ATZ
AT&B1&W
```

Az alkalmazkodó sebességû beállítás

Amikor változó sebességû konfigurációval dolgozunk, akkor a modemet úgy kell beállítani, hogy a bejövõ hívásnak megfelelõ adatátviteli sebességre váltson a soros portján. A U.S. Robotics® Sportster® 14400-as külsõ modem esetében az alábbi parancsokkal rögzítjük a modemnek küldött hibamentesített parancsok sebességét, miközben engedélyezzük, hogy a soros port sebessége változhasson a nem hibamentesített kapcsolatoknál:

```
ATZ
AT&B2&W
```

A modem beállításainak ellenõrzése

A legtöbb nagysebességû modem biztosít valamilyen lehetõséget arra, hogy emberi formában is le tudjuk kérdezni a belsõ mûködésének paramétereit. A U.S. Robotics® Sportster® 14400-as külsõ modem esetében az `ATI5` parancs a statikus memóriában tárolt beállításokat mutatja meg. A modem valós mûködési paramétereit (amit ugyebár befolyásolnak a DIP kapcsolók állásai is) viszont az `ATZ` majd `ATI4` parancsok küldésével tudjuk lekérni.

Ha azonban másmilyen márkájú modemünk lenne, akkor a modem leírásában próbáljunk tájékozódni arról, miként tudjuk a modem beállításait ellenõrizni.

26.4.6. Hibaelhárítás

Ebben a szakaszban bemutatunk néhány lépést, amelyeken keresztül ellenõrizhetjük a rendszerünkhöz csatlakoztatott modemet.

A FreeBSD rendszer ellenõrzése

Csatlakoztassuk a modemet a FreeBSD rendszerre, indítsuk be a gépet, majd ezután figyeljük a modemünk állapotát jelzõ lámpákat, hogy közülük a DTR világít-e, amikor a `login:` felirat megjelenik a rendszerkonzolon. Amennyiben erre a válasz igen, akkor az arra utal, hogy a FreeBSD a hozzá tartozó kommunikációs porton elindította a megfelelõ `getty` programot és a modem várja a hívásokat.

Amikor viszont a DTR lámpa nem világít, a konzolon keresztül jelentkezzünk be a FreeBSD rendszerbe és adjuk ki egy `ps ax` parancsot, amivel így ellenõrizni tudjuk, hogy a porthoz tartozó `getty` elindult. A futó programok között tehát valami ilyesmit kell majd látnunk:

```
114 ??  I      0:00.10 /usr/libexec/getty V19200 ttyd0
115 ??  I      0:00.10 /usr/libexec/getty V19200 ttyd1
```

Ha viszont például ezt látjuk:

```
114 d0  I      0:00.10 /usr/libexec/getty V19200 ttyd0
```

és modem még nem fogadott hívást, akkor ez azt jelenit, hogy a `getty` megnyitotta a kommunikációs csatornát. Ez utalhat egyaránt egy hibás kábelre vagy a modem helytelen beállítására, mivel a `getty` egészen addig nem lesz képes megnyitni az adott portot, amíg a modem vissza nem küld neki egy CD (Carrier Detect) jelet.

Ha a listában az adott ttydN eszközhöz semmilyen `getty` programot nem találunk, akkor újra nézzük át az /etc/ttys állományban szereplõ bejegyzéseket, mert elõfordulhat, hogy azokban vétettünk valamilyen hibát. Emellett még a /var/log/messages naplóban is érdemes utánanézni, hátha az `init` vagy a `getty` küldött valamilyen hibáról értesítést. Ha még ezek után sem találunk semmit, akkor megint kezdjük el keresni hibákat, hiányzó bejegyzéseket vagy eszközöket az /etc/ttys, /etc/gettytab és a megfelelõ /dev/ttydN állományokban.

A betárcsázás kipróbálása

Próbáljunk meg bejutni a rendszerünkbe. Ehhez a távoli rendszeren ne felejtsük el beállítani a 8 bites adatátvitelt és az 1 stopbitet, illetve a paritást kikapcsolni. Ha erre közvetlenül nem kapunk egy bejelentkezési képernyõt vagy csak szemét jelenik meg, akkor kb. másodpercenként egyszer nyomjuk le az `Enter` billentyût. Ha még ezután sem látjuk a bejelentkezési képernyõt felbukkani, akkor próbáljunk kiküldeni egy `BREAK` parancsot. Ha a híváshoz nagysebességû modemet használunk, akkor próbáljuk meg a modem sebességét rögzíteni és úgy tárcsázni (ezt például a U.S. Robotics® Sportster® modemnél az `AT&B1` paranccsal tudjuk elérni):

Ha viszont még ezek után sem kapjuk meg a bejelentkezõ képernyõt, akkor a /etc/gettytab állományban megint nézzük át az összes beállítást:

- Az /etc/ttys állományban megadott alaptulajdonság neve egyezik az /etc/gettytab állományban találhatóval.
- Mindegyik `nx=` bejegyzés után egy másik gettytab tulajdonság neve jön.
- Mindegyik `tc=` bejegyzés után egy másik gettytab tulajdonság neve következik.

Ha hívunk, de a FreeBSD rendszerünkre kapcsolt modem továbbra sem veszi fel, akkor a modem beállításai között ellenõrizzük, hogy a DTR jel küldésekor a modem fogadja-e a hívást. Ha úgy tûnik, hogy a modem minden ezzel kapcsolatos beállítása stimmel, akkor nézzük meg, hogy a modem lámpái közül a DTR világít-e (már ha van ilyen).

Ha mindent többször is végignéztünk és még mindig nem leljük a megoldást, akkor tartsunk egy kis szünetet és térjünk vissza a problémához késõbb. Ha még ezután sem tudjuk mûködésre bírni, akkor küldjünk egy levelet a FreeBSD general questions levelezési lista címére, amelyben leírjuk a modemünket és a vele kapcsolatos problémát, és a lista tagjai majd megpróbálnak nekünk segíteni.

26.5. A betárcsázós szolgáltatások használata

A következõkben arra vonatkozóan igyekszünk tanácsokat adni, amikor mi magunk akarunk modemmel csatlakozni valamilyen számítógéphez. Ezek tehát olyan esetekben hasznosak, amikor egy távoli géppel akarunk terminálkapcsolatot létesíteni.

A BBS-ek használatára is érvényes.

Ez ilyen típusú kapcsolatok kifejezetten hasznosak tudnak lenni olyan esetekben, amikor az interneten el akarunk érni egy állományt, de gondjaink akadtak a PPP használatával. Ha például egy állományt akarunk letölteni, de a PPP valamiért nem mûködik, akkor ezt a terminál alapú kapcsolaton keresztül is meg tudjuk tenni. Ilyenkor egy zmodem segítségével tudjuk áttölteni a számítógépünkre.

26.5.1. A gyári Hayes-modem erre nem alkalmas, mihez tudunk vele kezdeni?

A `tip` man oldala valójában már nem is teljesen aktuális, ugyanis tartalmaz egy beépített Hayes-tárcsázót. Úgy tudjuk engedélyezni, ha az /etc/remote állományban megadjuk az `at=hayes` beállítást.

A Hayes-eszközök meghajtója nem elég ügyes ahhoz, hogy felismerje az újabb modemek által

felkínált fejlettebb lehetõségeket - például a BUSY, NO DIALTONE vagy a CONNECT 115200 üzenetek csak megzavarják. Ezért a tip használata során kapcsoljuk ki ezeket az üzeneteket (az ATXO&W paranccsal).

Emellett még érdemes tudni, hogy a tip a híváskor 60 másodpercig vár. A modemünkön ennél kisebb idõt kell beállítanunk, máskülönben a tip azt hiszi, hogy valamilyen kommunikációs probléma merült fel. Ehhez próbálkozzunk az ATS7=45&W paranccsal.

26.5.2. Hogyan adjuk meg ezeket az AT parancsokat?

Az /etc/remote állományban hozzunk létre egy "direct" bejegyzést. Például, ha a modemünk az elsõ soros porton, vagyis a /dev/cuad0 eszközön tanyázik, akkor a következõ sort kell beleírnunk:

```
cuad0:dv=/dev/cuad0:br#19200:pa=none
```

A br tulajdonságnál a modem által ismert legnagyobb adatátviteli sebességet adjuk meg. Ezután gépeljük be a tip cuad0 parancsot és már kapcsolódunk is a modemhez.

Vagy root felhasználóként a cu parancsot is használhatjuk:

```
# cu -lvonal -ssebesség
```

Itt a *vonal* a soros port (például /dev/cuad0) és a *sebesség* annak sebessége (például 57600) lesz. Miután befejeztük az AT parancsok kiadását, az ~. begépelésével tudunk kilépni.

26.5.3. A pn tulajdonságnál a @ jel nem használható!

A pn ("phone number") tulajdonság értékében szereplõ @ jel segítségével az /etc/phones állományban tudunk hivatkozni egy telefonszámra. A @ a tulajdonságokat tároló állományok azonban, így például az /etc/remote állomány esetén is megkülönböztetett jelentéssel bírnak. Ezért itt csak egy visszaper jellel tudjuk beírni:

```
pn=\@
```

26.5.4. Hogyan hívjunk fel egy számot parancssorból?

Tegyünk egy "általános" bejegyzést az /etc/remote állományunkba. Például egy ilyet:

```
tip115200|Dial any phone number at 115200 bps:\
        :dv=/dev/cuad0:br#115200:at=hayes:pa=none:du:
tip57600|Dial any phone number at 57600 bps:\
        :dv=/dev/cuad0:br#57600:at=hayes:pa=none:du:
```

Ezután már ilyet is tudni fogunk:

```
# tip -115200 5551234
```

Ha viszont a `tip` helyett inkább a `cu` programot használnánk szívesen, akkor ehhez készítsünk egy általános bejegyzést:

```
cu115200|Use cu to dial any number at 115200bps:\
        :dv=/dev/cuad1:br#57600:at=hayes:pa=none:du:
```

Majd gépeljük be ezt:

```
# cu 5551234 -s 115200
```

26.5.5. Ehhez minden adandó alkalommal meg kell adnom a sebességet is?

Hozzunk létre egy `tip1200` vagy `cu1200` nevû bejegyzést, de a `br` tulajdonságnál adjuk meg a használni kívánt sebességet. Mivel a `tip` szerint az 1200 bps egy megfelelõ alapértelmezés, ezért alapból a `tip1200` bejegyzést fogja keresni. Ez természetesen nem jelenti azt, hogy ilyen sebsséggel is akarunk dolgozni.

26.5.6. A terminálszerveren keresztül több más gépet is elérek

Ahelyett, hogy minden alkalommal megvárnánk a kapcsolódás befejezést és begépelnénk a `CONNECT gép` parancsot, használjuk a `cm` tulajdonságát. Például nézzük meg ilyen bejegyzést az /etc/remote állományban:

```
pain|pain.deep13.com|Forrester's machine:\
        :cm=CONNECT pain\n:tc=deep13:
muffin|muffin.deep13.com|Frank's machine:\
        :cm=CONNECT muffin\n:tc=deep13:
deep13:Gizmonics Institute terminal server:\
        :dv=/dev/cuad2:br#38400:at=hayes:du:pa=none:pn=5551234:
```

Ennek hatására elég csak annyit megadnunk, hogy `tip pain` vagy `tip muffin`, és már kapcsolódunk is a `pain` vagy `muffin` gépekhez. A `tip deep13` paranccsal pedig egyenesen a terminálszerverhez jutunk el.

26.5.7. Több vonalon is lehet egy géphez csatlakozni?

Ez gyakran okoz gondot olyan esetekben, amikor egy egyetemnek több betárcsázó vonala van, és azokon keresztül többezer hallgató próbál meg dolgozni.

Vegyük fel az egyetemet az /etc/remote állományba és használjuk a `pn` tulajdonság megadásánál a `@` jelet:

```
nagy-egyetem:\
        :pn=\@:tc=dialout
dialout:\
        :dv=/dev/cuad3:br#9600:at=courier:du:pa=none:
```

Ezután adjuk hozzá az /etc/phones állományhoz az egyetem telefonszámait:

```
nagy-egyetem 5551111
nagy-egyetem 5551112
nagy-egyetem 5551113
nagy-egyetem 5551114
```

A `tip` mindegyik telefonszámot az adott sorrendben próbálja tárcsázni és végén feladja a próbálkozást. Ha folyamatosan akarjuk ezeket a számokat hívni, akkor `tip` parancsot tegyük egy ciklusba.

26.5.8. Miért kell kétszer lenyomni a `Ctrl` + `P` gombokat, hogy egyszer elküldje a `Ctrl` + `P` kombinációt?

A `Ctrl` + `P` billentyûkombináció alapértelmezés szerint a "kikényszerítést" jelenti, amivel a `tip` programnak tudunk szólni, hogy a következõ adat szó szerint értendõ. A `~s` szekvenciával bármelyik másik karakternek át tudjuk adni ezt a szerepet, ami egy változó beállítását jelenti ("set a variable").

Gépeljük be, hogy `~sforce=egyetlen-karakter` és zárjuk le egy újsorral. Az *egyetlen-karakter* helyére tetszõleges, egykarakteres szimbólumot megadhatunk. Ha itt nem adunk meg semmit, akkor a kikényszerítõ karakter a `nul` lesz, amit a `Ctrl` + `2` vagy a `Ctrl` + `Szóköz` lenyomásával tudunk elõhozni. Az *egyetlen-karakter* szerepére például tökéletes a `Shift` + `Ctrl` + `6`, amit csak nagyon kevés terminálszerver alkalmaz.

A kikényszerítést végzõ karaktert az $HOME/.tiprc állományban tetszõleges karakterre át tudjuk állítani:

```
force=egyetlen-karakter
```

26.5.9. Miért lett hirtelen minden begépelt betû nagybetûs??

Valószínûleg sikerült lenyomnunk a `Ctrl` + `A` gombkombinációt, ami a `tip` "betûmód váltás" funkciójának felel meg. Ezt olyanok számára dolgozták ki, akiknél nem mûködik a `CapsLock` billentyû. Az elõbb bemutatott `~s` használatával állítsuk át a `raisechar` változót valami másra. Tulajdonképpen akár ugyanarra is állíthatjuk, mint a kikényszerítõ karaktert, ha nem áll szándékunkban használni.

Ebben a példában egy olyan .tiprc állomány szerepel, amely tökéletesen megfelel azon Emacs felhasználók számára, akik sokat használják a `Ctrl` + `2` és `Ctrl` + `A` kombinációkat:

```
force=^^
raisechar=^^
```

A ^^ a Shift + Ctrl + 6 billentyûkombinációt jelenti.

26.5.10. Hogyan mozgassunk állományokat a tip használatával?

Amikor más UNIX® rendszerekkel vesszük fel a kapcsolatot, akkor állományokat a ~p (mint put, vagyis adni) és ~t (mint take, vagyis venni) használatával tudunk mozgatni. Ezek a parancsok a távoli rendszeren a cat és az echo felhasználásával fogadnak és küldenek állományokat. Alakjuk a következõ:

~p helyi-állomány [távoli-állomány]

~t távoli-állomány [helyi-állomány]

Ilyenkor nincs hibaellenõrzés, ezért inkább egy másik protokollt, például zmodemet érdemes használnunk.

26.5.11. Hogyan lehet zmodemet használni a tip programban?

Állományokat úgy tudunk fogadni, ha elõtte a kapcsolat távolabbi végén elindítjuk a küldést végzõ programot. Ezután a ~C rz parancs kiadásával kezdhetjük meg helyben a fogadását.

Állományokat úgy tudunk küldeni, ha elõtte a kapcsolat másik végén elindítjuk a fogadó programot. Ezután a ~C sz állományok parancs kiadásával tudjuk megkezdeni a küldést.

26.6. A soros vonali konzol beállítása

26.6.1. Bevezetés

A FreeBSD képes úgy is elindulni, ha konzolként mindössze egy buta terminált kapcsolunk rá soros porton keresztül. Az ilyen típusú konfigurációs alapvetõen két típus számára bizonyul hasznosnak: azon rendszergazdák számára, akik billentyûzettel és monitorral nem rendelkezõ gépekre akarnak FreeBSD-t telepíteni, és olyan fejlesztõk számára, akik a rendszermag vagy különbözõ eszközmeghajtók mûködését akarják nyomon követni.

Ahogy arról már a A FreeBSD rendszerindítási folyamataben is szó esett, a FreeBSD három indítási fokozattal rendelkezik. Az elsõ két fokozat a rendszerindító blokk kódjában foglal helyet, amely pedig a lemezen található FreeBSD slice elején. A rendszer indulásakor ez a blokk betöltõdik és lefuttatja a harmadik fokozatot képviselõ rendszertöltõt (a /boot/loader állományt).

Ha soros vonali konzol beállításához tehát be kell állítanunk a rendszerindító blokkot, a rendszertöltõt és a rendszermagot.

26.6.2. A soros konzol beállítása, rövidített változat

Ebben a szakaszban azt feltételezzük, hogy az alap beállításokkal dolgozunk és csupán egy gyors

áttekintésre van szükségünk a soros vonali konzolról.

1. Csatlakoztassunk egy soros kábelt a COM1 portra és a terminálra.
2. Rendszeradminisztrátorként a következõ parancs kell kiadnunk ahhoz, hogy a soros konzolon láthassuk az összes rendszerindításhoz tartozó üzenetet:

```
# echo 'console="comconsole"' >> /boot/loader.conf
```

3. Nyissuk meg az /etc/ttys állományt, és a ttyd0 eszközhöz tartozó sorban írjuk át az `off` paramétert az `on` értékre és a `dialup` paramétert a `vt100` értékre. Ha nem ezeket állítjuk be, akkor a soros konzol keresztül jelszó megadása nélkül is be tudunk jelentkezni, ami viszont egy biztonsági rés veszélyével fenyeget.
4. A változtatások érvényesítéséhez indítsuk újra a rendszerünket.

Ha ettõl eltérõ beállításokra lenne szükségünk, akkor a folyamat egyes lépéseibe a A soros vonali konzol beállításaban kaphatunk mélyebb betekintést.

26.6.3. A soros vonali konzol beállítása

1. Készítsük elõ a soros kábelt.

   Vagy a null-modem kábelre vagy pedig egy szabványos soros kábelre és egy null-modem átalakítóra lesz szükségünk. A soros kábelekkel kapcsolatosan a Kábelek és portokt érdemes elolvasni.

2. Húzzuk ki a billentyûzetet.

   A legtöbb személyi számítógép az indítása (vagyis a Power-On Self-Test, POST) során hibát jelez, ha nem érzékel billentyûzetet. Egyes gépek hangosan panaszolják a billentyûzet hiányát, és nem is hajlandóak egészen addig elindulni, amíg nem csatlakoztatunk egyet.

   Ha a számítógépünk hibát küld, de ennek ellenére mégis elindul, akkor semmit nem kell csinálnunk. (Némelyik Phonix BIOS-os gépen ilyenkor megjelenik a `Keyboard failed` hibaüzenet, de ettõl még rendesen elindul a gép.)

   Amennyiben a számítógépünk nem hajlandó billentyûzet nélkül elindulni, állítsuk be a BIOS-ban a "hiba" figyelmen kívül hagyását (már ha ez lehetséges). Az alaplap leírásában találhatjuk meg ennek pontos részleit.

A BIOS paraméterei között a billentyûzetet állítsuk "Not installed" állapotúra. Ilyenkor még továbbra is használható a billentyûzet, ezzel mindössze csak a BIOS számára tiltjuk le az indításkori ellenõrzést, ezért nem fog panaszkodni a hiánya miatt. Tehát a billentyûzetet még a "Not installed" beállítása esetén is nyugodtan csatlakoztatjuk, mert mûködni fog.

Ha a rendszerünkön PS/2®-es egér is található, akkor jó eséllyel a billentyûzettel együtt az egeret is ki tudjuk húzni. Mivel a PS/2®-es egér osztozik a billentyûzettel bizonyos hardvereken, ezért ha nem húzzuk ki az egeret is, akkor az alaplap még továbbra is képes azt gondolni, hogy a billentyûzet ott van. Például az AMI BIOS-os Gateway 2000-as 90 MHz-es Pentium rendszer pontosan így mûködik. Általában véve azonban ez nem szokott gondot okozni, mivel az egér billentyûzet nélkül úgy sem ér túlságosan sokat.

3. Csatlakoztassunk egy buta terminált a COM1 (sio0) portra.

   Ha nem rendelkezünk buta terminállal, akkor erre célra ugyanúgy alkalmas egy régi XT-s PC valamilyen modemprogrammal vagy egy soros porton csatlakozó másik UNIX®-os gép. Ha nincs COM1 (sio0) portunk, akkor szerezzünk egyet. Jelen pillanatban a rendszerindító blokk újrafordítása nélkül a COM1 porton kívül nem tudunk másikat választani. Ha a COM1 portra már raktunk valamilyen másik eszközt, akkor azt ideiglenesen húzzuk le, majd a FreeBSD telepítése és elindítása után tegyünk fel egy másik rendszerindító blokkot. (Egyébként feltételezzük, hogy a COM1 elérhetõ egy állomány/számító/terminálszerveren - ha valóban valamilyen másik célra szükségünk lenne a COM1 portra (és semmiképpen sem tudjuk átrakni a COM2 (sio1) portra), akkor valószínûleg nem is ezzel kellene elsõként foglalkoznunk.)

4. Gondoskodjunk róla, hogy a rendszermag beállításait tartalmazó állományban a COM1 (sio0) eszközhöz megadtuk a megfelelõ paramétereket.

   Ezek az alábbiak:

   `0x10`

   A konzolos mûködési mód engedélyezése az adott egységhez. Ha megadjuk ezt a paramétert, akkor a többit a rendszer figyelmen kívül hagyja. Pillanatnyilag legfeljebb egy egység birtokolhatja ezt a beállítást. Ha több ilyet adtunk volna meg, akkor (a felírás sorrendje szerint) az elsõ kap ilyen szerepet. Ez a beállítás önmagában még nem teszi a soros portot konzollá. Ehhez még szükségünk van a következõ beállításra, vagy a `-h` megadására is.

   `0x20`

   Az egység konzollá nyilvánítása (hacsak nincs egy tõle nagyobb prioritású konzol), függetlenül a lentebb ismertetendõ `-h` opciótól. A `0x20` értéket a `0x10` értékkel együtt kell megadni.

   `0x40`

   (A `0x10` értékkel együtt) az egységet kivonja a normális elérés alól. Ezt a beállítást ne használjuk, ha soros vonali konzolt akarunk üzemeltetni az adott porton. Ezzel az egységet csak a rendszermag távoli nyomkövetéséhez tudjuk használni. A távoli nyomkövetésrõl a fejlesztõk kézikönyvében olvastunk bõvebben.

   Példa:

```
device sio0 at isa? port IO_COM1 flags 0x10 irq 4
```

A további részletekrõl a sio(4) man oldal tud felvilágosítást nyújtani.

Ha nem állítottuk be a megfelelõ paramétereket, akkor (egy másik konzolon) futtassuk a UserConfig programot vagy fordítsuk újra a rendszermagot.

5. Hozzunk létre egy boot.config állományt a rendszer indításához használt meghajtó `a` partíciójának gyökerében.

   Ez az állomány mondja meg a rendszerindító blokkban található kódnak, hogy miként akarjuk indítani a rendszerünket. A soros vonali konzol életrekeltéséhez a most következõ opciók közül kell megadnunk egyet vagy többet - amennyiben többet akarunk megadni, akkor mindegyiket egyetlen sorban szerepeltessük:

   `-h`

   A belsõ és a soros vonali konzolok közti átkapcsolás. Ezzel tudunk a konzolos eszközök között váltani. Például, ha egy belsõ (video) konzolról indítjuk a rendszert, akkor a rendszertöltõnek és a rendszermagnak átadott `-h` paraméterrel arra tudjuk ezeket utasítani, hogy konzolként a soros portot használják. Vagy ha soros porton keresztül indítjuk a rendszert, akkor `-h` megadásával megkérhetjük a rendszertöltõt és a rendszermagot, hogy ezután már a videokártyát használja konzolként.

   `-D`

   Az egy- és kétkonzolos beállítások közti váltás. Az egykonzolos konfigurációban a konzol lehet belsõ (video) vagy soros vonali, attól függõen, hogy miként használtuk a fenti `-h` opciót. A kétkonzolos konfigurációban azonban a videokártyán és a soros vonalon keresztül is egyszerre megjelenik a konzol, függetlenül a `-h` hatásától. Ilyenkor viszont vegyük figyelembe, hogy ez a kétkonzolos konfiguráció csak a rendszerindító blokk futása alatt él. Amint a rendszerindító megkapja a vezérlést, a `-h` által megadott konzol válik az egyedülivé.

   `-P`

   A rendszerindító blokk megpróbálja megkeresni a billentyûzetet. Ha nem találja, akkor magától beállítja a `-D` és `-h` opciókat.

   Tárbeli korlátozások miatt a rendszerindító blokk jelenlegi változata a `-P` paraméterrel csak a kiterjesztett billentyûzeteket képes kezelni. A 101 gombnál kevesebbel (tehát F11 és F12 gombokkal nem) rendelkezõ billentyûzeteket ezért nem feltétlenül fogja észlelni. Ugyanezen korlátozás miatt egyes laptopokon sem minden esetben sikerül érzékelni a billentyûzetet. Ha ez a rendszerünkön problémához vezetne, akkor egyszerûbb lesz elhagyni a `-P` használatát. Sajnos, jelenleg semmilyen megoldás nincs erre.

   Vagy a `-P` opcióval állítassuk be automatikusan a konzolt, vagy pedig a `-h` opcióval engedélyezzük a soros vonali konzolt.

Természetesen itt a boot(8) man oldalon szereplõ összes többi paramétert is megadhatjuk.

A `-P` kivételével az összes opció a rendszertöltõnek (/boot/loader) kerül átadásra. A rendszertöltõ egyedül a `-h` állapotából dönti el, hogy mely belsõ videoeszközön vagy soros porton legyen a konzol. Ez azt jelenti, hogy a /boot.config állományban ha megadjuk a `-D` opciót, de mellette nem szerepel a `-h`, akkor a soros vonali konzolt csak a rendszerindító blokk futása alatt tudjuk elérni - a rendszertöltõ ugyanis alapból a videokártyát használja konzolként.

6. Kapcsoljuk be a számítógépünket.

Amikor elindítjuk a FreeBSD-s gépünket, a rendszerindító blokk kiírja a /boot.config tartalmát a konzolra. Például így:

```
/boot.config: -P
Keyboard: no
```

A második sor csak olyankor jelenik meg, ha a /boot.config állományban a `-P` beállítás is szerepel, és a billentyûzet jelenlétét (yes) vagy hiányát (no) jelzi. A /boot.config tartalmától függõen ezek az üzenetek vagy a soros vonali vagy a belsõ konzolon jelennek meg, esetleg mind a kettõn.

Beállítás	**Ahol megjelenik**
nincs	belsõ konzol
`-h`	soros vonali konzol
`-D`	soros vonali és belsõ konzol
`-Dh`	soros vonali és belsõ konzol
`-P`, van billentyûzet	belsõ konzol
`-P`, nincs billentyûzet	soros vonali konzol

Az iménti üzenetek felbukkanása után a további konzolos üzenetek küldésében egy rövid szünet következik, amíg a rendszerindító blokk a rendszertöltõ betöltésével folytatja a rendszer indítását. Normális körülmények között ezt a folyamatot nem kell megszakítanunk, de esetleg olyankor mégis érdemes lehet, ha le akarjuk ellenõrizni a beállításainkat.

A rendszerindítási folyamat félbeszakításához az `Enter` billentyûn kívül nyomjuk le valamelyik másikat. Ekkor a rendszerindító blokk megáll és várja a további parancsokat. Ekkor valami ilyesmit láthatunk:

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

Nézzük meg, hogy /boot.config beállításainak megfelelõen a fenti üzenet a soros vonali konzolon vagy a belsõ konzolon, illetve mind a kettõn megjelenik-e. Ha az üzenet a megfelelõ konzolon megjelenik, akkor az `Enter` lenyomásával folytathatjuk a rendszer indítását.

Ha nekünk a soros vonali konzolra lenne szükségünk, de semmi nem jelenik meg a soros terminálon, akkor valamit valószínûleg nem jól állítottunk be. A rendszerindító blokktól kapott parancssorban a `-h` begépelésével és az `Enter` vagy `Return` lenyomásával (ha lehetséges) jelezzük neki (és így a rendszertöltõnek és a rendszermagnak is) a soros vonali konzol kiválasztását. Miután befejezõdött a rendszer indítása, menjünk vissza és ellenõrizzük a megfelelõ paramétereket.

Ahogy sikerült elindítani a rendszertöltõt és a rendszerindítás harmadik fokozatába léptünk, a rendszertöltõ megfelelõ környezeti változóin keresztül még mindig van lehetõségünk váltani a soros vonali és a belsõ konzol között, lásd A konzol megváltoztatása a rendszertöltõbõl.

26.6.4. Összefoglalás

Itt most röviden összefoglaljuk az eddig tárgyalt különbözõ beállításokat és ténylegesen kiválasztott konzolt.

1. eset: a sio0 eszköznél a 0x10 beállítást adjuk meg

```
device sio0 at isa? port IO_COM1 flags 0x10 irq 4
```

A /boot.config beállításai	Konzol a rendszerindító blokk alatt	Konzol a rendszertöltõ alatt	Konzol a rendszermagban
nincsenek	belsõ	belsõ	belsõ
`-h`	soros vonali	soros vonali	soros vonali
`-D`	soros vonali és belsõ	belsõ	belsõ
`-Dh`	soros vonali és belsõ	soros vonali	soros vonali
`-P`, van billentyûzet	belsõ	belsõ	belsõ
`-P`, nincs billentyûzet	soros vonali és belsõ	soros vonali	soros vonali

2. eset: a sio0 eszköznél 0x30 beállítása

```
device sio0 at isa? port IO_COM1 flags 0x30 irq 4
```

A /boot.config beállításai	Konzol a rendszerindító blokk alatt	Konzol a rendszertöltõ alatt	Konzol a rendszermagban
nincsenek	belsõ	belsõ	soros vonali
`-h`	soros vonali	soros vonali	soros vonali
`-D`	soros vonali és belsõ	belsõ	soros vonali
`-Dh`	soros vonali és belsõ	soros vonali	soros vonali
`-P`, van billentyûzet	belsõ	belsõ	soros vonali
`-P`, nincs billentyûzet	soros vonali és belsõ	soros vonali	soros vonali

26.6.5. Tanácsok a soros vonali konzol használatához

Nagyobb soros vonali sebesség beállítása

A soros port alapértelmezései a következõk: 9600 baud, 8 bites átvitel, paritás nincs és 1 stopbit. Ha a konzol alapsebességét meg akarjuk változtatni, akkor ahhoz a következõket kell tennünk:

- Fordítsuk újra a rendszerindító blokkokat úgy, hogy a `BOOT_COMCONSOLE_SPEED` változóban a konzolnak egy másik sebességet adunk meg. Az új rendszerindító blokkok fordításáról és telepítésérõl a Soros vonali konzol a sio0 porton kívül máshol ban kapunk részletes leírást.

  Ha a soros vonali konzolt nem a `-h` opcióval állítottuk be, vagy ha a rendszermag a rendszerindító blokkoktól eltérõ módon éri el a soros vonali konzolt, akkor a rendszermag beállításai közé még az alábbit is fel kell vennünk, majd újra kell fordítanunk:

  ```
  options CONSPEED=19200
  ```

- A rendszermagnak adjuk át a `-S` rendszerindítási paramétert. A `-S` parancssori opció a /boot.config állományban is megadható. A boot(8) man oldalon tudhatjuk meg, hogy a /boot.config beállításai közé hogyan tudjuk felvenni és ott milyen további lehetõségeink vannak még.
- A /boot/loader.conf állományban engedélyezzük a `comconsole_speed` beállítást.

  Ez a beállítás a szintén a /boot/loader.conf állományban megadható `console`, `boot_serial` és `boot_multicons` változóktól függ. A soros vonali konzol sebességét tehát például így tudjuk megváltoztatni a `comconsole_speed` megadásával:

  ```
  boot_multicons="YES"
  boot_serial="YES"
  comconsole_speed="115200"
  console="comconsole,vidconsole"
  ```

Soros vonali konzol a sio0 porton kívül máshol

Ha valamilyen okból kifolyólag nem a sio0 porton keresztül akarjuk használni a konzolt, akkor ahhoz a rendszerindító blokkok, a rendszertöltõ és a rendszermag forrásait újra kell fordítanunk az alábbiak szerint:

1. Szerezzük be a rendszermag forrását. (Lásd A FreeBSD frissítése és frissen tartása)
2. Írjuk át a /etc/make.conf állományban a `BOOT_COMCONSOLE_PORT` címét az általunk használt porthoz tartozóéra (0x3F8, 0x2F8, 0x3E8 vagy 0x2E8). Itt csak a sio0 és sio3 (COM1 és COM4) közti portok használhatóak - a többportos soros kártyák címei nem adhatóak meg. A megszakításokat nem kell beállítanunk.
3. Készítsünk egy saját rendszermag beállításait tartalmazó állományt, és vegyük fel bele a használni kívánt soros port megfelelõ paramétereit. Például, ha a sio1 (COM2) eszközt akarjuk konzolként használni:

   ```
   device sio1 at isa? port IO_COM2 flags 0x10 irq 3
   ```

   vagy

   ```
   device sio1 at isa? port IO_COM2 flags 0x30 irq 3
   ```

   A konzolra vonatkozó beállításokat a többi soros portnál ne adjuk meg.
4. Fordítsuk újra és telepítsük a rendszerindító blokkot és a rendszertöltõt:

   ```
   # cd /sys/boot
   # make clean
   # make
   # make install
   ```
5. Fordítsuk és telepítsük újra a rendszermagot.
6. A bsdlabel(8) segítségével másoljuk az új rendszerindító blokkot a rendszer indítását végzõ lemezre és töltsük be az új rendszermagot.

A DDB elérése a soros vonalról

Ha a soros vonali konzolról akarjuk használni a rendszermagba épített nyomkövetõt (ami hasznos lehet távoli vizsgálódáskor, de egyben veszélyes is, ha a soros porton tévesen kiküldünk egy BREAK jelzést!), akkor a rendszermagot a következõ beállításokkal kell fordítanunk:

```
options BREAK_TO_DEBUGGER
options DDB
```

A bejelentkezõ képernyõ elérése a soros vonali konzolról

Habár erre nincs feltétlenül szükségünk, a rendszer üzeneteinek és a rendszermag nyomkövetõjének elérése után akár *be is tudunk jelentkezni* a soros vonalon keresztül. Íme!

Nyissuk meg az /etc/ttys állományt a kedvenc szövegszerkesztõnkkel és keressük meg a következõ sorokat:

```
ttyd0 "/usr/libexec/getty std.9600" unknown off secure
ttyd1 "/usr/libexec/getty std.9600" unknown off secure
ttyd2 "/usr/libexec/getty std.9600" unknown off secure
ttyd3 "/usr/libexec/getty std.9600" unknown off secure
```

A ttyd0 és ttyd3 közti sorok pontosan a COM1 és COM4 közti portoknak felelnek meg. A használni kívánt port sorában szereplõ `off` paramétert írjuk át az `on` értékre. Ha a soros port sebességét is megváltoztattuk, minden bizonnyal a `std.9600` helyett is az adott sebességhez illeszkedõ paramétert kell megadnunk, például az `std.19200` értékkel.

Érdemes továbbá még az `unknown` helyett megadni az adott terminál típusát.

Az állomány módosítását követõen a változatások érvényesítéséhez ki kell adnunk a `kill -HUP 1` parancsot is.

26.6.6. A konzol megváltoztatása a rendszertöltõbõl

A korábbi szakaszokban arról beszéltünk, hogy miként állítsuk be a soros vonali konzolt a rendszerindító blokk megpiszkálásával. Ebben a szakaszban viszont azt mutatjuk meg, hogy különbözõ parancsokon és környezeti változókon keresztül miként tudjuk megadni a konzolt a rendszertöltõben. Mivel a rendszertöltõre a rendszerindítás harmadik fokozatában kerül sor, az ott megadott értékekkel felül tudjuk bírálni a rendszerindító blokk beállításait.

A soros vonali konzol beállítása

A rendszertöltõ és a rendszermag az /boot/loader.conf állományon keresztül elég könnyen rávehetõ a soros vonali konzol használatára:

```
set console="comconsole"
```

Ez a rendszerindító blokk elõzõ szakaszban tárgyalt beállításaitól függetlenül érvényesül.

A fenti sort a /boot/loader.conf állomány elejére érdemes tennünk, így a soros vonali konzolon már a lehetõ leghamarabb megjelennek a rendszer üzenetei.

Ehhez hasonló módon a belsõ konzolt is megadhatjuk:

```
set console="vidconsole"
```

Ha a rendszertöltõben nem adjuk meg a `console` környezeti változó értékét, akkor a rendszertöltõ, és így a rendszermag is, a rendszerindító blokkban a `-h` opció által meghatározott konzolt fogja használni.

A konzol a /boot/loader.conf.local vagy a /boot/loader.conf állományokban adható meg.

A részletekkel kapcsolatban lásd a loader.conf(5) man oldalt.

Jelen pillanatban a rendszertöltõnek nincs a `-P` paraméterrel ekvivalens értékû beállítása, ezért a billentyûzet jelenléte alapján nem képes magától választani a belsõ és a soros vonali konzol között.

Soros vonali konzol a sio0 porton kívül máshol

A rendszertöltõt ne a sio0 eszközzel fordítsuk újra a soros vonali konzolhoz. Ehhez kövessük a Soros vonali konzol a sio0 porton kívül máshol ban leírt eljárás lépéseit.

26.6.7. Figyelmeztetések

A szakaszban szereplõ ötletek alapján sokan így most már könnyen be tudnak állítani egy billentyûzet és grafikus hardver nélküli dedikált szervert. Sajnos azonban a legtöbb rendszer nem engedi a billentyûzet nélküli indítást, és akad néhány olyan is, amely pedig a grafikus kártya hiányában nem is indul el. Az AMI BIOS-os gépeknél a grafikus kártya nélküli indításhoz elegendõ csupán a beállítások között a grafikus kártyát ("graphics adapter") "Not installed" (nem telepített) állapotúra állítani. Ha ilyen opció nem található a BIOS-ban, akkor helyette keressük a "Halt on Error" (leállás hiba esetén) változatot. Ha ezt a "All but Keyboard" (a billentyûzet kivételével minden) vagy akár a "No Error" (soha) értékre állítjuk, az elõbbi eredményt kapjuk.

Ennek ellenére elõfordulhat azonban, hogy egyes gépeken egyáltalán nem találunk ilyen lehetõséget és videokártya nélkül nem indulnak el. Ezekben az esetekben tegyünk a gépbe valamilyen kártyát (ehhez elég egy egyszerû típus is), de monitort már ne kössünk rá. Esetleg megpróbálkozhatunk még AMI BIOS telepítésével is.

Chapter 27. A PPP és a SLIP

27.1. Áttekintés

A FreeBSD számos módon képes összekötni két számítógépet. Ha betárcsázós modemmel akarunk hálózati vagy internetes kapcsolatot felépíteni, esetleg azt szeretnénk, hogy mások képesek legyenek minket ilyen módon elérni, akkor ahhoz PPP-t, illetve SLIP-et kell használnunk. Ebben a fejezetben a modemes kommunikáció beállításait mutatjuk be részletesebben.

A fejezet elolvasása során megismerjük:

- hogyan állítsunk be felhasználói PPP-t;
- hogyan állítsunk be rendszerszintû PPP-t (csak FreeBSD 7.*X*);
- hogyan állítsunk be egy PPPoE (PPP over Ethernet, vagyis "PPP Ethernet felett") kapcsolatot;
- hogyan állítsunk be egy PPPoA (PPP over ATM, vagyis "PPP ATM felett") kapcsolatot;
- hogyan állítsunk be SLIP klienst és szervert (csak FreeBSD 7.*X*).

A fejezet elolvasásához ajánlott:

- az alapvetõ hálózati technológiák ismerete;
- a betárcsázós kapcsolatok, a PPP és/vagy SLIP alapjainak és céljainak megértése.

Talán érdekli a kedves olvasót, hogy mi az alapvetõ különbség a felhasználói és a rendszerszintû PPP között. A válasz egyszerû: a felhasználói PPP a beérkezõ és kimenõ adatokat nem a rendszermagban, hanem a felhasználói szinten dolgozza fel. Ez költséges abból a szempontból, hogy emiatt adatokat kell másolgatni a rendszer és a felhasználói szint között, azonban egy sokkal többet tudó PPP implementációnak ad ezzel utat. A felhasználói PPP a tun eszközön keresztül kommunikál a külvilággal, miközben a rendszermagban található PPP mindezt a ppp eszközzel valósítja meg.

A fejezetben a felhasználói PPP-t egyszerûen csak ppp néven fogjuk hivatkozni, hacsak nem lesz szükséges különbséget tennünk közte és más PPP szoftverek, mint például a pppd között. Ha mást nem mondunk, akkor a fejezetben ismertetett összes parancsot `root` felhasználóként kell kiadni.

27.2. A felhasználói PPP alkalmazása

A FreeBSD 8.0 változatától kezdõdõen a soros portokhoz tartozó eszközök nevei /dev/cuadN helyett /dev/cuauN, illetve /dev/ttydN helyett /dev/ttyuN lettek. A FreeBSD 7.*X* felhasználóknak ezeknek a változásoknak megfelelõen kell olvasniuk az itt szereplõ dokumentációt.

27.2.1. A felhasználói PPP

Elõfeltételek

A leírás feltételezi, hogy rendelkezünk a következõkkel:

- Olyan internet-elõfizetés, ahol PPP-n keresztül csatlakozunk
- Egy modem vagy más olyan rendszerünkhöz csatlakozó eszköz, amelyen keresztül el tudjuk érni az internet-szolgáltatónkat
- Az internet-elõfizetés betárcsázásához szükséges telefonszámok
- A bejelentkezési nevünk és jelszavunk. (Vagy a megszokott UNIX®-os felhasználói név és jelszó páros, vagy egy PAP esetleg CHAP bejelentkezési név és jelszó.)
- Egy vagy több névszerver IP-címe. Ehhez az internet-szolgáltatók általában két IP-címet adnak meg. Ha egyet sem kaptunk, akkor a ppp.conf állományban erre a célra használhatjuk az `enable dns` parancsot, és ekkor a ppp majd automatikusan be fogja állítani nekünk a névszervereket. Ezt a lehetõséget az befolyásolja, hogy az internet-szolgáltató oldalán mûködõ PPP implementáció támogatja-e a névfeloldás egyeztetését (DNS negotiation).

A következõ információkat is megkaphatjuk az internet-elõfizetésünkhöz, de nem feltétlenül szükségesek:

- Az internet-szolgáltató átjárójának IP-címe. Az átjáró az a gép, amelyen keresztül a gépünk csatlakozik és számára ez lesz az *alapértelmezett átjáró*. Ha nem rendelkezünk ezzel az információval, akkor csak állítsunk be valamit, és majd a csatlakozáskor a szolgáltató PPP szervere felülírja a megfelelõ beállításokkal.

  Erre a címre a ppp `HISADDR` néven hivatkozik.

- A használandó hálózati maszk. Amennyiben a szolgáltató ezt nem adta meg, nyugodtan használjuk erre a `255.255.255.255` értéket.
- Ha a szolgáltatónk statikus IP-címet és rögzített hálózati nevet is biztosít nekünk, ezt is megadhatjuk. Minden más esetben egyszerûen csak hagyjuk, hogy a rendszer automatikusan válasszon nekünk egyet.

Ha a szükséges információknak nem vagyunk birtokában, akkor vegyük fel a kapcsolatot az internet-szolgáltatókkal.

Ebben a szakaszban a példákban szereplõ konfigurációs állományok sorait számozva láthatjuk. Ezek a sorszámok a bemutatás és a tárgyalás megkönnyítése érdekében szerepelnek, és nem az eredeti állományok részei. Mindezek mellett a tabulátorok és szóközök megfelelõ használata is fontos.

A PPP automatikus beállítása

A `ppp` és a `pppd` (a PPP rendszerszintû megvalósítása) egyaránt az /etc/ppp könyvtárban található konfigurációs állományokat használja. A felhasználói PPP-hez ezenkívül még a /usr/shared/examples/ppp/ könyvtárban vannak példák.

A `ppp` parancs beállítása az igényeinktõl függõen számos állomány módosítását igényelheti. A tartalmukat nagyban befolyásolja, hogy a szolgáltatónk részérõl a címeket kiosztása statikus (vagyis

egy adott címet kapunk és folyamatosan azt használjuk) esetleg dinamikus (vagyis az IP-címünk minden egyes kapcsolódáskor más és más).

PPP statikus IP-címmel

Ebben az esetben az /etc/ppp/ppp.conf konfigurációs állományt kell átszerkesztenünk. Tartalma az alábbi példához hasonlítható.

A : karakterrel végzõdõ sorok mindig az elsõ oszlopban kezdõdnek (tehát a sor elején), míg az összes többi sort tabulátorok vagy szóközök használatával bentebb kell raknunk.

```
1     default:
2       set log Phase Chat LCP IPCP CCP tun command
3       ident user-ppp VERSION (built COMPILATIONDATE)
4       set device /dev/cuau0
5       set speed 115200
6       set dial "ABORT BUSY ABORT NO\\sCARRIER TIMEOUT 5 \
7                 \"\" AT OK-AT-OK ATE1Q0 OK \\dATDT\\T TIMEOUT 40 CONNECT"
8       set timeout 180
9       enable dns
10
11    szolgaltato:
12      set phone "(123) 456 7890"
13      set authname ize
14      set authkey mize
15      set login "TIMEOUT 10 \"\" \"\" gin:--gin: \\U word: \\P col: ppp"
16      set timeout 300
17      set ifaddr x.x.x.x y.y.y.y 255.255.255.255 0.0.0.0
18      add default HISADDR
```

1- sor

Ez azonosítja be az alapértelmezett bejegyzést. Az itt szereplõ parancsok a `ppp` minden egyes futásakor magukból végrehajtódnak.

2- sor

Beállítja a naplózás paramétereit. Amikor a beállításaink már kifogástalanul mûködnek, akkor ezt a sort érdemes átírni a következõre:

```
set log phase tun
```

Ezzel jelentõs mértékben vissza tudjuk fogni a naplózás mértékét.

3- sor

Ezzel mondjuk meg a PPP-nek, hogy a többiek felé miként azonosítsa magát. A PPP akkor azonosítja magát a társak felé, ha valamilyen gondja akad az egyeztetésekkel és a kapcsolat beállításával. Az így továbbított információk a másik oldal rendszergazdái számára nyújthatnak

segítséget az ilyen jellegû problémák felderítésében.

4- sor

Itt adjuk meg az eszközt, amelyre a modem csatlakozik. A COM1 neve /dev/cuau0, a COM2 neve pedig /dev/cuau1.

5- sor

A csatlakozás sebességét adjuk meg. Ha a 115 200-as érték itt nem mûködne (ez egyébként minden újabb gyártmányú modem esetében elfogadható), akkor helyette használjuk a 38400-as beállítást.

6- és 7- sorok

A híváshoz használt karakterlánc. A felhasználói PPP a chat(8) programhoz hasonló "küldök-várok" típusú szerkesztést alkalmaz. A kihasználható lehetõségekrõl a man oldalán olvashatunk részletesebben.

Az olvashatóság kedvéért a parancs a következõ sorban folytatódik. A ppp.conf állományban bármelyik parancs, ahol a \ karakterrel zárjuk a sort, az ugyanígy folytatható a következõben.

8- sor

A kapcsolathoz tartozó üresjárati idõt állítja be. Ennek értéke alapból 180 másodperc, így ez a sor pusztán csak az érthetõséget szolgálja.

9- sor

Arra utasítja a PPP-t, hogy a többiektõl kérdezze le a helyi névfeloldó beállításait. Ha saját névszervert futtatunk, akkor ezt a sort tegyük inkább megjegyzésbe vagy töröljük ki.

10- sor

Ez az üres sor az átláthatóság kedvéért került bele. A PPP az összes üres sort figyelmen kívül hagyja.

11- sor

Itt kezdõdik a "szolgaltato" nevû szolgáltatóhoz tartozó bejegyzés. Ezt késõbb akár ki is cserélhetjük az internet-szolgáltatónk nevére, így a `load szolgáltató` beállítással tudjuk majd beindítani a kapcsolatot.

12- sor

Beállítjuk a szolgáltatóhoz tartozó telefonszámot. A kettõspont (:) vagy a csõvezeték (|) karakterekkel elválasztva több telefonszámot is meg tudunk adni. A ppp(8) oldalon olvashatunk a két elválasztó közti különbségekrõl. Röviden ezeket úgy foglalhatnánk össze, hogy ha váltogatni akarunk a számok között, akkor használjuk a kettõspontot. Ha mindig az elsõként megadott számot akarjuk hívni és a többit csak akkor, ha ez nem mûködik, akkor a csõvezeték karakterre lesz szükségünk. Ahogy a példa is mutatja, az összes telefonszámot tegyük mindig idézõjelek közé.

Ha a telefonszámban egyébként is szerepelnek szóközök, akkor is idézõjelek (") közé kell tennünk. Ennek elhagyásával egy egyszerû, ámde kényes hibát ejtünk.

13- és 14- sor

A felhasználói nevet és jelszót tartalmazza. Amikor egy UNIX® fajtájú bejelentkezést kapunk, akkor ezekre az értékekre a `set login` parancsban \U és \P változókkal tudunk hivatkozni. Ha PAP vagy CHAP használatával jelentkezünk be, akkor ezek az értékek a hitelesítéskor kerülnek felhasználásra.

15- sor

Ha a PAP vagy CHAP protokollok valamelyikét használjuk, akkor nem lesz szükségünk a login változóra, ezért ezt megjegyzésbe is tehetjük, vagy akár ki is törölhetjük. A PAP és CHAP hitelesítésrõl szóló részben olvashatjuk ennek további részleteit.

A bejelentkezéshez használt karakterlánc hasonlít a behíváshoz használt, chat-szerû felépítéssel rendelkezõ karakterlánchoz. A példában látható karakterlánc egy olyan szolgáltatáshoz illeszkedik, ahol a bejelentkezés valahogy így néz ki:

```
A Világ Legjobb Szolgáltatója
login: izé
password: mizé
protocol: ppp
```

Ezt a szkriptet alakítsuk a saját igényeinkhez. Ha elõször próbálkozunk ilyen szkript írásával, akkor lehetõleg kapcsoljuk be a rendszerek között lezajló "beszélgetés" naplózását, hogy ellenõrizni tudjuk minden a megfelelõen módon történik-e.

16- sor

Beállítjuk a kapcsolathoz tartozó alapértelmezett idõkorlátot (másodpercben). Itt a kapcsolat automatikusan lezárul 300 másodperc tétlenséget követõen. Ha nem akarunk ilyen korlátot szabni, akkor ezt az értéket állítsuk nullára vagy használjuk a `-ddial` paranccsori kapcsolót.

17- sor

A felülethez tartozó címeket állítja be. A *x.x.x.x* helyére a szolgáltató által kiosztott IP-címet kell beírnunk. A *y.y.y.y* helyett pedig a szolgáltató átjárója kerül be (lényegében az a gép, amelyhez csatlakozunk). Amennyiben az internet-szolgáltatónk nem adott meg semmilyen átjárót, erre a célra a `10.0.0.2/0` címet is használhatjuk. Amikor "nekünk kell kitalálnunk" ezeket a címeket, akkor ne felejtsünk el létrehozni hozzájuk egy bejegyzést az /etc/ppp/ppp.linkup állományban a PPP dinamikus IP-címmel szakaszban szereplõek szerint. Ha nem adjuk meg ezt a sort, akkor a `ppp` parancs nem képes `-auto` módban mûködni.

18- sor

A szolgáltató átjárójához felvesz egy alapértelmezett útvonalat. A `HISADDR` kulcsszót a 17. sorban megadott átjáró címével helyettesítjük. Ezért fontos, hogy ez a 17. sor után szerepeljen, különben a `HISADDR` nem lesz képes inicializálódni.

Ha a `ppp` parancsot nem akarjuk `-auto` módban futtatni, akkor ezt a sort a ppp.linkup állományba is átrakhatjuk.

Ha statikus IP-címmel rendelkezünk és a `ppp -auto` módban fut, akkor a ppp.linkup állományba egészen addig nem kell semmit sem írnunk, amíg a csatlakozás elõtt az útválasztási táblázatokban a

megfelelõ adatok találhatóak. Olyankor is jól jöhet, amikor a csatlakozást követõen meg akarunk hívni bizonyos programokat. Ezt majd a sendmailes példában fogjuk bõvebben kifejteni.

Erre példákat a /usr/shared/examples/ppp/ könyvtárban találhatunk.

PPP dinamikus IP-címmel

Ha az internet-szolgáltatónktól nem kaptunk statikus IP-címet, akkor a `ppp` paranccsal is be tudjuk állítani a helyi és távoli címeket. Ez az IP-címek "kitalálásával" történik, valamint úgy, hogy a `ppp` számára a csatlakozás után lehetõvé tesszük az IP konfigurációs protocol (IP Configuration Protocol, IPCP) használatát. A ppp.conf tartalma szinte teljesen megegyezik a PPP statikus IP-címmel részben szereplõvel, egyetlen apró különbséggel:

```
17      set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.255
```

Ismét szeretnénk elmondani, hogy a sorszámot ne írjuk bele, hiszen az csak hivatkozási céllal szerepel. Legalább egy szóközzel kezdjünk bentebb.

17- sor

A `/` után megjelenõ szám azoknak a biteknek a számát adja meg, amire a ppp támaszkodik. A környezetünknek jobban megfelelõ IP-címeket is megadhatunk, de a fenti példa minden esetben mûködni fog.

Az utolsó paraméterrel (`0.0.0.0`) azt mondjuk a PPP-nek, hogy az egyeztetést ne a `10.0.0.1`, hanem a `0.0.0.0` címmel kezdje meg, amire egyes szolgáltatók esetén szükségünk is lesz. A `set ifaddr` elsõ paramétereként azonban soha ne adjuk meg a `0.0.0.0` címet, mivel ezzel a PPP `-auto` módban nem tudja beállítani a kezdeti útvonalat.

Ha nem `-auto` módban indítjuk, akkor az /etc/ppp/ppp.linkup állományban meg kell adnunk még egy bejegyzést is. A ppp.linkup állományt a kapcsolat létrejötte után dolgozzuk fel. Itt már a `ppp` megkapta a felülethez tartozó címeket, így az útválasztási táblázatba fel tudjuk venni hozzájuk a megfelelõ bejegyzéseket:

```
1     szolgaltato:
2      add default HISADDR
```

1- sor

A kapcsolat felépítése során a `ppp` a ppp.linkup állományban a következõ szabályok szerint fogja keresni a bejegyzéseket: elõször a ppp.conf állományban megadott címkét próbálja megtalálni. Ha ez nem sikerül, akkor az átjárónknak megfelelõ bejegyzést kezdi el keresni. Ez egy négy byte-ból álló, felírásában az IP-címekhez hasonlító címke. Ha még ez a címke sem található, akkor a `MYADDR` bejegyzést keresi.

2- sor

Ez a sor mondja meg a `ppp` programnak, hogy vegyen fel egy `HISADDR` címre vonatkozó alapértelmezett útvonalat. A `HISADDR` címet az IPCP által egyeztetett átjáró IP-címére cseréljük ki.

Ha erre a részletesebb példát akarunk látni, akkor a /usr/shared/examples/ppp/ppp.conf.sample és /usr/shared/examples/ppp/ppp.linkup.sample állományokban a `pmdemand` bejegyzést nézzük meg.

A bejövõ hívások fogadása

Amikor egy helyi hálózathoz csatlakozó gépen akarjuk a ppp programot beállítani a bejövõ hívások fogadására, akkor azt is el kell döntenünk, hogy engedélyezzük-e a csomagok továbbküldését a belsõ hálózat felé. Amennyiben igen, akkor a becsatlakozó gépenek a belsõ hálózatunkon ki kell osztani egy külön címet és az /etc/ppp/ppp.conf állományban, és meg kell adnunk az `enable proxy` parancsot. Emellett még az /etc/rc.conf állományban se feleljtsük el megadni a következõ sort:

```
gateway_enable="YES"
```

Melyik getty?

A FreeBSD beállítása betárcsázós kapcsolatokhoz nagyon jól bemutatja a betárcsázós szolgáltatások beállítását a getty(8) segítségével.

A `getty` helyett egyébként az mgetty, a `getty` egy ügyesebb változata is használható (a comms/mgetty+sendfax portból), amely kifejezetten a betárcsázós vonalakhoz készült.

A `mgetty` használatának többek közt az egyik elõnye, hogy *aktívan tartja a kapcsolatot* a modemekkel, tehát hogy ha az /etc/ttys állományban letiltjuk a modemet, akkor nem is fog válaszolni a hívásokra.

Emellett az `mgetty` késõbbi változatai (a 0.99 beta változatától kezdve) még a PPP folyamok automatikus észlelését is támogatják, ezáltal a kliensek szkriptek nélkül is képesek elérni a szerverünket.

Ha errõl többet akarunk megtudni, akkor az `mgetty` paranccsal kapcsolatban olvassuk el Az mgetty és az AutoPPP címû szakaszt.

A PPP engedélyei

A `ppp` parancsot általában `root` felhasználóként kell futtatni. Ha viszont a `ppp` parancsot tetszõleges felhasználóval akarjuk szerver módban futtatni az iméntiek szerint, akkor ahhoz fel kell vennünk az /etc/group állományban szereplõ `network` csoportba.

Ezeken kívül még az `allow` paranccsal is engedélyeznünk kell konfigurációs állomány egy vagy több részének elérését is:

```
allow users fred mary
```

Ha ezt a parancsot a `default` bejegyzésnél adjuk meg, akkor az így megadott felhasználók mindenhez hozzá tudnak férni.

PPP shellek a dinamikus IP-címek használóinak

Hozzunk létre egy /etc/ppp/ppp-shell nevû állományt, amelyben a következõk szerepelnek:

```
#!/bin/sh
IDENT=`echo $0 | sed -e 's/^.*-\(.*\)$/\1/'`
CALLEDAS="$IDENT"
TTY=`tty`

if [ x$IDENT = xdialup ]; then
        IDENT=`basename $TTY`
fi

echo "PPP for $CALLEDAS on $TTY"
echo "Starting PPP for $IDENT"

exec /usr/sbin/ppp -direct $IDENT
```

Ez a szkript legyen végrehajtható. Ezután az alábbi paranccsal ppp-dialup néven készítsünk egy szimbolikus linket erre a szkriptre:

```
# ln -s ppp-shell /etc/ppp/ppp-dialup
```

Ez a szkript lesz az összes betárcsázó felhasználónk *shellje*. A most következõ példa az /etc/passwd állományban szereplõ, `pchilds` nevû PPP felhasználó bejegyzését mutatja be (ne felejtsük el, hogy soha ne közvetlenül szerkesszük a jelszavakat tároló állományt, hanem a vipw(8) segítségével).

```
pchilds:*:1011:300:Peter Childs PPP:/home/ppp:/etc/ppp/ppp-dialup
```

Hozzunk létre egy /home/ppp nevû könyvtárat a következõ bárki által olvasható 0 byte-os állományokkal:

```
-r--r--r--   1 root     wheel           0 May 27 02:23 .hushlogin
-r--r--r--   1 root     wheel           0 May 27 02:22 .rhosts
```

Ezek hatására az /etc/motd állomány tartalma nem jelenik meg.

PPP shellek a statikus IP-címek használóinak

Az iméntiekhez hasonló módon készítsük el a ppp-shell állományt, és mindegyik statikus IP-vel rendelkezõ hozzáféréshez csináljunk egy szimbolikus linket a ppp-shell szkriptre.

Például, ha három betárcsázós ügyfelünk van, `fred`, `sam` és `mary`, feléjük 24 bites CIDR hálózatokat közvetítünk, akkor a következõket kell begépelnünk:

```
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-fred
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-sam
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-mary
```

A fentebb szereplõ betárcsázós felhasználók eléréseihez tartozó shelleket állítsuk be az itt létrehozott szimbolikus linkekre (így tehát `mary` shellje az /etc/ppp/ppp-mary lesz).

A ppp.conf beállítása a dinamikus IP-címek használóinak

Az /etc/ppp/ppp.conf állományban a következõ sorok valamelyikének kellene szerepelnie:

```
default:
  set debug phase lcp chat
  set timeout 0

ttyu0:
  set ifaddr 203.14.100.1 203.14.100.20 255.255.255.255
  enable proxy

ttyu1:
  set ifaddr 203.14.100.1 203.14.100.21 255.255.255.255
  enable proxy
```

A bentebb kezdett sorokat mi is kezdjünk bentebb.

A `default:` szakasz minden kapcsolat esetén betöltõdik. Az /etc/ttys állományban engedélyezett mindegyik betárcsázós vonal létrehoz a fenti `ttyu0:` szakaszhoz hasonló bejegyzést. Minden vonal kap egy egyedi IP-címet a dinamikus felhasználók számára szánt címtartományból.

A ppp.conf beállítása a statikus IP-vel rendelkezõk számára

A /usr/shared/examples/ppp/ppp.conf állományban szereplõ tartalom mellett az összes statikus kiosztású IP-címmel rendelkezõ betárcsázó felhasználóhoz még hozzá kell tennünk egy szakaszt. A példánkban ezek továbbra is `fred`, `sam` és `mary`.

```
fred:
  set ifaddr 203.14.100.1 203.14.101.1 255.255.255.255

sam:
  set ifaddr 203.14.100.1 203.14.102.1 255.255.255.255

mary:
  set ifaddr 203.14.100.1 203.14.103.1 255.255.255.255
```

Amennyiben szükséges, az /etc/ppp/ppp.linkup tartalmazhat további útválasztási információkat is az egyes statikus IP-címmel rendelkezõ felhasználókhoz. A lentebb bemutatott sor a kliens ppp összekötettésén keresztül vesz fel egy útvonalat a `203.14.101.0/24` hálózat felé.

```
fred:
  add 203.14.101.0 netmask 255.255.255.0 HISADDR

sam:
```

```
  add 203.14.102.0 netmask 255.255.255.0 HISADDR

mary:
  add 203.14.103.0 netmask 255.255.255.0 HISADDR
```

Az `mgetty` és az AutoPPP

Az comms/mgetty+sendfax port alapértelmezés szerint az `AUTO_PPP` beállítással érkezik, amely lehetõvé teszi, hogy az `mgetty` képessé legyen a PPP kapcsolatok LCP fázisát észlelni és magától létrehozni hozzá egy ppp shellt. Mivel az alapértelmezett név/jelszó páros azonban ilyenkor nem jelenik meg, a felhasználókat a PAP vagy a CHAP protokollon keresztül lehet hitelesíteni.

Ez a szakasz most feltételezi, hogy a sikeresen beállítottuk, lefordítottuk és telepítettük az comms/mgetty+sendfax portot.

Az /usr/local/etc/mgetty+sendfax/login.config állományban ne felejtsük ellenõrizni, hogy szerepel a következõ:

```
/AutoPPP/ -     -            /etc/ppp/ppp-pap-dialup
```

Ezzel utasítjuk az `mgetty` programot arra, hogy az észlelt PPP kapcsolatokhoz futtassa le a ppp-pap-dialup szkriptet.

Hozzunk létre az /etc/ppp/ppp-pap-dialup nevû állományt, amelyben majd a következõk fognak szerepelni (az állomány legyen végrehajtható):

```
#!/bin/sh
exec /usr/sbin/ppp -direct pap$IDENT
```

Az /etc/ttys állományban engedélyezett összes betárcsázós vonalhoz készítsük el a megfelelõ bejegyzést az /etc/ppp/ppp.conf állományban. Ezek remekül meg fognak férni az imént készített definíciókkal.

```
pap:
  enable pap
  set ifaddr 203.14.100.1 203.14.100.20-203.14.100.40
  enable proxy
```

Minden olyan felhasználónak, aki ezzel a módszerrel jelentkezik be, szüksége lesz egy név/jelszó kombinációra az /etc/ppp/ppp.secret állományban, vagy az alábbi beállítás megadásával választhatjuk azt is, hogy a felhasználókat az /etc/passwd állományon keresztül a PAP protokoll segítségével azonosítjuk.

```
enable passwdauth
```

Ha statikus IP-címet akarunk kiosztani némely felhasználóknak, akkor az /etc/ppp/ppp.secret állományban ezt megadhatjuk a harmadik paraméternek. Erről bővebben a /usr/shared/examples/ppp/ppp.secret.sample állományban láthatunk példát.

A Microsoft kiterjesztései

A PPP úgy is beállítható, hogy kérésre DNS és NetBIOS típusú névfeloldáshoz is szolgáltasson információkat.

A PPP 1.x változatával úgy lehet engedélyezni ezeket a kiterjesztéseket, ha az /etc/ppp/ppp.conf állomány megfelelő részeibe felvesszük a következő sorokat:

```
enable msext
set ns 203.14.100.1 203.14.100.2
set nbns 203.14.100.5
```

A PPP második és későbbi változataiban pedig:

```
accept dns
set dns 203.14.100.1 203.14.100.2
set nbns 203.14.100.5
```

Ezzel a kliens megkapja az elsődleges és másodlagos névszerverek címeit, valamint a NetBIOS névszervert.

Ha a második és az azt követő verziókban a `set dns` sort elhagyjuk, akkor a PPP az /etc/resolv.conf állományban található értékeket fogja használni.

A PAP és CHAP hitelesítés

Egyes internet-szolgáltatók úgy állítják be a rendszerüket, hogy a kapcsolat felépítése során a hitelesítés a PAP vagy CHAP mechanizmusok valamelyikével történik. Ilyenkor a szolgáltató nem egy `login:` sorral fogja bekérni a szükséges adatokat, hanem közvetlenül a PPP kapcsolatot kezdi el használni.

A PAP nem olyan biztonságos, mint a CHAP, de itt a biztonság nem is annyira fontos, mivel a jelszavak, amelyeket ugyan a PAP titkosítatlan formában küld tovább, csak egy soros vonalon haladnak át. A rossz indulatú támadók itt nem sok mindent tudnak "lehallgatni".

A PPP statikus IP-címmel és a PPP dinamikus IP címmel című szakaszokhoz képest a következő módosításokat kell elvégeznünk:

```
13      set authname AFelhasználóiNevem
14      set authkey AJelszavam
15      set login
```

13- sor

Ebben a sorban adjuk meg a PAP/CHAP felhasználói nevünket, amelyet *AFelhasználóiNevem* helyett kell beírni.

14- sor

Ebben a sorban adjuk meg a PAP/CHAP jelszavunkat, *AJelszavam* helyett. Szándékunk egyértelmûsítése érdekében ezek mellett még egy további sort is érdemes felvennünk, tehát:

```
16      accept PAP
```

vagy

```
16      accept CHAP
```

Alapértelmezés szerint a PAP és CHAP is egyaránt elfogadott.

15- sor

A PAP és CHAP alkalmazásakor általában nem is kell bejelentkeznünk a szolgáltató szerverére. Ezért a "set login" parancsnál használt karakterláncot le is kell tiltanunk.

A `ppp` beállításainak megváltoztatása menet közben

A háttérben futó `ppp` programhoz menet közben is tudunk beszélni, de csak olyankor, amikor az ehhez szükséges portot megadtuk. Ezt úgy tudjuk megtenni, ha beállítások közé felvesszük az alábbit:

```
set server /var/run/ppp-tun%d DiagnosticPassword 0177
```

Így a PPP az elõre megadott UNIX® tartománybeli socketen keresztül fogja várni a kapcsolódásunkat, és a konkrét hozzáféréshez jelszót kér. A névben szereplõ `%d` a használatban levõ tun eszköz sorszámát jelöli.

Miután a csatlakozás beállítódott, a szkriptekben a pppctl(8) program használható a futó program vezérléséhez.

A PPP hálózati címfordítási képességének kihasználása

A PPP képes a rendszermag rásegítése nélkül képes hálózati címfordítást végezni. Ezt a lehetõséget a következõ sor hozzáadásával tudjuk aktiválni az /etc/ppp/ppp.conf állományban:

```
nat enable yes
```

A PPP-be épített hálózati címfordítás a `-nat` parancssori paraméterrel is bekapcsolható. Az /etc/rc.conf állományban is található hozzá egy `ppp_nat` változó, amely alapértelmezés szerint engedélyezett.

Amikor használjuk ezt a lehetõséget, az /etc/ppp/ppp.conf állományban a következõ opciókkal engedélyezhetjük a bejövõ kapcsolatok továbbítását:

```
nat port tcp 10.0.0.2:ftp ftp
nat port tcp 10.0.0.2:http http
```

vagy egyáltalán ne bízzunk meg a külvilágban:

```
nat deny_incoming yes
```

A rendszer végsõ beállítása

Mostanra ugyan már beállítottuk a `ppp` programot, azonban még néhány dolgot be kell állítanunk, mielõtt ténylegesen nekilátnánk használni. Ezek mindegyike az /etc/rc.conf állomány módosítását igényli.

Az állományt fentrõl lefelé fogjuk feldolgozni, de elõtte ne felejtsünk el értéket adni a `hostname=` változónak, például:

```
hostname="ize.minta.com"
```

Amennyiben a szolgáltatónk statikus IP-címet és nevet biztosít számunkra, az lesz a legjobb, ha itt a tõle kapott nevet adjuk meg.

Keressük meg a `network_interfaces` változót. Ha a rendszerünkben kérésre akarjuk tárcsázni a szolgáltatónkat, akkor a tun0 eszközt mindenképpen vegyük fel az értékébe, minden más esetben pedig távolítsuk el.

```
network_interfaces="lo0 tun0"
ifconfig_tun0=
```

Az `ifconfig_tun0` változónak üres értéket kell megadnunk, és létre kell hoznunk egy /etc/start_if.tun0 nevû állományt. Ebben a következõ sornak kell szerepelnie:

```
ppp -auto arendszerem
```

Ez a szkript a hálózat beállításakor fut le, és a ppp démont automatikus módban indítja el. Ha az adott gép egy helyi hálózat átjárója is egyben, akkor az `-alias` kapcsolót is érdemes megadnunk mellette. A pontosabb részletek tekintetében olvassuk el a megfelelõ man oldalt.

Az /etc/rc.conf állományban a `NO` érték megadásával tiltsuk le az útválasztást végzõ program használatát:

```
router_enable="NO"
```

Fontos, hogy a `routed` démon ne induljon el, mivel `routed` hajlamos törölni a `ppp` által létrehozott alapértelmezett útválasztási bejegyzéseket.

Ezenkívül még a `sendmail_flags` változóról szóló sorból is érdemes kivenni a `-q` opciót, máskülönben a `sendmail` minden mûvelet megkezdése elõtt nekiáll felderíteni a hálózatot, és ezzel megindítja a tárcsázást. Próbáljuk meg így átírni az értékét:

```
sendmail_flags="-bd"
```

Ezért cserébe viszont a `sendmail` programot a ppp kapcsolat létrejöttekor mindig utasítanunk kell, hogy újból ellenõrizze a levelezési sort. Ezt a következõk begépelésével érhetjük el:

```
# /usr/sbin/sendmail -q
```

Ugyanezt automatikusan is meg tudjuk tenni a `!bg` paranccsal a ppp.linkup állományban:

```
1     szolgaltato:
2       delete ALL
3       add 0 0 HISADDR
4       !bg sendmail -bd -q30m
```

Ha nem felelne meg ez a megoldás, akkor egy "dfilter" is beállítható az SMTP forgalom szûrésére. A példák között megtaláljuk ennek pontos minkéntjét.

Ezután már csak a gépünk újraindítása maradt hátra. Az újraindítás után már be is gépelhetjük:

```
# ppp
```

ahol a `dial szolgaltato` parancs kiadásával meg tudjuk kezdeni a PPP kapcsolat felépítését, vagy a `ppp` programot megkérhetjük arra, hogy automatikusan kezdje el, amint van kimenõ forgalom (és nem készítettük el a start_if.tun0 szkriptet). Ekkor gépeljük be ezt:

```
# ppp -auto szolgaltato
```

Összefoglalás

Gyorsan foglaljuk össze, hogy az ppp beállításához milyen lépések megtétele szükséges az elsõ alkalommal:

A kliens oldalán:

1. Gyõzõdjünk meg róla, hogy a tun eszköz benne van a rendszermagban.
2. Ellenõrizzük, hogy a tunN eszközhöz tartozó állomány rendelkezésre áll a /dev könyvtárban.
3. Hozzunk létre egy bejegyzést az /etc/ppp/ppp.conf állományban. A pmdemand példából a legtöbb szolgáltató esetében ki tudunk indulni.
4. Ha dinamikus IP-címet kapunk, akkor az /etc/ppp/ppp.linkup állományba is vegyünk fel egy bejegyzést.
5. Frissítsük az /etc/rc.conf állományunkat.
6. Ha igény szerint akarunk tárcsázni, akkor hozzunk létre start_if.tun0 néven egy szkriptet.

A szerver oldalán:

1. Gondoskodjunk róla, hogy a tun eszköz támogatása szerepel rendszermagban.
2. Gyõzõdjünk meg róla, hogy a tunN eszköz megtalálható a /dev könyvtárban.
3. Az /etc/passwd állományban (a vipw(8) program használatával) hozzunk létre bejegyzéseket.
4. A felhasználók könyvtáraiban hozzunk létre egy olyan profilt, amely `ppp -direct direct-server` vagy egy ehhez hasonló parancsot futtat le.
5. Az /etc/ppp/ppp.conf állományban adjuk meg egy bejegyzést. A direct-server példa ehhez egy remek alapot biztosít.
6. Az /etc/ppp/ppp.linkup állományban hozzunk létre egy bejegyzést.
7. Frissítsük az /etc/rc.conf állományunkat.

27.3. A rendszerszintû PPP alkalmazása

Ez a szakasz csak FreeBSD 7.*X* esetén érvényes.

27.3.1. A rendszerszintû PPP beállítása

Mielõtt a gépünkön nekikezdünk a PPP beállításának, ellenõrizzük, hogy a `pppd` megtalálható a /usr/sbin könyvtárban és az /etc/ppp könyvtár létezik.

A `pppd` két módban képes mûködni:

1. "kliensként" - a gépünket soros vonali vagy modemes PPP kapcsolaton keresztül csatlakoztatjuk a külvilághoz
2. "szerverként" - a számítógépünk egy hálózat része, ahol a többieket a PPP használatával kapcsoljuk össze

Mind a két esetben egy konfigurációs állomány tartalmát kell összeállítanunk (ez az

/etc/ppp/options vagy a ~/.ppprc, ha a gépünkön több felhasználó is PPP-t akar használni).

Egy modemes vagy soros vonali szoftverre is szükségünk lesz (ez többnyire a comms/kermit), amellyel távoli gépeket tudunk felhívni és feléjük kapcsolatot felépíteni.

27.3.2. A pppd mint kliens

A most következõ /etc/ppp/options állománnyal egy Cisco terminál szerverhez tudunk kapcsolódni egy PPP vonalon keresztül.

```
crtscts         # a hardveres forgalomirányítás engedélyezése
modem           # modem vezérlõvonal
noipdefault     # a távoli PPP szervernek kell IP-címet adnia
                # ha az IPCP alapú egyeztetés során a távoli gép nem küld
                # nekünk IP-címet, akkor vegyük ki ezt a beállítást
passive         # LCP csomagokat várunk
domain ppp.ize.com      # ide írjuk be a hálózati nevünket

:távoli_ip    # ide kell írni a távoli PPP szerver IP-címét
                # a PPP kapcsolaton keresztül erre fogjuk továbbküldeni a csomagokat
                # ha nem adtuk meg "noipdefault" beállítást, akkor ezt a sort
                # írjuk át helyi_ip:távoli_ip alakúra

defaultroute    # adjuk meg ezt a sort is, ha a PPP szerverünket egyben az
                # alapértelmezett átjárónak is be akarjuk állítani
```

Így kapcsolódunk:

1. Tárcsázzuk a távoli gépet a Kermit (vagy bármilyen más modemes program) elindításával, majd adjuk meg a felhasználói nevünket és jelszavunkat (vagy bármi mást, amivel a távoli gépen engedélyezni tudjuk a PPP használatát).
2. Lépjünk ki a Kermit programból (anélkül, hogy bontanánk a vonalat).
3. Írjuk be a következõket:

   ```
   # /usr/sbin/pppd /dev/tty01 19200
   ```

   Ne felejtsük el megadni a megfelelõ sebességet és eszközt.

A számítógépünk most már PPP-n keresztül csatlakozik. Ha valamilyen okból nem sikerülne felépíteni a kapcsolatot, akkor vegyük fel a debug beállítást is az /etc/ppp/options állományba, majd a konzolra érkezõ üzenetek segítségével próbáljuk meg felderíteni a probléma okát.

Az alábbi /etc/ppp/pppup szkript mind a három fázist automatikussá teszi:

```
#!/bin/sh
```

```
pgrep -l pppd
pid=`pgrep pppd`
if [ "X${pid}" != "X" ] ; then
        echo 'killing pppd, PID=' ${pid}
        kill ${pid}
fi
pgrep -l kermit
pid=`pgrep kermit`
if [ "X${pid}" != "X" ] ; then
        echo 'killing kermit, PID=' ${pid}
        kill -9 ${pid}
fi

ifconfig ppp0 down
ifconfig ppp0 delete

kermit -y /etc/ppp/kermit.dial
pppd /dev/tty01 19200
```

Az /etc/ppp/kermit.dial egy olyan Kermit szkript, amivel tárcsázni tudunk és a távoli gépen elvégezni az összes szükséges hitelesítést (a leírás végén találhatunk is egy ilyen szkriptet példaként).

Az alábbi /etc/ppp/pppdown szkripttel tudjuk bontani a PPP vonalat:

```
#!/bin/sh
pid=`pgrep pppd`
if [ X${pid} != "X" ] ; then
        echo 'killing pppd, PID=' ${pid}
        kill -TERM ${pid}
fi

pgrep -l kermit
pid=`pgrep kermit`
if [ "X${pid}" != "X" ] ; then
        echo 'killing kermit, PID=' ${pid}
        kill -9 ${pid}
fi

/sbin/ifconfig ppp0 down
/sbin/ifconfig ppp0 delete
kermit -y /etc/ppp/kermit.hup
/etc/ppp/ppptest
```

A /usr/etc/ppp/ppptest elindításával ellenõrizni tudjuk, hogy a `pppd` még mindig fut. Ez valahogy így néz ki:

```
#!/bin/sh
pid=`pgrep pppd`
```

```
if [ X${pid} != "X" ] ; then
        echo 'pppd running: PID=' ${pid-NONE}
else
        echo 'No pppd running.'
fi
set -x
netstat -n -I ppp0
ifconfig ppp0
```

A vonal bontásához az /etc/ppp/kermit.hup szkriptet kell elindítanunk, amiben a következõ szerepelnek:

```
set line /dev/tty01 ; ide írjuk be a saját modemünket
set speed 19200
set file type binary
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none

pau 1
out +++
inp 5 OK
out ATH0\13
echo \13
exit
```

A `kermit` helyett a `chat` programot is használhatjuk:

A következõ két állomány már elég egy kapcsolat létrehozásához `pppd` használatával:

/etc/ppp/options:

```
/dev/cuad1 115200

crtscts     # a hardveres forgalomirányítás engedélyezése
modem       # modemes vezérlõvonal
connect "/usr/bin/chat -f /etc/ppp/login.chat.script"
noipdefault # a távoli PPP kiszolgálónak adnia kell egy IP-címet
            # ha a távoli gép nem küldi az IP-címünk az IPCP alapú egyeztetés során
                # akkor távolítsuk el ezt a beállítást
passive         # LCP csomagokat várunk
domain sajat.tartomany  # ide írjuk be a saját tartománynevünket

:       # a távoli PPP kiszolgáló IP-címét tegyük ide
```

```
            # ezen keresztül fogjuk továbbküldeni a PPP kapcsolaton áthaladó
csomagokat
                # nem adtuk meg a "noipdefault" beállítást, akkor ezt
                # sort írjuk át helyi_ip:távoli_ip alakúra

defaultroute    # ez a sor akkor kell, ha a PPP szerver lesz az
            # alapértelmezett átjárónk is
```

/etc/ppp/login.chat.script:

A most következõt egyetlen sorba kell írnunk.

```
ABORT BUSY ABORT 'NO CARRIER' "" AT OK ATDTtelefon.szám
  CONNECT "" TIMEOUT 10 ogin:-\\r-ogin: bejelentkezési-azonosító
  TIMEOUT 5 sword: jelszó
```

Miután ezeket telepítettük és a megfelelõképpen módosítottuk, már csak a `pppd` parancsot kell kiadnunk, valahogy így:

```
# pppd
```

27.3.3. A `pppd` mint szerver

Az /etc/ppp/options állományban nagyjából a következõknek kell szerepelnie:

```
crtscts                         # hardveres forgalomirányítás
netmask 255.255.255.0           # hálózati maszk (nem kötelezõ)
192.114.208.20:192.114.208.165  # a helyi és távoli gépek IP-címei
                                # a helyi IP-nek el kell térnie az Ethernet
                                # (vagy más egyéb) felülethez tartozó címtõl.
                                # a távoli IP a távoli géphez rendelt IP-cím
domain ppp.ize.com              # a saját tartományunk
passive                         # az LCP csomagok várása
modem                           # modemes vonal
```

Az alábbi /etc/ppp/pppserv szkript a pppd démont szervernek állítja be:

```
#!/bin/sh
pgrep -l pppd
pid=`pgrep pppd`
if [ "X${pid}" != "X" ] ; then
        echo 'killing pppd, PID=' ${pid}
        kill ${pid}
fi
pgrep -l kermit
```

```
pid=`pgrep kermit`
if [ "X${pid}" != "X" ] ; then
        echo 'killing kermit, PID=' ${pid}
        kill -9 ${pid}
fi

# reset ppp interface
ifconfig ppp0 down
ifconfig ppp0 delete

# enable autoanswer mode
kermit -y /etc/ppp/kermit.ans

# run ppp
pppd /dev/tty01 19200
```

A szerver leállítására a következõ /etc/ppp/pppservdown szkriptet kell használnunk:

```
#!/bin/sh
pgrep -l pppd
pid=`pgrep pppd`
if [ "X${pid}" != "X" ] ; then
        echo 'killing pppd, PID=' ${pid}
        kill ${pid}
fi
pgrep -l kermit
pid=`pgrep kermit`
if [ "X${pid}" != "X" ] ; then
        echo 'killing kermit, PID=' ${pid}
        kill -9 ${pid}
fi
ifconfig ppp0 down
ifconfig ppp0 delete

kermit -y /etc/ppp/kermit.noans
```

A következõ Kermit szkript (/etc/ppp/kermit.ans) engedélyezi vagy tiltja le a modem automatikus válaszadását. Körülbelül így épül fel:

```
set line /dev/tty01
set speed 19200
set file type binary
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
```

```
set flow none

pau 1
out +++
inp 5 OK
out ATH0\13
inp 5 OK
echo \13
out ATS0=1\13   ; "ATS0=0\13"-ra írjuk át, ha le akarjuk tiltani az
                ; automatikus válaszadást
inp 5 OK
echo \13
exit
```

Az /etc/ppp/kermit.dial elnevezésû szkriptet használhatjuk arra, hogy tárcsázzunk távoli gépeket és hitelesítsük magunkat rajtuk. Írjuk át az igényeinknek megfelelõen, tegyük bele a bejelentkezéshez szükséges azonosítót és jelszót, illetve a modemünk és a távoli gép válaszai szerint módosítsuk az `input` utasításokat.

```
;
; írjuk ide azt a com vonalat, amire a modemünk csatlakozik:
;
set line /dev/tty01
;
; ide kerül a modem sebessége:
;
set speed 19200
set file type binary            ; teljes 8 bites állomány-átvitel
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none
set modem hayes
set dial hangup off
set carrier auto                ; adjuk meg a SET CARRIER utasítást is, ha kell
set dial display on             ; adjuk meg a SET DIAL utasítást is, ha kell
set input echo on
set input timeout proceed
set input case ignore
def \%x 0                       ; a bejelentkezés számlálója
goto slhup

:slcmd                          ; tegyük a modemet parancs módba
echo Tegyuk a modemet parancs modba.
clear                           ; töröljük a be nem olvasott karaktereket a bemeneti
pufferbõl
```

```
pause 1
output +++                      ; a Hayes-féle helyettesítési szekvenciák használata
input 1 OK\13\10                ; várjuk meg az OK jelzést
if success goto slhup
output \13
pause 1
output at\13
input 1 OK\13\10
if fail goto slcmd              ; ha a modem nem válaszol OK-val, akkor próbálkozzunk
újra

:slhup                          ; bontsuk a vonalat
clear                           ; töröljük ki a be nem olvasott karaktereket a
bemeneti pufferbõl
pause 1
echo A vonal bontasa.
output ath0\13                  ; a kapcsolat létrejöttét jelzõ Hayes-parancs
input 2 OK\13\10
if fail goto slcmd              ; ha nincs OK válasz, akkor tegyük a modemet parancs
módba

:sldial                         ; tárcsázzuk a számot
pause 1
echo Dialing.
output atdt9,550311\13\10       ; ide írjuk a telefonszámot
assign \%x 0                    ; nullázzuk le az idõzítõt

:look
clear                           ; töröljük az olvasatlan karaktereket a bemeneti
pufferbõl
increment \%x                   ; számoljuk a másodperceket
input 1 {CONNECT }
if success goto sllogin
reinput 1 {NO CARRIER\13\10}
if success goto sldial
reinput 1 {NO DIALTONE\13\10}
if success goto slnodial
reinput 1 {\255}
if success goto slhup
reinput 1 {\127}
if success goto slhup
if < \%x 60 goto look
else goto slhup

:sllogin                        ; bejelentkezés
assign \%x 0                    ; nullázzuk le az idõzítõt
pause 1
echo A bejelentkezes keresese.

:slloop
increment \%x                   ; számoljuk a másodperceket
```

```
clear                           ; töröljük az olvasatlan karaktereket a bemeneti
pufferbõl
output \13
;
; ide írjuk be a várható bejelentkezési sablont:
;
input 1 {Felhasznaloi nev: }
if success goto sluid
reinput 1 {\255}
if success goto slhup
reinput 1 {\127}
if success goto slhup
if < \%x 10 goto slloop      ; tízszer próbálkozzunk a bejelentkezéssel
else goto slhup                 ; 10 sikertelen próbálkozás után bontsuk a vonalat és
kezdjük újra

:sluid
;
; ide írjuk be a felhasználói azonosítónkat:
;
output ppp-login\13
input 1 {Jelszo: }
;
; ide tegyük a hozzá tartozó jelszót:
;
output ppp-password\13
input 1 {Atvaltas SLIP modba.}
echo
quit

:slnodial
echo \7Nincs vonal.  Ellenorizzuk a telefonvonalat!\7
exit 1

; local variables:
; mode: csh
; comment-start: "; "
; comment-start-skip: "; "
; end:
```

27.4. PPP kapcsolatok hibaelhárítása

A FreeBSD 8.0 kiadásától kezdõdõen a sio(4) meghajtó szerepét a uart(4) veszi át. Emiatt a soros vonali eszközöket /dev/cuadN és /dev/cuauN helyett /dev/ttydN és /dev/ttyuN néven lehet elérni. A FreeBSD 7.*X* változatok felhasználóinak ennek megfelelõen kell olvasniuk ezt a leírást.

Ebben a szakaszban összefoglalunk néhány olyan problémát, ami a PPP modemen keresztüli használata során keletkezhet. Például pontosan tisztában kell lennünk azzal, hogy a tárcsázott

rendszer milyen adatokat és hogyan fog tõlünk bekérni. Egyes szolgáltatók egy `ssword` promptot, míg mások egy `password` promptot adnak. Ha a `ppp` szkript nem illeszkedik ezekhez az elvárásokhoz, akkor nem tudunk bejelentkezni. A `ppp` csatlakozások nyomonkövetésének egyik leggyakoribb módja a manuális kapcsolódás. A következõkben ezért a manuális csatlakozásokra vonatkozó legszükségesebb ismereteket mutatjuk be lépésrõl lépésre.

27.4.1. Az eszközleírók ellenõrzése

Ha saját rendszermagot használunk, ne felejtsük el felvenni a következõ sort a konfigurációs állományba:

```
device   uart
```

A `GENERIC` rendszermag az uart eszközt már alapértelmezés szerint tartalmazza, ezért ilyenkor már nincs több teendõnk. Egyszerûen csak a `dmesg` parancs kimenetében keressük meg a modemes eszközhöz tartozó adatokat:

```
# dmesg | grep uart
```

Ennek eredményeképpen kapunk egy rövid összefoglalást a uart típusú eszközökrõl. Ezek lesznek a számunkra fontos COM portok. Amennyiben a modemünk egy szabványos soros portként mûködik, akkor a uart1 vagy COM2 néven kell keresnünk. Ha megtaláltuk, akkor nem kell új rendszermagot fordítanunk. Amikor a soros vonali modemünk a uart1 vagy COM2 porton csatlakozik DOS-ban, akkor itt a neki megfelelõ eszköz a /dev/cuau1 lesz.

27.4.2. Kapcsolódás manuálisan

A `ppp` kézi irányításával gyorsan, egyszerûen és minden fájdalomtól mentesen tudunk csatlakozni az internethez, de olyankor is hasznos, ha ki akarjuk deríteni, hogy az internet-szolgáltatónk milyen módon kezeli a kliensek `ppp` csatlakozásait. Nos, akkor ehhez indítsuk is el a PPP alkalmazást a paranccsorból. Az alábbi példákban rendre a *pelda* névvel hivatkozunk a PPP-t mûködtetõ gépre. A `ppp` tehát a `ppp` parancs begépelésével indítható:

```
# ppp
```

Ezzel elindítottuk a `ppp` programot.

```
ppp ON pelda> set device /dev/cuau1
```

Beállítjuk a modemünket, ami ebben az esetben a cuau1.

```
ppp ON pelda> set speed 115200
```

Beállítjuk a csatlakozás sebességét, ami ebben az esetben 115 200 kbit/mp.

```
ppp ON pelda> enable dns
```

Azt mondjuk a ppp programnak, hogy állítsa be a névfeloldót és az /etc/resolv.conf állományt egészítse ki a megfelelõ névszerverekkel. Ha a ppp nem képes megállapítani a gépünk nevét, akkor késõbb ezt még kézzel is be tudjuk állítani.

```
ppp ON pelda> term
```

Váltsunk "terminál" módba, így mi irányítjuk a modemet.

```
deflink: Entering terminal mode on /dev/cuau1
type '~h' for help
```

```
at
OK
atdt123456789
```

Az at paranccsal hozzuk alaphelyzetbe a modemet, majd a atdt paranccsal és egy telefonszám megadásával megkezdjük a szolgáltató tárcsázását.

```
CONNECT
```

Ezzel jelez vissza a kapcsolódás megkezdéséről. Ha itt bármilyen hardvertõl független csatlakozási probléma merülne fel, akkor ezen a ponton tudunk ellene tenni valamit.

```
ISP Login:felhasznalonev
```

Itt kell megadnunk a felhasználói nevünket, ami megegyezik a szolgáltató által adott azonosítónkkal.

```
ISP Pass:jelszo
```

Ezúttal a jelszavunkat kell megadni, amit szintén a szolgáltató bocsátott rendelkezésünkre az azonosító mellett. Akárcsak amikor bejelentkezünk a FreeBSD-be, itt sem fog látszódni a jelszavunk.

```
Shell or PPP:ppp
```

Szolgáltatótól függõen elõfordulhat, hogy ez a sor soha nem is jelenik meg. Itt kérdezik meg, hogy a szolgáltatónál egy shellt akarunk használni, vagy csak elindítani egy ppp kapcsolatot. Ebben a példában természetesen a ppp opciót választjuk, mivel egy internet-elõfizetés birtokosai vagyunk.

```
Ppp ON pelda>
```

Figyeljük meg, hogy az elsõ `p` nagybetûssé vált. Ezzel jelzi a program, hogy sikeresen csatlakoztunk a szolgáltatónkhoz.

```
PPp ON pelda>
```

Sikeresen azonosítottuk magunkat a szolgáltató felé és várjuk az IP-címünket.

```
PPP ON pelda>
```

Megkaptuk az IP-címünket és ezzel sikeresen felépült a kapcsolat.

```
PPP ON pelda>add default HISADDR
```

Itt adjuk hozzá az alapértelmezett útvonalat, amire mindenképpen szükségünk van ahhoz, hogy a külvilággal is kapcsolatban tudjunk lépni, mivel jelenleg csak a vonal másik végén lévõ gépet érjük el. Ha ezt bizonyos, már meglevõ útvonalak miatt nem sikerül felvenni, akkor az `add` elé tegyünk egy `!` jelet. Ezt viszont a kapcsolat felépítése elõtt is megtehetjük, így menet közben az új útvonalat felveszi a többi közé.

Ha eddig minden remekül ment, akkor ezen ponton már egy élõ internet-kapcsolattal rendelkezünk, és a programot a `CTRL` + `z` lenyomásával a háttérbe is tehetjük. Ha a `PPP` felirat ismét a `ppp` feliratra váltana, akkor az arra utal, hogy elvesztettük a kapcsolatot. Erre nem árt figyelni, mivel ezzel jelzi az aktuális kapcsolat állapotát. A nagybetûs P-k jelölik, hogy az adott szinten megvan a kapcsolat a szolgáltató felé, a kisbetûs p-k pedig arra utalnak, hogy azon a szinten a kapcsolat valamiért megszûnt. A `ppp` csak ezt a két állapotot ismeri.

Nyomkövetés

Ha közvetlen vonalunk van és mégsem sikerül kapcsolatot létesíteni, akkor tiltsuk le a hardveres CTS/RTS forgalomirányítást a `set ctsrts off` paranccsal. Ez leginkább akkor fordul elõ, ha csatlakoztunk egy olyan terminálszerverhez, amely valamennyire képes kezelni a PPP kapcsolatokat, de a PPP megáll, mikor adatot próbál írni a kommunikációs csatornára, mivel arra a CTS (Clear To Send - "lehet küldeni") jelzésre vár, amely soha nem fog megérkezni. Ha mégis ezt a beállítást akarjuk használni, akkor a `set accmap` beállításra is szükségünk lesz, mivel ez kell bizonyos karakterek hardverfüggõ átküldésének felülbírálásához, legtöbb esetben a XON/XOFF miatt. A ppp(8) man oldalon találhatunk errõl és ennek használatáról részletesebb leírást.

Ha egy régebbi gyártmányú modemünk van, akkor a `set parity even` beállítás alkalmazása is javasolt. Alapértelmezés szerint ugyanis nincs paritás, de a régebbi modemek és (a forgalom növekedésével) egyes szolgáltatók még használják hibaellenõrzésre. Ha Compuserve elõfizetésünk van, mindenképpen kapcsoljuk be.

Amikor a PPP nem tér vissza parancs módba, akkor gyaníthatóan az egyeztetésben lesz valahol

probléma, mivel a szolgáltató a kliensüktõl várja a kezdeményezését. Ezen a ponton a `~p` paranccsal utasíthatjuk a ppp programot a konfigurációs információk átküldésének megkezdésére.

Ha egyáltalán nem kapunk promptot a bejelentkezéshez, akkor nagy a alószínûsége, hogy az iménti UNIX® stílusú hitelesítés helyett PAP vagy CHAP protokollt kell használnunk. A PAP vagy CHAP használatához mindössze a következõ beállításokat kell megadnunk PPP programnak a terminál mód aktiválása elõtt:

```
ppp ON pelda>set authname felhasznalonev
```

ahol a *felhasznalonev* helyett a szolgáltatótól kapott azonosítót kell beírnunk.

```
ppp ON pelda>set authkey jelszo
```

ahol a *jelszo* helyett a szolgáltatótól kapott jelszót kell megadnunk.

Ha sikeresen csatlakoztunk, de még nem találunk semmilyen tartománynevet, akkor a ping(8) és IP-cím segítségével tudjuk megvizsgálni, hogy mûködõképes-e a kapcsolat. Ha 100 százalékos (100%) csomagvesztést (packet loss) tapasztalunk, akkor szinte biztos, hogy nincs meg az alapértelmezett útvonal. Nézzük meg újra, hogy az `add default HISADDR` beállítást megadtuk-e a kapcsolat felépítésekor. Ha viszont már el tudunk érni egy távoli IP-címet, akkor nagyon valószínû, hogy az /etc/resolv.conf állományba nem került bele a megfelelõ névfeloldó címe. Az említett állománynak valahogy így kellene kinéznie:

```
domain minta.com
nameserver x.x.x.x
nameserver y.y.y.y
```

Ahol az *x.x.x.x* és *y.y.y.y* címeket a szolgáltatónk névszervereinek címével kell behelyettesíteni. Ez nem minden esetben található meg az elõfizetõi szerzõdésben, de ha felhívjuk a szolgáltatónkat, akkor minden bizonnyal elárulják ezeket a címeket.

A syslog(3) is alkalmas a PPP kapcsolatok naplózására. Ehhez csupán ennyit kell megadnunk az /etc/syslog.conf állományban:

```
!ppp
*.*     /var/log/ppp.log
```

A legtöbb esetben ez a lehetõség már eleve adott.

27.5. A PPP használata Ethernet felett (PPPoE)

Ebben a szakaszban azt ismertetjük, hogyan állítsuk be a PPP-t Ethernet felett (PPP over Ethernet, PPPoE).

27.5.1. A rendszermag beállítása

A PPPoE mûködéséhez most már semmilyen módosításra nincs szükség a rendszermag beállításaiban. Amennyiben a hozzá szükséges Netgraph támogatás nem található a rendszermagban, akkor azt a ppp önmûködõen betölti.

27.5.2. A ppp.conf beállítása

Íme egy mûködõ ppp.conf állomány:

```
default:
  set log Phase tun command # itt akár egy részletesebb naplózást is be tudunk
állítani
  set ifaddr 10.0.0.1/0 10.0.0.2/0

a_szolgaltato_neve:
  set device PPPoE:xl1 # az xl1 helyére írjuk be a saját Ethernet eszközünket
  set authname FELHASZNALONEV
  set authkey JELSZO
  set dial
  set login
  add default HISADDR
```

27.5.3. A ppp futtatása

`root` felhasználóként adjuk ki az alábbi parancsot:

```
# ppp -ddial a_szolgaltato_neve
```

27.5.4. A ppp indítása a rendszerindítás során

Az /etc/rc.conf állományba vegyük fel a következõket:

```
ppp_enable="YES"
ppp_mode="ddial"
ppp_nat="YES"   # csak akkor, ha címfordítás kell a helyi hálózaton, máskülönben "NO"
ppp_profile="a_szolgaltato_neve"
```

27.5.5. A szolgáltatási címkék használata

Bizonyos esetekben szolgáltatási címkét (service tag) is használnunk kell a kapcsolat létrehozásához. A szolgáltatási címkék segítségével tudjuk megkülönböztetni az adott hálózaton elérhetõ különbözõ PPPoE szervereket.

A szolgáltatótól kapott dokumentációban szerepelnie kell minden ehhez kapcsolódó információnak. Amennyiben nem találjuk, érdeklõdjünk a szolgáltatónál.

Utolsó reményként megpróbálhatjuk a Portgyûjteményben található Roaring Penguin PPPoE nevû program által javasolt módszert. Ennél vegyük azonban számításba, hogy félre tudja programozni a modemünket, amitõl akár használhatatlanná is válhat, ezért kétszer is gondoljuk meg, mielõtt használni kezdjük. Egyszerûen csak tegyük fel a szolgáltatótól a modemünk mellé kapott szoftvert. Ezután lépjünk be a program **System** menüjébe. Itt kell lennie a megfelelõ profilnak, ami általában az *ISP*.

A profil neve (a szolgáltatás címkéje) a ppp.conf állományban a PPPoE bejegyzés részeként jelenik meg a `set device` parancsban (ennek pontos részleteit lásd a ppp(8) man oldalon). Tehát nagyjából így néz ki:

```
set device PPPoE:xl1:ISP
```

Az *xl1* eszköz nevét ne felejtsük el a megfelelõ Ethernet kártyához tartozó eszköz nevére kicserélni.

Az *ISP* helyett pedig írjuk be az imént kiderített profil nevét.

A témával kapcsolatban az alábbi helyeken találhatunk további információkat:

- Cheaper Broadband with FreeBSD on DSL, írta: Renaud Waldura (angolul).
- Nutzung von T-DSL und T-Online mit FreeBSD, írta: Udo Erdelhoff (németül).

27.5.6. PPPoE és a 3Com® HomeConnect™ ADSL Modem Dual Link

Ez a modem nem felel meg az RFC 2516 elõírásainak (*A Method for transmitting PPP over Ethernet (PPPoE)*, írta: L. Mamakos, K. Lidl, J. Evarts, D. Carrel, D. Simone és R. Wheeler). Helyette az Ethernet keretekben eltérõ csomagtípus kódokat használ. A 3Com-nál panaszkodjunk, ha szerintünk is be kellene tartaniuk a PPPoE specifikációját.

A FreeBSD is csak akkor lesz képes együttmûködni ezzel az eszközzel, ha beállítjuk a megfelelõ sysctl változót. Ezt a rendszerindítás során automatikusan meg tudjuk tenni az /etc/sysctl.conf módosításával:

```
net.graph.nonstandard_pppoe=1
```

vagy közvetlenül az alábbi paranccsal:

```
# sysctl net.graph.nonstandard_pppoe=1
```

Sajnos, mivel ez egy rendszerszintû beállítás, ezért a 3Com® HomeConnect™ ADSL Modem és más normális PPPoE kliens vagy szerver egyszerre nem használható.

27.6. PPP ATM felett (PPPoA)

Most a PPP ATM feletti (PPP over ATM, PPPoA) beállítását fogjuk bemutatni. A PPPoA az európai DSL szolgáltatók körében igen nagy népszerûségnek örvend.

27.6.1. PPPoA használata az Alcatel SpeedTouch™ USB-vel

Az ilyen eszközökhöz tartozó PPPoA támogatás a FreeBSD-ben portként áll rendelkezésre, mivel az ehhez szükséges firmware csak az Alcatel licencelési feltételei szerint terjeszthetõ, ezért nem lehet része az alap FreeBSD rendszernek.

A szoftver telepítéséhez ezért a Portgyûjteményt kell használnunk. Telepítsük a net/pppoa portot és kövessük a mellékelt utasításokat.

Sok más USB-s eszközhöz hasonlóan az Alcatel SpeedTouch™ USB-nek a gépünkrõl kell letöltenie a mûködéséhez szükséges firmware-t. Ez a folyamat FreeBSD alatt automatizálható, tehát ez a másolás minden esetben megtörténik, amikor az eszközt az USB portra csatlakoztatjuk. Ehhez az /etc/usbd.conf állományba a következõ adatokat kell beletennünk. Az állományt `root` felhasználóként tudjuk csak szerkeszteni.

```
device "Alcatel SpeedTouch USB"
    devname "ugen[0-9]+"
    vendor 0x06b9
    product 0x4061
    attach "/usr/local/sbin/modem_run -f /usr/local/libdata/mgmt.o"
```

Az usbd, vagyis az USB démon engedélyezéséhez az /etc/rc.conf állományba tegyük bele az alábbit:

```
usbd_enable="YES"
```

Emellett még a ppp kapcsolatot is be tudjuk állítani az indítás során. Ehhez mindössze a következõ sort kell megadnunk az /etc/rc.conf állományban. Ismét megemlítjük, hogy ezt a mûveletet csak a `root` felhasználóval tudjuk végrehajtani.

```
ppp_enable="YES"
ppp_mode="ddial"
ppp_profile="adsl"
```

Ezután úgy tudjuk szóra bírni a kapcsolatot, ha a net/pppoa porthoz mellékelt ppp.conf állományt használjuk fel kiindulásként.

27.6.2. Az mpd használata

Az mpd segítségével többféle szolgáltatáshoz, köztük a PPTP-hez hozzá tudunk férni. Az mpd a Portgyûjteményben net/mpd néven található meg. Sok ADSL modemnek szüksége van egy PPTP tunnelre közte és gép között. Ilyen modem például az Alcatel SpeedTouch™ Home is.

Elõször magát a portot kell telepítenünk, majd ezután már be tudjuk állítani az mpd-t a saját és a szolgáltatónk igényei szerint. A port a rengeteg leírással megtûzdelt minta konfigurációs állományait a PREFIX/etc/mpd/ könyvtárba teszi. Itt a *PREFIX* azt a könyvtárat jelöli, ahova a portok kerülnek. Ez alapból a /usr/local/. Az mpd beállításáról szóló teljes dokumentáció a telepítés után

elérhetõ HTML formátumban a PREFIX/shared/doc/mpd/ könyvtárban. Íme egy példa az mpd beállítására ADSL kapcsolatok esetében. Az ezzel kapcsolatos beállításaink két állományra bomlanak, melyek közül az elsõ az mpd.conf:

```
default:
    load adsl

adsl:
    new -i ng0 adsl adsl
    set bundle authname felhasználónév ①
    set bundle password jelszó ②
    set bundle disable multilink

    set link no pap acfcomp protocomp
    set link disable chap
    set link accept chap
    set link keep-alive 30 10

    set ipcp no vjcomp
    set ipcp ranges 0.0.0.0/0 0.0.0.0/0

    set iface route default
    set iface disable on-demand
    set iface enable proxy-arp
    set iface idle 0

    open
```

① A felhasználói azonosító, amellyel a szolgáltató felé hitelesítjük magunkat.

② Az azonosítóhoz tartozó jelszó, amelyet szintén a szolgáltatól kaptunk.

Az mpd.links állomány tartalmazza a felépítendõ kapcsolatra vagy kapcsolatokra vonatkozó információkat. Például az elõbbiekhez tartozó mpd.links tartalma ez:

```
adsl:
    set link type pptp
    set pptp mode active
    set pptp enable originate outcall
    set pptp self 10.0.0.1 ①
    set pptp peer 10.0.0.138 ②
```

① A FreeBSD-s számítógépünk címe, ahonnan az mpd indul.

② Az ADSL modemünk IP-címe. Az Alcatel SpeedTouch™ Home esetén ez a cím alapértelmezés szerint a `10.0.0.138`.

A kapcsolat ezek után pillanatok alatt felépíthetõ, ha a `root` felhasználóval kiadjuk a következõ parancsot:

```
# mpd -b adsl
```

A kapcsolat állapotát a következõ paranccsal tudjuk ezután ellenõrizni:

```
% ifconfig ng0
ng0: flags=88d1<UP,POINTOPOINT,RUNNING,NOARP,SIMPLEX,MULTICAST> mtu 1500
     inet 216.136.204.117 --> 204.152.186.171 netmask 0xffffffff
```

FreeBSD alatt az mpd használata ajánlott az ADSL szolgáltatások eléréséhez.

27.6.3. A pptpclient használata

FreeBSD alatt a net/pptpclient segítségével is tudunk PPPoA típusú szolgáltatásokhoz kapcsolódni.

A net/pptpclient felhasználásával úgy tudunk DSL szolgáltatásokat elérni, ha feltelepítjük a hozzá tartozó portot vagy csomagot, majd módosítjuk az /etc/ppp/ppp.conf állományt. Mind a két mûveletet csak `root` felhasználóként tudjuk lebonyolítani. Ehhez egy ppp.conf állományt lentebb adtunk meg. A ppp.conf állományban található további beállítási lehetõségekrõl a ppp(8) man oldalon olvashatunk.

```
adsl:
 set log phase chat lcp ipcp ccp tun command
 set timeout 0
 enable dns
 set authname felhasználónév ①
 set authkey jelszó ②
 set ifaddr 0 0
 add default HISADDR
```

① A DSL szolgáltatónktól kapott felhasználói név.

② Az elõfizetéshez tartozó jelszó.

Mivel az elõfizetéshez tartozó jelszót a ppp.conf állományba titkosítatlan formában kell szerepeltetnünk, ezért gondoskodjunk róla, hogy senki sem képes olvasni a tartalmát. A most következõ parancsokkal beállítjuk, hogy ez az állomány csak a `root` felhasználó számára legyen olvasható. A részletekért lásd a chmod(1) és chown(8) man oldalakat.

```
# chown root:wheel /etc/ppp/ppp.conf
# chmod 600 /etc/ppp/ppp.conf
```

Ezzel a paranccsal a DSL útválasztónk felé nyitunk egy tunnelt a PPP kapcsolathoz. Az Ethernetes DSL modemek általában egy elõre beállított helyi hálózati IP-címmel rendelkeznek, amelyhez tudunk csatlakozni. Az Alcatel SpeedTouch™ Home esetében ez a cím a `10.0.0.138`. Az útválasztóhoz adott dokumentációban keressük meg, hogy az eszközünkhöz konkrétan milyen cím

tartozik. A tunnel megnyitásához és a PPP kapcsolat megindításához a következõ parancsot kell kiadnunk:

```
# pptp cím adsl
```

Az iménti parancs végére még érdemes odatenni az "et" jelet ("&") is, mivel így a pptp mûködését a háttérben folytatja.

A parancs hatására a virtuális tunnelt megtestesítõ tun eszköz jön létre a pptp és ppp programok között. Miután visszakaptuk a parancssort, vagy a pptp program megerõsítette a kapcsolódás sikerességét, a keletkezett járatot így tudjuk ellenõrizni:

```
% ifconfig tun0
tun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1500
        inet 216.136.204.21 --> 204.152.186.171 netmask 0xffffff00
        Opened by PID 918
```

Ha nem tudnánk valamiért csatlakozni, akkor elõször nézzük meg az útválasztónk beállításait, ami általában a telnet vagy egy böngészõ segítségével elérhetõ. Ha még mindig nem vagyunk képesek csatlakozni, akkor a `pptp` parancs kimenetében és ppp /var/log/ppp.log néven elérhetõ naplójában kereshetünk árulkodó nyomokat.

27.7. A SLIP használata

Ez a szakasz csak FreeBSD 7.*X* rendszerekre érvényes.

27.7.1. A SLIP kliensek beállítása

A következõkben azt mutatjuk be, hogy egy FreeBSD-s gépet miként tudunk egy hálózaton statikus névvel beállítani a SLIP használatával. A dinamikus hálózati nevek használatakor (vagyis amikor a címünk minden egyes tárcsázáskor megváltozhat) egy valamivel bonyolultabb beállításra van szükségünk.

Elõször is állapítsuk meg, hogy a modemünk melyik soros portra csatlakozik. Sokan /dev/modem néven egy szimbolikus linket hoznak létre a valódi eszközre, például a /dev/cuadN leíróra. Ennek köszönhetõen az eszköz tényleges névetõl el tudunk vonatkoztatni és soha nem kell módosítanunk semmit, ha a modemet például egy másik portra kell átraknunk. Ugyanis könnyedén kacifántossá tud válni a helyzet, amikor egyszerre kell megváltoztatnunk egy rakat dolgot az /etc könyvtárban és módosítanunk az összes .kermrc állományt!

A /dev/cuad0 a COM1 port, a /dev/cuad1 a COM2 és így tovább.

A rendszermag beállításait tartalmazó állományban a következõnek mindenképpen szerepelnie kell:

```
device   sl
```

Mivel ez általában a GENERIC rendszermagban megtalálható, így ez nem okoz semmilyen gondot, kivéve, hogy ha korábban már kitöröltük.

Amit csak egyszer kell megtenni

1. Vegyük fel az otthoni gépünket, az átjárónkat és a névszervereket az /etc/hosts állományba. Erre álljon itt egy konkrét példa:

   ```
   127.0.0.1               localhost loghost
   136.152.64.181          water.CS.Example.EDU water.CS water
   136.152.64.1            inr-3.CS.Example.EDU inr-3 slip-gateway
   128.32.136.9            ns1.Example.EDU ns1
   128.32.136.12           ns2.Example.EDU ns2
   ```

2. Figyeljünk oda, hogy az /etc/nsswitch.conf állományban szereplõ `hosts` szakaszban a `dns` szó elõtt a `files` szónak kell megjelennie. Ezek nélkül mókás dolgok tudnak történni rendszerünkben.
3. Szerkesszük át az /etc/rc.conf állományt.
   a. A hálózati nevünket a következõ sorban tudjuk megadni:

      ```
      hostname="az.en.nevem"
      ```

      Ide a gépünk teljes internetes hálózati nevét kell beírnunk.

   b. Az alapértelmezett átjárót az alábbi sor módosításával tudjuk beállítani úgy, hogy a

      ```
      defaultrouter="NO"
      ```

      változó értékét átírjuk:

      ```
      defaultrouter="slip-gateway"
      ```

4. Készítsük el az /etc/resolv.conf állományt, amelyben majd a következõk legyenek:

   ```
   domain CS.Example.EDU
   nameserver 128.32.136.9
   nameserver 128.32.136.12
   ```

   Látható, hogy ezek a névfeloldásért felelõs szerverek címei. Természetesen a ténylegesen beírandó tartomány (domain) neve és a névszerverek címei mindig az adott

környezetünktõl függenek.

5. Állítsuk be egy jelszót a `root` és `toor` felhasználóknak (és mindenki másnak, akinek még nem lenne).
6. Indítsuk újra a számítógépünket és utána gyõzõdjünk meg róla, hogy a megfelelõ hálózati névvel rendelkezik.

A SLIP kapcsolatok felépítése

1. Tárcsázzunk és gépeljük be a `slip` parancsot, majd ezt követõen a gépünk nevét és a jelszót. Ez leginkább a konkrét környezettõl függ. Ha a Kermit nevû programot használjuk, akkor egy ilyen szkripttel is próbálkozhatunk:

```
# a kermit beállítása
set modem hayes
set line /dev/modem
set speed 115200
set parity none
set flow rts/cts
set terminal bytesize 8
set file type binary
# a következõ makró felelõs a tárcsázásért és a bejelentkezésért
define slip dial 643-9600, input 10 =>, if failure stop, -
output slip\x0d, input 10 Azonosito:, if failure stop, -
output silvia\x0d, input 10 Jelszo:, if failure stop, -
output ***\x0d, echo \x0aCONNECTED\x0a
```

   Természetesen a felhasználói nevet és a jelszót a sajátunkra kell benne kicserélnünk. Miután ezzel is megvagyunk, a Kermit paranccsorában a csatlakozáshoz egyszerûen csak írjuk be, hogy `slip`.

   Nem javasoljuk, hogy az állományrendszeren a jelszavakat titkosítatlan formában tároljuk. Mindeki csak a saját felelõsségére tegyen ilyet.

2. Hagyjuk el a Kermit programot (a `Ctrl` + `z` billentyûkombinációval bármikor fel tudjuk függeszteni a futását) és `root` felhasználóként írjuk be a következõt:

```
# slattach -h -c -s 115200 /dev/modem
```

   Ha ezután már képesek vagyunk a `ping` paranccsal elérni az útválasztó másik oldalán található gépet, akkor az azt jelenti, hogy sikerült csatlakoznunk! Ha viszont itt még nem járnánk sikerrel, akkor az `slattach` parancsnak ne a `-c` paramétert adjuk meg, hanem a `-a` paramétert.

Hogyan bontsunk egy kapcsolatot

Tegyük a következõket:

```
# kill -INT `cat /var/run/slattach.modem.pid`
```

Ez leállítja az `slattach` programot. Ne felejtsük el azonban, hogy ezt csak a `root` felhasználóval tudjuk végrehajtani. Ezután térjünk vissza a `kermit` programhoz (ha felfüggesztettük volna, akkor ehhez a `fg` parancsra lesz szükségünk), és lépjünk ki belõle (`q`).

Az slattach(8) man oldala ehhez a `ifconfig sl0 down` parancsot javasolja, amellyel lényegében leállítjuk a hozzá tartozó felületet. Igazából a kettõ között nincs semmilyen komolyabb eltérés (mivel az (`ifconfig sl0` is ugyanezt eredményezi.)

Néha elõfordulhat, hogy a modem egyszerûen nem hajlandó eldobni a vonalat. Ilyen esetekben indítsuk el a `kermit` programot és lépjünk ki megint. Másodjára általában már sikerül.

Hibaelhárítás

Ha valamiért ez mégsem válna be, akkor csak nyugodtan kérdezõsködjünk a freebsd-net levelezési listán. A tapasztalatok szerint az embereknek eddig a következõkkel voltak problémáik:

- Az `slattach` meghívásakor sem a `-c`, sem pedig a `-a` paramétert nem adták meg. (Ez ugyan nem végzetes hiba, de egyes felhasználók szerint ez segített megoldani a gondokat.)
- Az `sl0` helyett `s10`-et írtak be (egyes betûtípusoknál könnyen össze lehet téveszteni ezeket).
- Az `ifconfig sl0` segítségével ellenõrizhetõ a felület állapota. Például ilyet láthatunk:

  ```
  # ifconfig sl0
  sl0: flags=10<POINTOPOINT>
          inet 136.152.64.181 --> 136.152.64.1 netmask ffffff00
  ```

- Ha a ping(8) `no route to host` hibaüzenetet ad, akkor az útválasztási táblázattal van a gond. A `netstat -r` paranccsal gyorsan ki tudjuk listázni a rendszerünkben jelenleg nyilvántartott utakat:

  ```
  # netstat -r
  Routing tables
  Destination      Gateway           Flags     Refs     Use  IfaceMTU    Rtt
  Netmasks:

  (root node)
  (root node)

  Route Tree for Protocol Family inet:
  (root node) =>
  default          inr-3.Example.EDU  UG          8   224515  sl0 -      -
  localhost.Exampl localhost.Example. UH          5    42127  lo0 -       0.438
  ```

```
inr-3.Example.ED water.CS.Example.E UH          1        0  sl0 -      -
water.CS.Example localhost.Example. UGH        34 47641234  lo0 -       0.438
(root node)
```

Az elõzõ példákat egy viszonylag forgalmas rendszerbõl ragadtuk ki. A rendszerünkön megjelenõ számok a hálózati aktivitás mértékének függvényei.

27.7.2. A SLIP szerverek beállítása

Ebben a leírásban igyekszünk bemutatni hogyan kell egy FreeBSD típusú rendszer alatt SLIP szervert beállítani, ami általában annyit jelent, hogy a rendszerünben a távoli SLIP kliensek csatlakozásakor automatikusan elindítjuk a kapcsolatokat.

Elõfeltételek

Ez a szakasz igen szakmai jellegû, ezért az olvasó részérõl feltételezünk a témában némi alapismeretet. Ez alatt alapvetõen a TPC/IP hálózati protokollt értjük, különös hangsúllyal a hálózatok és hálózati csomópontok címzéséen, a hálózati maszkokon, alhálózatokon, útválasztáson, az olyan útválasztási protokollokon, mint például a RIP. A SLIP beállítása egy betárcsázós szerveren mindezen fogalmak ismeretét igényli, és ha ezekkel még nem lennénk tisztában, akkor olvassuk el például Craig Hunt *TCP/IP Network Administration* címû könyvét (O'Reilly & Associates, Inc.; ISBN: 0-937175-82-X) vagy Douglas Comer TCP/IP protokollról szóló könyveit.

Mindezek mellett még feltételezzük, hogy már beállítottuk a modem(ek)et és a rajtuk keresztüli bejelentkezéshez szükséges állományokat. Ha még nem készítettük volna fel erre a rendszerünket, akkor a Betárcsázós szolgáltatások ad részletes tájékoztatást a betárcsázós szolgáltatások beállításáról. A soros vonali eszközmeghajtóval kapcsolatban továbbá érdemes átolvasni a sio(4) oldalt, valamint a ttys(5), gettytab(5), getty(8) és init(8) oldalakat a bejelentkezések modemen keresztüli fogadásáról, illetve talán az stty(1) oldalt a soros port paramétereinek megfelelõ beállításáról (mint például a `clocal` a közvetlenül csatlakozó soros felületek esetében).

Gyors áttekintés

A FreeBSD SLIP szerverként általában a következõ módon üzemel: a SLIP felhasználó tárcsázza a FreeBSD-s SLIP szerverünket, majd bejelentkezik egy specális SLIP bejelentkezési azonosító használatával, amely a /usr/sbin/sliplogin shellt használja. A `sliplogin` program az /etc/sliphome/slip.hosts állományban megkeresi a speciális felhasználóhoz tartozó sort, és ha talál egy ilyet, akkor csatlakoztatja a soros vonalat egy rendelkezésre álló SLIP felületre, amelyen aztán a SLIP felültet beállításához lefuttatja az /etc/sliphome/slip.login shell szkriptet.

Példa SLIP szerveren keresztüli bejelentkezésre

Például, ha a SLIP felhasználó azonosítója `Shelmerg`, akkor az /etc/master.passwd állományban a hozzá tartozó bejegyzést nagyjából ilyen:

```
Shelmerg:password:1964:89::0:0:Guy Helmer -
SLIP:/usr/users/Shelmerg:/usr/sbin/sliplogin
```

Amikor Shelmerg bejelentkezik, a sliplogin az /etc/sliphome/slip.hosts állományban keresni fog egy felhasználó azonosítójához illeszkedõ sort. Például tegyük fel, hogy az /etc/sliphome/slip.hosts állományban szerepel egy ilyen sor:

```
Shelmerg        dc-slip sl-helmer       0xfffffc00        autocomp
```

A sliplogin ezt a sor fogja megtalálni, majd a soros vonalat a következõ elérhetõ SLIP felülethez kapcsolja, amelyen ezután végrehajtja az /etc/sliphome/slip.login szkriptet a következõ módon:

```
/etc/sliphome/slip.login 0 19200 Shelmerg dc-slip sl-helmer 0xfffffc00 autocomp
```

Ha minden jól megy, akkor az /etc/sliphome/slip.login kiad egy ifconfig parancsot azon a SLIP felületen, amelyre a sliplogin magát csatlakoztatta (amely a fenti példában a 0. SLIP felület volt, és amelyet meg is adtunk slip.login elsõ paramétereként), és így beállítja a helyi IP-címet (dc-slip), a távoli IP-címet (sl-helmer), a SLIP felülethez tartozó hálózati maszkot (0xfffffc00) valamint a további opciókat (autocomp). Ha valami rosszul sülne el, akkor a sliplogin ezekrõl általában nagyon jó minõségû, információdús üzeneteket készít, amelyeket a syslogd démon pedig a /var/log/messages állományba rögzít. (A syslogd(8) és syslog.conf(5) man oldalak és talán maga az /etc/syslog.conf segíthet kideríteni, hogy a syslogd jelenleg naplóz-e, és ha igen, akkor hova.)

A rendszermag beállítása

A FreeBSD alap (vagyis a GENERIC) rendszermagja támogatja a SLIP (sl(4)) használatát. Ha viszont saját rendszermagunk van, akkor elõfordulhat, hogy beállítások közé fel kell vennünk a következõ sort is:

```
device   sl
```

Alapértelmezés szerint a FreeBSD nem továbbít semmilyen csomagot. Amennyiben a FreeBSD SLIP szerverünket útválasztóként is mûködtetni akarjuk, úgy az /etc/rc.conf állományban a gateway_enable változót át kell állítanunk a YES értékre. Ennek hatására az újraindítás után is megmarad a csomagok továbbítása.

A változtatások azonnali életbeléptetéséhez adjuk ki root felhasználóként a következõ parancsot:

```
# /etc/rc.d/routing start
```

Ha a FreeBSD rendszermag beállítása során segítségre szorulnánk, akkor olvassuk el A FreeBSD rendszermag testreszabásaet.

A sliplogin beállítása

Ahogy arra már korábban is utaltunk, az /etc/sliphome könyvtárban három állomány felelõs a /usr/sbin/sliplogin beállításáért (lásd sliplogin(8)): a slip.hosts, amelyekben a SLIP felhasználókat és a hozzájuk tartozó IP-címeket adjuk meg; a slip.login, amely általában csak a SLIP felületet állítja

be; (az elhagyható) slip.logout, amely a soros vonal bontásakor a slip.login hatását igyekszik visszafordítani.

A slip.hosts beállítása

Az /etc/sliphome/slip.hosts soraiban whitespace karakterekkel tagoltan legalább négy elem szerepel:

- a SLIP felhasználó bejelentkezési azonosítója
- a SLIP kapcsolat helyi címe (a SLIP szerveréhez képest)
- a SLIP kapcsolat távoli címe
- hálózati maszk

A helyi és távoli címek lehetnek hálózati nevek is (amelyeket vagy az /etc/hosts, vagy pedig az /etc/nsswitch.conf állományban szereplõ beállítások alapján tudunk feloldani IP-címre), illetve a hálózati maszk is lehet egy olyan név, amelyet az /etc/networks fel tud oldani. A példaként bemutatott rendszerünkben az /etc/sliphome/slip.hosts állomány nagyjából így épül fel:

```
#
# login helyi-cím       távoli-cím      maszk            opc1    opc2
#                                               (normal,compress,noicmp)
#
Shelmerg  dc-slip       sl-helmerg      0xfffffc00      autocomp
```

A sorok végén az alábbi opciók közül egy vagy több szerepelhet:

- `normal` - a fejléceket nem tömörítjük
- `compress` - a fejlécek tömörítése
- `autocomp` - ha a távoli végpont engedi, akkor tömörítsük a fejléceket
- `noicmp` - az ICMP csomagok tiltása (így például a "ping" által generált csomagok is eldobódnak a sávszélesség felemésztese helyett)

A SLIP kapcsolathoz tartozó helyi és távoli címek megválasztása függ attól, hogy egy külön TCP/IP alhálózatot szentelünk-e neki, vagy a SLIP szerverünkön egy "ARP proxy"-t használunk (amely tulajdonképpen nem egy "valódi" ARP proxy, de ebben a szakaszban így fogunk rá hivatkozni). Ha nem vagyunk biztosak benne, hogy melyik módszert válasszuk vagy hogy miként osszuk ki az IP-címeket, akkor nézzünk utána ezekenek a SLIP használatával kapcsolatos elõfeltételek között megemlített könyvekben (Elõfeltételek) és/vagy konzultáljunk a hálózatunk karbantartójával.

Ha a SLIP klienseknek külön alhálózatokat osztunk ki, akkor a saját IP-címünkbõl kell létrehoznunk és kiadnunk ezeket. Ezután valószínûleg a SLIP szerverünkön keresztül még meg kell adnunk egy statikus útvonalat legközelebbi IP útválasztó felé.

Minden más esetben az "ARP proxy" módszert kell alkalmaznunk, ahol a SLIP kliensek IP-címeit a SLIP szerver Ethernet alhálózatából osztjuk ki, és ennek megfelelõen az /etc/sliphome/slip.login és /etc/sliphome/slip.logout szkripteket módosítanunk kell úgy, hogy az arp(8) segítségével képesek legyenek a SLIP szerver ARP táblázatában kezelni a "proxy ARP" bejegyzéseket.

A slip.login beállítása

Egy átlagos /etc/sliphome/slip.login állomány körülbelül ilyen:

```
#!/bin/sh -
#
#       @(#)slip.login  5.1 (Berkeley) 7/1/90

#
# Egy általános slip vonali bejelentkezési állomány. A sliplogin ezt az alábbi
# paraméterekkel hívja meg:
#      1        2         3        4          5         6     7-n
#   slipegys. ttyseb.  azonosító helyi-cím távoli-cím maszk egyéb-pmek.
#
/sbin/ifconfig sl$1 inet $4 $5 netmask $6
```

Ez a slip.login állomány az `ifconfig` segítségével pusztán beállítja a megfelelõ SLIP felülethez tartozó helyi, valamint távoli címet és a hálózati maszkot.

Ha ehelyett azonban az "ARP proxy" módszerét választottuk volna (tehát a SLIP kliensekenek nem akarunk egész alhálózatokat kiutalni), akkor az /etc/sliphome/slip.login állomány eképpen alakul:

```
#!/bin/sh -
#
#       @(#)slip.login  5.1 (Berkeley) 7/1/90

#
# Egy általános slip vonali bejelentkezési állomány. A sliplogin ezt az alábbi
# paraméterekkel hívja meg:
#      1        2         3        4          5         6     7-n
#   slipegys. ttyseb.  azonosító helyi-cím távoli-cím maszk egyéb-pmek.
#
/sbin/ifconfig sl$1 inet $4 $5 netmask $6
# A SLIP kliensre vonatkozó ARP kéréseket a mi Ethernet címünkkel
# válaszoljuk meg:
/usr/sbin/arp -s $5 00:11:22:33:44:55 pub
```

Láthatjuk, hogy az elõbbi slip.login állomány egy `arp -s $5 00:11:22:33:44:55 pub` paranccsal egészült ki, ami a SLIP szerver ARP táblázatában hoz létre egy ARP bejegyzést. Ez az ARP bejegyzés gondoskodik róla, hogy a SLIP szerver válaszoljon a saját Ethernetes MAC-címével, amikor egy másik IP csomópont a SLIP kliens IP-címe felõl érdeklõdik.

Amikor a fenti példából indulunk ki, a benne megadott MAC-címet (`00:11:22:33:44:55`) feltétlenül cseréljük a rendszerünk Ethernet kártyájának MAC-címével, mert különben az "ARP proxy" egyáltalán nem fog mûködni! A SLIP szerverünk MAC-címét a `netstat -i` paranccsal deríthetjük ki, amelynek a kimenetében a második sor valahogy így néz ki:

```
ed0   1500  <Link>0.2.c1.28.5f.4a         191923    0   129457     0   116
```

Ebbõl derül ki, hogy az adott rendszer valódi MAC-címe a `00:02:c1:28:5f:4a` - az arp(8) számára azonban a `netstat -i` kimenetében szereplõ pontokat kettõspontokra kell cserélni, és a tagokat ki kell egészíteni kétkarakteres hexadecimális számokká. Az arp(8) man oldalán tudhatunk meg ennek részleteirõl többet.

Amikor létrehozzuk az /etc/sliphome/slip.login és /etc/sliphome/slip.logout állományokat, akkor ne felejtsük el hozzájuk beállítani a "végrehajtást" engedélyezõ bitet sem (tehát ilyenkor mindig adjuk ki a `chmod 755 /etc/sliphome/slip.login /etc/sliphome/slip.logout` parancsokat is), különben a `sliplogin` ezeket nem tudja majd elindítani.

A slip.logout beállítása

Az /etc/sliphome/slip.logout állományra nincs feltétlenül szükségünk (hacsak nem egy "ARP proxy"-t akarunk csinálni), de ha valamiért mégis el akarjuk készíteni, akkor ehhez a következõ alapvetõ slip.logout szkript használható:

```
#!/bin/sh -
#
#       slip.logout

#
# Egy logout állomány a slip vonalhoz. A sliplogin ezt a szkriptet a
# következõ paraméterekkel hívja:
#      1        2         3        4          5         6     7-n
#   slipegys. ttyseb.   login helyi-cím  távoli-cím   maszk opc-pmek.
#
/sbin/ifconfig sl$1 down
```

Ha az "ARP proxy" módszert használjuk, és az /etc/sliphome/slip.logout felhasználásával akarjuk a SLIP klienshez tartozó ARP bejegyzést törölni, akkor ebbõl induljunk ki:

```
#!/bin/sh -
#
#       @(#)slip.logout

#
# Egy logout állomány a slip vonalhoz. A sliplogin ezt a szkriptet a
# következõ paraméterekkel hívja:
#      1        2         3        4          5         6     7-n
#   slipegys. ttyseb.   login helyi-cím  távoli-cím   maszk opc-pmek.
#
sbin/ifconfig sl$1 down
# Ne válaszoljunk többet a SLIP kliensre vonatkozó ARP kérésekre
```

```
/usr/sbin/arp -d $5
```

Az `arp -d $5` parancs eltávolítja az "ARP proxy" mûködéséhez bejegyzést, amelyet még a slip.login szkripttel vettünk fel a SLIP kliens bejelentkezésekor.

Talán felesleges ismételgetésnek tûnhet: az /etc/sliphome/slip.logout állománynak létrehozása után állítsuk be a végrehajtásra szóló bitet (vagyis adjuk ki a `chmod 755 /etc/sliphome/slip.logout` parancsot).

Az útválasztással kapcsolatos megfontolások

Ha a hálózatunk többi része (lényegében az internet) és a SLIP klienseink között nem az "ARP proxy" módszerrel közvetítjük a csomagokat, akkor a legközelebbi alapértelmezett átjárókhoz minden bizonnyal fel kell vennünk statikus útvonalakat, így a SLIP kliensek alhálózatai a SLIP szerverünkön keresztül ki tudnak jutni.

Statikus útvonalak

A legközelebbi alapértelmezett átjárók felé nem minden esetben könnyû felvenni statikus útvonalakat (vagy egyes esetekben pedig egyenesen lehetetlen, mivel nincsenek meg hozzá a jogaink). Ha az intézményünkön belül több átjáró is megtalálható, akkor bizonyos útválasztók, például a Cisco és Proteon gyártmányúak esetében nem csak a SLIP alhálózatok felé kell beállítanunk statikus útvonalakat, hanem azt is meg kell mondanunk, hogy ezekrõl milyen más útválasztók is tudjanak. Pontosan emiatt a statikus útválasztás beüzemeléséhez szükségünk lesz egy kis utánajárásra és próbálgatásra.

Chapter 28. Elektronikus levelezés

28.1. Áttekintés

Az "elektronikus levelezés", más néven e-mail, a kommunikáció egyik legjobban elterjedt formája. Ebben a fejezetben bemutatjuk, hogyan futtassunk FreeBSD-n levelezõ szervert, illetve hogyan küldjünk és fogadjunk e-maileket a FreeBSD használatával. Ez azonban semmiképpen sem tekinthetõ egy teljes referenciának és tulajdonképpen számos fontos tényezõrõl szót sem ejtünk. A témára úgy kaphatunk egy sokkal átfogóbb rálátást, ha a Irodalomjegyzékben felsorolt remek könyveket is elolvassuk.

A fejezet elolvasása során megismerjük:

- milyen szoftverkomponensek játszanak szerepet az elektronikus levelek küldésében és fogadásában;
- FreeBSD-ben hol találhatóak a sendmail konfigurációs állományai;
- mi a különbség a helyi és távoli postaládák között;
- hogyan akadályozzuk meg, hogy a levelezõ szerverünk a kéretlen levélszemetet továbbítson;
- rendszerünkön hogyan telepítsünk és állítsunk be más levelezõ szervereket a sendmail helyett;
- hogyan oldjuk meg a levelezõ szerverekkel kapcsolatban felmerülõ általános problémákat;
- hogyan használjuk az SMTP protokollt az UUCP protokollal;
- hogyan kell rendszerüket csak levélküldésre beállítani;
- hogyan levelezzünk betárcsázós kapcsolattal;
- hogyan növeljük rendszerünk védelmét az SMTP hitelesítésének engedélyezésével;
- hogyan telepítsünk és használjunk a levelek küldésére és fogadására például a mutthoz hasonló levelezõ klienseket;
- hogyan töltsük le leveleinket egy távoli POP vagy IMAP szerverrõl;
- hogyan alkalmazzunk automatikusan adott szabályokat vagy szûrõket az érkezõ levelekre.

A fejezet elolvasása elõtt ajánlott:

- az internet-csatlakozásunk megfelelõ beállítása (Egyéb haladó hálózati témák);
- a névfeloldás beállítása (Hálózati szerverek);
- a külsõ fejlesztésû alkalmazások telepítésének ismerete (Alkalmazások telepítése. csomagok és portok).

28.2. Az elektronikus levelezés használata

Öt fontosabb részre bonthatjuk a levelezést. Ezek: a felhasználói program (mail user agent), a levélküldõ démon (mail transfer agent), a névfeloldás, a helyi vagy távoli postaláda és természetesen maga a levelezõ szerver (mail host).

28.2.1. A felhasználói program

Ide soroljuk a különbözõ parancssoros programokat, mint például a mutt, pine, elm és `mail`, valamint a különféle grafikus alkalmazásokat, mint például a balsa és az xfmail, csak hogy felsoroljuk néhány újabb, egy webböngészõhöz hasonlóan "kifinomult" eszközt is. Ezek a programok egyszerûen átküldik az elektronikus levelekkel kapcsolatos tranzakciókat a helyi "levelezõ szervernek" vagy meghívják valamelyik levélküldõ démont, esetleg közvetlenül a TCP protokollon keresztül kézbesítenek.

28.2.2. A levélküldõ démon

A FreeBSD alapból a sendmail nevû programot ajánlja fel erre a célra, de támogat más levelezõ szervereket is, ezek közül meg is említünk néhányat ízelítõként:

- exim
- postfix
- qmail

Ez a démon általában két feladatot lát el - a beérkezõ levelek fogadásáért és a kimenõ levelek elküldéséért felelõs. *Nem* tartozik azonban a feladatai közé, hogy a POP vagy IMAP protokollokhoz hasonlóan olvashatóvá tegye a leveleinket, illetve csatlakozni engedjen a helyi mbox vagy Maildir formátumú postaládáinkhoz. Ezekhez a mûveletekhez egy külön démon szükségeltetik.

A sendmail régebbi változatai tartalmaznak olyan komoly biztonsági hibákat, amelyek kihasználásával az illetéktelen behatolók helyi és/vagy távoli hozzáférést tudnak szerezni a gépünkön. Az ilyen jellegû problémák elkerülése érdekében igyekezzünk mindig a legfrissebb verzióját használni. Vagy a FreeBSD Portgyûjteményébõl telepítsünk fel egy másik levélküldõ démont.

28.2.3. Az elektronikus levelek és a névfeloldás

A névfeloldás (Domain Name System, DNS) és a hozzá tartozó `named` démon nagy szerepet játszik az elektronikus levelek továbbításában. A démon a leveleket úgy küldi át az egyik géprõl a másikra, hogy a névfeloldáson keresztül megkeresi azt a távoli gépet, amelynek a leveleket címezték. Ez a folyamat szintén végbemegy, amikor egy távoli géprõl levelet küldenek a mi szerverünkre.

A DNS valósítja meg a hálózati nevek és az IP-címek összerendelését valamint ez tárolja el a levélküldésre vonatkozó információkat is, amelyeket MX rekordoknak hívnak. Az MX (Mail eXchanger, "levélváltó") rekord adja meg azt a gépet vagy azokat a gépeket, amelyek az adott névtartományban fogadják a leveleket. Ha a hálózati nevünkhöz vagy tartományunkhoz nem tartozik MX rekord, akkor a levél közvetlenül a gépünkre vándorol feltéve, hogy rendelkezik olyan A rekorddal, amely összerendeli a gépünk nevét az IP-címével.

A host(1) parancs használatával az alábbi példához hasonlóan tetszõleges tartomány MX rekordját meg tudjuk nézni:

```
% host -t mx
FreeBSD.org FreeBSD.org mail is handled (pri=10) by
```

```
mx1.FreeBSD.org
```

28.2.4. Az elektronikus levelek fogadása

A tartományunkhoz tartozó leveleket fogadását a levelezõ szerver végzi. Összegyûjti a tartományunkba küldött összes levelet és ezeket a beállításainktól függõen vagy mbox (a levelek tárolásának alapértelmezett módja) vagy pedig Maildir formátumban eltárolja. Ahogy eltárolt egy levelet, úgy helyben egybõl el is tudjuk olvasni például a mail(1) vagy a mutt használatával, illetve távolról a POP vagy IMAP és a hasonló protokollokkal tudjuk elérni és begyûjteni. Ezért tehát ha csak a helyi gépen kívánjuk olvasni a leveleinket, akkor ahhoz egyáltalán nem kell POP vagy IMAP szervert telepítenünk.

Távoli postaládák elérése a POP és IMAP használatával

A távoli postaládák eléréséhez tudnunk kell csatlakozni egy POP vagy IMAP szerverhez. Ezeken a protokollokon keresztül tudják a felhasználók minden különösebb nehézség nélkül elérni távolról a helyi postaládáikat. Noha a POP és az IMAP segítségével egyaránt el tudjuk így érni a postaládákat, az IMAP használatának mégis több elõnye van, íme néhány közülük:

- Az IMAP a levelek leszedése mellett tárolni is képes a távoli szerveren.
- Az IMAP támogat párhuzamos lekéréseket.
- Az IMAP hihetetlenül hasznos tud lenni lassabb összeköttetések esetében, mivel lehetõvé teszi a felhasználók számára, hogy csak az üzenetek vázát töltsék le és ne az egészet. Továbbá a szerver és a kliens közti adatmozgás csökkentése érdekében képes bizonyos feladatokat a szerveren elvégezni, például keresni.

Egy POP vagy IMAP szerver telepítéséhez az alábbi lépések megtétele szükséges:

1. Válasszuk ki az igényeinket legjobban kielégítõ IMAP vagy POP szervert. A következõ POP és IMAP szerverek eléggé elterjedtek és egyben remek példák:
   - qpopper
   - teapop
   - imap-uw
   - courier-imap
2. A Portgyûjteménybõl telepítsük fel a kiválasztott POP vagy IMAP démont.
3. Ha szükséges, akkor a POP vagy IMAP szerver betöltéséhez írjuk át az /etc/inetd.conf állományt.

Meg kell említenünk, hogy mind a POP és az IMAP az összes információt, tehát belértve a felhasználók neveit és jelszavait titkosítatlan formában továbbítja. Ez azt jelenti, hogy ha ezeket a protokollokat biztonságos módon szeretnénk elérni, akkor az ssh(1) használatával hozzunk létre hozzá egy tunnelt és azon keresztül használjuk. Errõl részletesebben a Tunnelezés SSH-valban olvashatunk.

A helyi postaládák elérése

A helyi postaládákat a szerveren levõ levelezõ kliensek közvetlen használatával érhetjük el. Ilyen alkalmazások például a mutt vagy a mail(1).

28.2.5. A levelezõ szerver

A levelezõ szerver az a szerver, amely a gépünk vagy akár az egész hálózatunk irányába érkezõ levelek fogadásáért és elküldéséért felelõs.

28.3. A sendmail beállítása

A sendmail(8) a FreeBSD alapértelmezett levéltovábbító ügynöke (Mail Transfer Agent, MTA). A sendmail feladata fogadni a levelezõ kliensektõl (Mail User Agent, MUA) érkezõ leveleket és kézbesíteni azokat a konfigurációs állományában megadott megfelelõ levelezõnek. A sendmail hálózati kapcsolatokat is fogad, képes a helyi postaládákba vagy akár más programoknak is leveleket továbbítani.

A sendmail a következõ állományban tárolja beállításait:

Állomány	Szerep
/etc/mail/access	A sendmail által engedélyezett hozzáféréseket tároló adatbázis
/etc/mail/aliases	A postaládák álnevei
/etc/mail/local-host-names	Azon nevek felsorolása, amelyek számára a sendmail leveleket fogad
/etc/mail/mailer.conf	A levelezõ programok beállításai
/etc/mail/mailertable	A levelezõ programok kézbesítési táblázata
/etc/mail/sendmail.cf	A sendmail központi beállításait tároló állomány
/etc/mail/virtusertable	Virtuális felhasználók és tartományok táblázatai

28.3.1. /etc/mail/access

Az engedélyezett hozzáféréseket tároló adatbázis tartalmazza milyen hálózati neveken vagy IP-címeken lehet elérni a helyi levelezõ szervert és azok milyen típusú hozzáférést kapnak. A gépek az `OK` (rendben), `REJECT` (visszautasít), `RELAY` (továbbítás) beállításokat alkalmazhatjuk, vagy egyszerûen meghívhatjuk hozzájuk a sendmail hibakezelõ rutinját egy adott kézbesítési hibával. Ha egy gépet az `OK` beállítással veszük fel a listára, ami egyébként alapértelmezés, akkor ez a gép levelet tud küldeni egészen addig, amíg a végsõ cél a helyi gép marad. A `REJECT` beállítással felsorolt gépek számára semmiféle levelezés nem engedélyezett. Ha pedig egy gép mellett a `RELAY` beállítás jelenik meg, akkor a szerveren keresztül tetszõleges címre küldhet.

Példa 30. A sendmail elérését szabályozó adatbázis beállítása

```
cyberspammer.com                550 Nem szeretjuk a spammereket
```

```
FREE.STEALTH.MAILER@            550 Nem szeretjuk a spammereket
another.source.of.spam          REJECT
okay.cyberspammer.com           OK
128.32                          RELAY
```

Ebben a példában öt bejegyzést láthatunk. A táblázat bal felének valamelyik sorára illeszkedõ küldõkre a táblázatban a sor jobb felén megjelenõ cselekvés érvényesül. Az elsõ két sorban a sendmail hibakezelõ rutinjának adunk át hibakódokat. A hozzá tartozó üzenet akkor fog megjelenni a távoli gépen, amikor a tõle érkezõ levél illeszkedik a bal oldali szabályra. Az ezeket követõ bejegyzésben visszalökünk minden olyan levelet, amely az internetrõl egy adott számítógéptõl érkezik, például az `another.source.of.spam` címrõl. A következõ bejegyzésben az `okay.cyberspammer.com` címrõl elfogadjuk a kapcsolódást, ami viszont sokkal pontosabb megjelölés a fentebb szereplõ `cyberspammer.com` sornál. A pontosabban kifejtett nevek felülbírálják a kevésbé pontosan megnevezetteket. Végül az utolsó bejegyzésben engedélyezzük a levelek továbbküldését minden olyan gép számára, amelynek címe a `128.32` elõtaggal kezdõdik. Ezek tehát képesek ezen a levelezõ szerveren keresztül bárhova leveleket küldeni.

Az állomány módosítása után az adatbázis frissítéséhez mindig le kell futtatnunk egy `make` parancsot az /etc/mail/ könyvtárban.

28.3.2. /etc/mail/aliases

Az álneveket tartalmazó adatbázis virtuális postaládákat sorol fel, amelyek más felhasználókra, állományokra, programokra vagy további álnevekre vonatkozhatnak. Íme néhány példa az /etc/mail/aliases állományban szereplõ bejegyzésekre:

Példa 31. Virtuális postaládák

```
root: localuser
ftp-bugs: joe,eric,paul
bit.bucket:  /dev/null
procmail: "|/usr/local/bin/procmail"
```

A formai szabályok egyszerûek: a kettõspont bal oldalára kell írni azt a postaládát, amely a jobb oldalán levõ célokra bomlik. A példa elsõ sorában egyszerûen megfeleltetjük a `root` postaládáját a `localuser` postaládájának, majd ezt a nevet keressük az álnevek adatbázisában. Ha nem találunk már rá illeszkedést, akkor az üzenetet a `localuser` nevû helyi felhasználónak továbbítjuk. A következõ sorban címek listáját láthatjuk. Ennek megfelelõen a `ftp-bugs` postaláda címére küldött levelek három további helyi postaládára mennek tovább: ezek név szerint a `joe`, `eric` és `paul` felhasználók postaládái. Itt a távoli postaládák felhasználó@példa.hu alakban adhatóak meg. A következõ sor az állományok használatát példázza, ahol konkrétan a /dev/null állományba irányítjuk át az adott címre érkezõ leveleket. Az utolsó sorban pedig a programok használatára láthatunk példát, ahol ebben az esetben a levél egy UNIX®-os csövön keresztül a /usr/local/bin/procmail szabványos bemenetére kerül.

Ha megváltoztatjuk ezt az állományt, akkor utána az adatbázis frissítéséhez ne felejtsük el

meghívni a `make` parancsot az /etc/mail/ könyvtárban.

28.3.3. /etc/mail/local-host-names

Ebben az állományban adhatjuk meg, hogy a sendmail(8) milyen hálózati neveket fogadjon el helyi hálózati névként. Ide kell raknunk azokat a tartományokat vagy címeket, amelyektõl a sendmail leveleket fogad el. Például, ha a levelezõ szerver az `minta.com` tartományból és a `level.minta.com` címrõl fogad el leveleket, akkor a local-host-names valahogy így fog kinézni:

```
minta.com
level.minta.com
```

Az állomány módosításakor a sendmail(8) programot újra kell indítani a változások érvényesítéséhez.

28.3.4. /etc/mail/sendmail.cf

Ahogy a sendmail központi konfigurációs állománya, a sendmail.cf irányítja a sendmail átfogó viselkedését, beleértve mindent az e-mail címek átírásától kezdve a távoli szervereknek küldött elutasító üzenetek küldéséig. Mivel ennyire sokfajta szerepet tölt be egyszerre, ezért ez a konfigurációs állomány meglehetõsen összetett és a részletezése meghaladná ennek a leírásnak a határait. Szerencsére az átlagos levelezõ szerverek esetében ezt az állományt nagyon ritkán kell módosítani.

A sendmail központi konfigurációs állománya a sendmail lehetõségeit és viselkedését meghatározó m4(1) makrókból építhetõ fel. A pontosabb részleteket a /usr/src/contrib/sendmail/cf/README állományban találjuk meg.

Az állomány megváltoztatása után a módosítások érvényesítéséhez újra kell indítani a sendmail programot.

28.3.5. /etc/mail/virtusertable

A virtusertable állomány képezi le a virtuális tartományokhoz tartozó címeket valódi postaládák címeire. Ezek a postaládák lehetnek helyiek, távoliak, az /etc/mail/aliases állományban megadott álnevek vagy állományok.

Példa 32. Példa a virtuális tartományok leképezésére

```
root@minta.com               root
postmaster@minta.com         postmaster@noc.minta.net
@minta.com                   joe
```

A fenti példában megadtunk egy leképezést a `minta.com` tartományhoz. Ez az állomány úgy dolgozódik fel, hogy fentrõl lefelé illesztõdnek a címek, egészen az elsõ egyezésig. Az elsõ bejegyzés szerint a root@minta.com a helyi `root` felhasználó postaládájára képzõdik le. A következõ bejegyzés

szerint a postmaster@minta.com a noc.minta.net címen található postmaster nevû felhasználó postaládájára képzõdik le. Végezetül, ha a minta.com címrõl eddig még semmi sem illeszkedett volna, akkor az utolsó leképezés veszi át, amely az minta.com tartományon belül az összes többi címre küldött levelet a helyi joe nevû felhasználó postaládájára képezi le.

28.4. A levéltovábbító ügynök megváltoztatása

Ahogy arról már korábban szó esett, a FreeBSD alapból tartalmazza a sendmail programot mint levéltovábbító ügynököt (MTA, Mail Transfer Agent). Ennélfogva alapértelmezés szerint ez a felelõs a kimenõ és beérkezõ levelek kezeléséért.

Számtalan okból eredõen egyes rendszergazdák azonban mégis szeretnék lecserélni a rendszerükhöz tartozó levéltovábbítót. Ennek oka lehet egyszerûen csak annyi, hogy ki akarunk próbálni egy másik programot vagy éppen egy olyan eszközre van szükségünk, amely kizárólag csak máshol található meg. Szerencsére a FreeBSD megkönnyíti ezt a váltást.

28.4.1. Az új levéltovábbító telepítése

A levéltovábbítók széles köre elérhetõ. A FreeBSD Portgyûjteményébõl elindulva sok ilyen programot találhatunk. Természetesen teljesen mindegy, hogy melyik levéltovábbítót választjuk egészen addig, amíg képesek vagyunk FreeBSD alatt rendesen futtatni.

Kezdjük tehát az új levéltovábbító telepítésével. Miután sikerült telepíteni, lehetõségünk van eldönteni, hogy valóban eleget tesz-e az igényeinknek, sõt az új szoftvert még az elõtt be tudjuk állítani, hogy átvenné a sendmail helyét. Vigyázzunk azonban, hogy az új szoftver telepítésekor ne írjon felül olyan rendszerszintû binárisokat, mint például a /usr/bin/sendmail. Másrészt az új levelezõ szoftvert szolgálatba helyezése elõtt mindenképpen fontos megfelelõen beállítanunk.

A kiválasztott levéltovábbító beállításával kapcsolatban olvassuk el a hozzá tartozó dokumentációt.

28.4.2. A sendmail letiltása

Amikor letiltjuk a sendmail kimenõ levél szolgáltatását, soha ne felejtsük el pótolni valamilyen más levelezõ rendszerrel. Ha nem így cselekszünk, akkor például a periodic(8) és a hozzá hasonló programok nem lesznek képesek a tõlük megszokott módon e-mailben elküldeni a futásuk eredményét. A rendszer bizonyos részei ráadásul egy mûködõ, sendmail-kompatibilis rendszert feltételeznek. Ha letiltása után az alkalmazások továbbra is a sendmail segítségével próbálnak levelet küldeni, akkor ez a levél a sendmail inaktív sorába kerülhet, ahonnan soha nem kerül kézbesítésre.

A sendmail teljes leállításához, beleértve a kimenõ levelekhez tartozó szolgáltatást is, a következõket kell megadni az /etc/rc.conf állományban:

```
sendmail_enable="NO"
sendmail_submit_enable="NO"
sendmail_outbound_enable="NO"
```

```
sendmail_msp_queue_enable="NO"
```

Ha csak a sendmail beérkezõ levelekre vonatkozó szolgáltatását akarjuk tiltani, akkor ahhoz az /etc/rc.conf állományban a következõt állítsuk be:

```
sendmail_enable="NO"
```

A sendmail indításával kapcsolatos további beállításokat az rc.sendmail(8) man oldalon találjuk.

28.4.3. Az új levéltovábbító elindítása a rendszerrel együtt

Az új levéltovábbítót úgy tudjuk elindítani a rendszerrel együtt, ha az /etc/rc.conf állományba felvesszük a következõ sort, például a postfix esetében:

```
# echo 'postfix_enable="YES"' >> /etc/rc.conf
```

Az új levéltovábbító így most már magától el fog indulni a rendszer indításakor.

28.4.4. A sendmail mint a rendszer alapértelmezett levelezõ eszközének lecserélése

A sendmail annyira elterjedt szabványos szoftver a UNIX® rendszereken, hogy egyes szoftverek egyszerûen feltételezik a jelenlétét. Emiatt sok levéltovábbítóhoz tartozik egy sendmail kompatibilis parancssoros felület is, amellyel igyekeznek megkönnyíteni a sendmail"gyors" lecserélését.

Ennek következtében tehát, ha egy másik levelezõ eszközt használunk, akkor valamilyen módon meg kell bizonyosodnunk róla, hogy a szabványos sendmail binárisok, mint például a /usr/bin/sendmail, valóban a kiválasztott levéltovábbítot fogják aktiválni. Szerencsére a FreeBSD pontosan emiatt tartalmaz egy mailwrapper(8) nevû rendszert.

Amikor a sendmail telepítése szerint mûködik, valami hasonlót fogunk találni az /etc/mail/mailer.conf állományban:

```
sendmail    /usr/libexec/sendmail/sendmail
send-mail   /usr/libexec/sendmail/sendmail
mailq       /usr/libexec/sendmail/sendmail
newaliases  /usr/libexec/sendmail/sendmail
hoststat    /usr/libexec/sendmail/sendmail
purgestat   /usr/libexec/sendmail/sendmail
```

Ez azt jelenti, hogy amikor az itt felsorolt általános parancsok közül lefuttatjuk valamelyiket (például magát a sendmail parancsot), akkor a rendszer magától meghívja a sendmail néven szereplõ wrapper programot, amely pedig a mailer.conf alapján kideríti, hogy az adott esetben a /usr/libexec/sendmail/sendmail hívására van szükség. Ez a rendszer megkönnyíti az alapértelmezett sendmail funkciók helyében lefuttatandó binárisok átállítását.

Így tehát, ha a /usr/local/supermailer/bin/sendmail-compat állományt akarjuk futtatni a megszokott sendmail helyében, akkor az /etc/mail/mailer.conf állományt a következõképpen kell módosítanunk:

```
sendmail    /usr/local/kedvenclevelezõ/bin/sendmail-compat
send-mail   /usr/local/kedvenclevelezõ/bin/sendmail-compat
mailq       /usr/local/kedvenclevelezõ/bin/mailq-compat
newaliases  /usr/local/kedvenclevelezõ/bin/newaliases-compat
hoststat    /usr/local/kedvenclevelezõ/bin/hoststat-compat
purgestat   /usr/local/kedvenclevelezõ/bin/purgestat-compat
```

28.4.5. A mûvelet befejezése

Ahogy a céljainknak megfelelõen mindent beállítottunk, akkor vagy egyszerûen leállítjuk a sendmail neve alatt futó programokat és helyettük elindítjuk az új szoftverhez tartozókat, vagy csak újraindítjuk a gépet. Az újraindítással mellesleg ellenõrizhetjük azt is, hogy jól állítottuk be a rendszerünket és az új levélküldõ tényleg elindul a rendszerünkkel együtt.

28.5. A hibák elhárítása

28.5.1. Miért kell teljes hálózati neveket megadni a gépemen?

Elõfordulhat, hogy a hivatkozni kívánt gép valójában egy másik tartományban szerepel. Például, ha az `ize.mize.edu` gépen vagyunk és a `vagyis` nevû gépet akarjunk innen elérni a `mize.edu` tartományban, akkor a teljes hálózati nevével, vagyis a `vagyis.mize.edu` néven kell rá hivatkoznunk, nem pedig egyszerûen csak `vagyis` néven.

Régebben egyébként ezt a BSD-típusú BIND névfeloldók megengedték. A FreeBSD jelenlegi változatai azonban már olyan BIND verziót tartalmaznak, amelyek alapértelmezés szerint már nem engedik a tartományunkon kívüli relatív nevek használatát. Tehát a `vagyis` vagy a `vagyis.ize.mize.edu` gép lesz, vagy a legfelsõ, gyökér tartományban keresi a rendszer.

Ez eltér a korábbi viselkedéstõl, ahol a keresés folytatódott a `vagyis.mize.edu` és `vagyis.edu` tartományokban is. Az RFC 1535 elolvasásából ki fog derülni, hogy miért nem vált be ez a gyakorlat és hogy miért tekinthetõ még akár biztonsági résnek is.

Ezt a problémát egyébként megoldhatjuk annyival, hogy az /etc/resolv.conf állományba a

```
search ize.mize.edu mize.edu
```

sor helyett a

```
domain ize.mize.edu
```

sort írjuk be. Arra viszont ügyeljünk, hogy a keresési rend ne lépje át a "helyi és nyilvános adminisztráció között meghúzódó határt", ahogy azt az RFC 1535 nevezi.

28.5.2. A sendmail szerint a levél a saját farkába harap

Ezt a sendmail gyakran ismértelt kérdései között a következõképpen válaszolták meg:

```
A következõ hibaüzenetet kapom:

553 MX list for taromány.net points back to felé.tartomány.net
554 felhasználó@tartomány.net... Local configuration error

Hogyan oldható meg ez a probléma?

Azt kértük, hogy a tartományba (például tartomány.net) küldött levél
az MXMX rekord rekord felhasználásával egy adott gépre legyen átirányítva
(ebben az esetben ez a felé.tartomány.net), de a továbbítást végzõ gép
nem ismeri fel magát a tartomány.net címen. Vegyük fel a tartomány.net
tartományt az /etc/mail/local-host-names állományba [melyet a 8.10 elõtti
verziókban /etc/sendmail.cw állománynak hívnak] (ha a
FEATURE(use_cw_file) beállítást használjuk) vagy tegyük hozzá a
Cw tartomány.net sort az /etc/mail/sendmail.cf
állományhoz.
```

A sendmail GYIK a http://www.sendmail.org/faq/ címen található meg (angolul) és mindenképpen javasolt elolvasni, ha "fel szeretnénk piszkálni" a levelezõ rendszerünk beállításait.

28.5.3. Hogyan tudok levelezõ szervert futtatni egy betárcsázós PPP kapcsolat esetében?

Egy helyi hálózaton levõ FreeBSD-s gépet akarunk tehát az internethez kapcsolni. Ez a FreeBSD-s gép lesz a helyi hálózat leveleket továbbító átjárója. A PPP kapcsolat nem dedikált.

Legalább két módon meg tudjuk oldani. Az egyik módszer szerint az UUCP használatára lesz szükségünk.

A másik módszer szerint szereznünk kell egy éjjel-nappal üzemelõ internetes szervert, amely majd szolgáltatja a másodlagos MX rekordot a tartományunkhoz. Például, ha a cégünk tartománya a `cég.hu` és az internet-szolgáltatónk a `szolgáltató.net` névre beállította a tartományunkhoz a másodlagos MX rekordokat:

```
cég.hu.          MX       10      cég.hu.
                 MX       20      szolgáltató.net.
```

Végsõ címzettként csak egy gépet kell megadni (az /etc/mail/sendmail.cf állományba a `cég.hu` címhez tegyük hozzá a `Cw cég.hu` sort).

Amikor a leveleket küldeni akaró `sendmail` megpróbál kézbesíteni, elõször hozzánk (`cég.hu`) próbál csatlakozni a modemes összeköttetésen keresztül. Ez valószínûleg idõtúllépéssel befejezõdik, mivel nem vagyunk fenn minden pillanatban a neten. A sendmail ekkor automatikusan a másodlagos MX rekord által megadott címre küldi a levelet, tehát a szolgáltatónkhoz (`szolgáltató.net`). Ez a

másodlagos MX cím próbálja majd idõlegesen elérni a gépünket és kézbesíteni a leveleket az elsõdleges MX rekord által megadott gépre (cég.hu).

A bejelentkezéskor ezért egy hasonló szkriptet kell lefuttatnunk:

```
#!/bin/sh
# Tegyük a /usr/local/bin/pppmyisp állományba:
( sleep 60 ; /usr/sbin/sendmail -q ) &
/usr/sbin/ppp -direct pppmyisp
```

Ha készítünk egy külön bejelentkezõ szkriptet a felhasználók számára, akkor a `sendmail -qRcég.hu` parancsot is használhatjuk a fenti szkript helyett. Ezzel a `cég.hu` sorában található összes levél azonnal feldolgozásra kerül.

A helyzetet így lehetne még jobban pontosítani:

Az alábbi üzenet a FreeBSD Internet service provider's levelezési lista archívumából származik.

```
> we provide the secondary MX for a customer. The customer connects to
> our services several times a day automatically to get the mails to
> his primary MX (We do not call his site when a mail for his domains
> arrived). Our sendmail sends the mailqueue every 30 minutes. At the
> moment he has to stay 30 minutes online to be sure that all mail is
> gone to the primary MX.
>
> Is there a command that would initiate sendmail to send all the mails
> now? The user has not root-privileges on our machine of course.

In the privacy flags section of sendmail.cf, there is a
definition Opgoaway,restrictqrun

Remove restrictqrun to allow non-root users to start the queue processing.
You might also like to rearrange the MXs. We are the 1st MX for our
customers like this, and we have defined:

# If we are the best MX for a host, try directly instead of generating
# local config error.
OwTrue

That way a remote site will deliver straight to you, without trying
the customer connection.  You then send to your customer.  Only works for
hosts, so you need to get your customer to name their mail
machine customer.com as well as
hostname.customer.com in the DNS.  Just put an A record in
the DNS for customer.com.
```

Az idézet fordítása:

```
> Másodlagos MX rekordot biztosítunk az ügyfeleinknek.  Az ügyfelek ezután
automatikusan
> csatlakoznak naponta akár többször is a szolgáltatásunkhoz és leszedik az elsõdleges
MX
> rekordhoz tartozó leveleket.  (Nem szólunk neki, amikor a tartományához levél
> érkezik.) A sendmail programunk minden 30 percben elküldi a sorban felhalmozódott
> leveleket.  Tehát jelen pillanatban legalább 30 percig fenn kell lennie az
ügyfélnek, hogy
> rendben megkapja az elsõdlegesre MX rekordra.
>
> Létezik valamilyen parancs a sendmail programhoz, amellyel azonnal lekérhetjük az
összes
> levelünket?  A felhasználómnak természetesen nincsenek rendszergazdai jogosultságai
az adott
> gépen.

A sendmail.cf privacy flags beállításai között van egy definíció, az
Opgoaway,restrictqrun.

Vegyük ki innen a restrictqrun beállítást, amivel a nem root felhasználók is
megindíthatják a
sor feldolgozását.  Valószínûleg az MX-ek átrendezésére is szükség lesz.  Mi vagyunk
az elsõ MX
az ilyen típusú ügyfelek számára, és ezt adtuk meg:

# Ha mi vagyunk a legjobb MX a levél számára, akkor ne generáljunk
# helyi beállítási hibát, hanem próbálkozzunk közvetlenül.
OwTrue

Ezzel már a távoli gép közvetlenül nekünk küld anélkül, hogy próbálkozna az ügyfél
kapcsolatával.
Ezt majd továbbküldjünk az ügyfélnek.  Ez csak hálózati nevek esetében mûködik, tehát
az ügyfelünknek
el kell neveznie a leveleket fogadó gépét customer.com-nak, valamint a fel kell venni
a
hostname.customer.com címet is a DNS-be.  Ehhez egyszerûen csak elegendõ egy A
rekordot
betenni a customer.com-hoz.
```

28.5.4. Miért kapok folyton Relaying Denied hibát, amikor más gépekrõl küldök levelet?

A FreeBSD alapértelmezett telepítése során a sendmail úgy állítódik be, hogy csak arról a géprõl küldhetünk vele levelet, ahol fut. Például, ha POP szerver is elérhetõ, akkor a felhasználók meg tudják nézni a leveleiket az iskolából, munkából vagy bármilyen más távoli helyrõl, de leveleket onnan továbbra sem tudnak küldeni. Általában pár pillanattal a próbálkozás után a MAILER-DAEMON küldeni fog egy `5.7 Relaying Denied` (`5.7 A továbbítás nem engedélyezett`) üzenetet.

Több lehetõségünk is van ennek megkerülésére. Az a legegyszerûbb módszer, ha az internet-

szolgáltatónk címét felvesszük az /etc/mail/relay-domains állományba. Például így:

```
# echo "az.internet.szolgáltató.net" > /etc/mail/relay-domains
```

Az állomány létrehozása vagy módosítása után újra kell indítanunk a sendmail programot. Ez remekül mûködik abban az esetben, ha rendszergazdák vagyunk és nem akarunk a helyi géprõl levelet küldeni, vagy egy másik gépen vagy akár másik internet-szolgáltatóval akarunk valamilyen kattingatós levelezõ programot használni. Olyankor is nagyon hasznos lehet, amikor csak egy vagy két e-mail hozzáférést állítottunk be. Ha egyszerre több címet is fel szeretnénk venni, akkor nyissuk meg ezt az állományt a kedvenc szövegszerkesztõnkkel és írjuk be a tartományokat, soronként egyet:

```
saját.internet.szolgáltató.net
másik.internet.szolgáltató.com
felhasználók-internet.szolgáltató.ja
www.minta.org
```

Innentõl kezdve a listában szereplõ bármelyik géprõl tudunk levelet küldeni (feltéve, hogy az adott felhasználó hozzáfér a gépünkhöz). Ezzel gyönyörûen megoldhatjuk, hogy a felhasználóink képesek legyenek távolról is levelet küldeni a rendszerünkön keresztül anélkül, hogy mások pedig szemetet küldenének át rajtunk.

28.6. Komolyabb témák

A következõ szakaszban szóba kerülnek olyan komolyabb témák, mint például a levelek konfigurációja és a levelezés beállítása az egész tartomány számára.

28.6.1. Alapvetõ beállítások

Alapból képesnek kell lennünk leveleket küldeni külsõ gépekre egészen addig, amíg az /etc/resolv.conf állomány a megfelelõ beállításokat tartalmazza vagy egy saját névszervert futtatunk. Ha szeretnénk, hogy a gépünkre érkezõ levelek elérjék a FreeBSD-s gépünkön futó levéltovábbító ügynököt (például a sendmail programot), akkor erre két módszer kínálkozik:

- Futtassunk saját névszervert és hozzunk létre magunknak egy tartományt. Például `FreeBSD.org`.
- Közvetlenül a gépünkre küldessük a leveleket. Ezt úgy tehetjük meg, ha egybõl a gépünkhöz tartozó DNS névre küldetjük a leveleket. Például az `enyem.FreeBSD.org` címre.

Függetlenül attól, hogy a fentiek közül melyik megoldást választjuk, a levelek csak akkor tudnak eljutni közvetlenül a gépünkre, ha állandó, statikus IP-címmel rendelkezünk (tehát nem dinamikus címmel, amit általában a betárcsázós PPP kapcsolatokhoz szoktak kiosztani). Ha tûzfal mögött vagyunk, akkor valamilyen módon felénk kell irányítani az SMTP forgalmat is. Ha közvetlenül a gépünkön akarjuk fogadni a leveleket, akkor a következõ kettõ közül az egyik mindenképpen kelleni fog:

- Gondoskodjunk róla, hogy a hozzánk tartozó DNS-ben (legkisebb sorszámú) MX rekord a

gépünk IP-címére mutat.

- Gondoskodjunk róla, hogy a hozzánk tartozó DNS-ben nincs semmilyen MX rekord a gépünkhöz.

A fentiek közül bármelyik elég ahhoz, hogy közvetlenül a gépünkre érkezzen meg a levél.

Próbáljuk ki:

```
# hostname
enyem.FreeBSD.org
# host enyem.FreeBSD.org
enyem.FreeBSD.org has address 204.216.27.XX
```

Ha ezt látjuk, akkor minden gond nélkül lehet küldeni levelet a nevem@enyem.FreeBSD.org a címre (feltételezve, hogy a sendmail megfelelõen mûködik az `enyem.FreeBSD.org` címen).

Ha viszont ehhez hasonlót tapasztalunk:

```
# host enyem.FreeBSD.org
enyem.FreeBSD.org has address 204.216.27.XX
enyem.FreeBSD.org mail is handled (pri=10) by kozpont.FreeBSD.org
```

A gépünkre (`enyem.FreeBSD.org`) küldött összes levelet a `kozpont` szedi össze ugyanazon felhasználói névvel ahelyett, hogy közvetlenül a gépünkre küldeni ezeket.

Az iménti adatokat a DNS szerver határozza meg. A levelek továbbításával kapcsolatos információkat az *MX* mint *M*ail e*X*change DNS-rekord tárolja. Ha nincs ilyen MX rekord, akkor az IP-cím alapján közvetlenül az adott géphez kerül a levél.

Például a `freefall.FreeBSD.org` MX rekordja hajdanán így nézett ki:

```
freefall        MX  30  mail.crl.net
freefall        MX  40  agora.rdrop.com
freefall        MX  10  freefall.FreeBSD.org
freefall        MX  20  who.cdrom.com
```

Láthatjuk, hogy a `freefall` esetében több MX bejegyzés is szerepel. A legalacsonyabb MX-számú gép fogja kapni az erre a címre beérkezõ leveleket, amennyiben elérhetõ. Ha valamilyen okból nem érhetõ el, akkor helyette ideiglenesen a többiek (melyeket néha csak "tartalék MX-eknek" neveznek) veszik át a levelet és átadják a legalacsonyabb számúnak, amint az újra elérhetõvé válik.

A tartalék jelleggel megadott MX gépek akkor érnek ténylegesen valamit, ha teljesen máshonnan csatlakoznak az internethez. Az internet szolgáltató vagy egy ismerõsünk gépe valószínûleg minden további nélkül segít ennek megoldásában.

28.6.2. Egy egész tartomány leveleinek kezelése

Egy levelezõ szerver beállításához valahogy meg kell tudnunk oldalni, hogy a különbözõ munkaállomásokra küldött levelek közvetlenül hozzá fussanak be. Alapvetõen tehát arról lenne szó, hogy a tartományunkon (ez ebben az esetben a `*.FreeBSD.org`) belüli gépekre címzett levelekre ez a gép "tart igényt" és így ezek ide irányítódnak át, majd a felhasználók errõl a központi levelezõ szerverrõl kapják meg a leveleiket.

Az életünk megkönnyítéséhez minden felhasználónak létrehozzuk a saját *felhasználói nevét* a levelezõ szerveren is. Ezt az adduser(8) paranccsal gyorsan el is végezhetjük.

A levelezõ szerver lesz a hálózat összes munkaállomásához kirendelt levélváltó. Ezt a DNS beállításai között így adhatjuk meg:

```
enyem.FreeBSD.org   A   204.216.27.XX       ; Munkaállomás
            MX  10 kozpont.FreeBSD.org  ; Levelezõ szerver
```

Ezzel lényegében az A rekord figyelmen kívül hagyásával átirányítjuk a munkaállomások számára érkezõ összes levelet a levelezõ szerverre. A levelek tehát az MX rekord által mutatott címre mennek ki.

Ezt önállóan nem tudjuk elvégezni, hacsak nem futattunk egy saját DNS szervert. Ha nincsen vagy nem is tudunk DNS szervert futtatni, akkor ebben a kérdésben egyeztessünk az internet-szolgáltatónkkal vagy bárkivel, aki a DNS beállításaiért felelõs.

Ha virtuális e-mail címket is kezelünk, akkor a most következõ információ még a hasznunkra lehet. A példa kedvéért most feltesszük, hogy a tartományunkban van egy ügyfelünk, jelen esetben az `ugyfel1.org`, és azt akarjuk, hogy az `ugyfel1.org` címére küldött levelek a saját levelezõ szerverünkre kerüljenek át, a `level.sajat.com` címre. A DNS-t ehhez így kell beállítani:

```
ugyfel1.org     MX  10  level.sajat.com
```

Ha csak az `ugyfel1.org` levelezését akarjuk kezelni, akkor ahhoz *nem* kell külön A rekord.

Vigyázzunk, mert az `ugyfel1.org` csak akkor pingelhetõ, ha létezik hozzá A rekord.

Befejezésül a levelezõ szerverünkön futó sendmail számára is fel kell tárnunk, hogy milyen tartományokhoz és/vagy hálózati nevekhez fogadjon leveleket. Ezt több módon is elvégezhetjük. A következõk bármelyik megfelel erre a célra:

- A `FEATURE(use_cw_file)` használata esetén vegyük fel a címeket az /etc/mail/local-host-names állományba. Ha a sendmail 8.10 elõtti változatai esetében ehhez az /etc/sendmail.cw állományra lesz szükségünk.
- Tegyük be a `Cwsajat.cimunk.com` sort az /etc/sendmail.cf vagy a sendmail 8.10 és késõbbi változatai esetén az /etc/mail/sendmail.cf állományba.

28.7. SMTP és az UUCP

A FreeBSD-hez tartozó sendmail olyan gépek számára lett kialakítva, amelyek közvetlenül az internethez csatlakoznak. Az UUCP használatával levelezõ rendszerek számára egy másik konfigurációs állományt kell telepíteni a sendmail számára.

Az /etc/mail/sendmail.cf állítása kézzel egyáltalán nem könnyû. A sendmail 8. változata ráadásul a konfigurációs állományokat az m4(1) elõfeldolgozó segítségével gyártja le, ahol a tényleges beállítások egy magasabb absztrakciós szinten jelennek meg. Az m4(1) típusú konfigurációs állományok a /usr/shared/sendmail/cf könyvtárban találhatóak. A cf alkönyvtárban levõ README állomány igyekszik a felhasználót bevezetni az m4(1) alapú beállítások világába.

A `mailertable` nevû lehetõség használatával tudjuk a legjobban támogatni az UUCP protokollon keresztüli kézbesítést. Ezzel felépül egy olyan adatbázis, amelyet a sendmail fel tud használni a továbbítást érintõ döntésekben.

Ehhez elsõként hozzuk is létre a saját .mc állományunkat. Ehhez a /usr/shared/sendmail/cf/cf könyvtár tartalmaz néhány példát. Hívjuk most ezt az állomnyunkat ize.mc néven. A következõ módszerrel tudjuk egy valós sendmail.cf állománnyá alakítani:

```
# cd /etc/mail
# make ize.cf
# cp ize.cf /etc/mail/sendmail.cf
```

Egy átlagos .mc állomány egyébként valahogy így épül fel:

```
VERSIONID(`verziószám') OSTYPE(bsd4.4)

FEATURE(accept_unresolvable_domains)
FEATURE(nocanonify)
FEATURE(mailertable, `hash -o /etc/mail/mailertable')

define(`UUCP_RELAY', sajat.uucp.relay)
define(`UUCP_MAX_SIZE', 200000)
define(`confDONT_PROBE_INTERFACES')

MAILER(local)
MAILER(smtp)
MAILER(uucp)

Cw    sajat.al.nev
Cw    azuucpgepneve.UUCP
```

Az `accept_unresolvable_domains`, `nocanonify` és `confDONT_PROBE_INTERFACES` lehetõségekre hivatkozó sorok megakadályozzák, hogy a levél kézbesítésében a DNS is szerepet játsszon. Az `UUCP_RELAY` az UUCP alapú kézbesítés támogatását engedélyezi. Egyszerûen csak írjunk ide egy internetes hálózati nevet, amely képes feldolgozni az .UUCP általtartomány címeit. Az esetek többségében ide az internet-

szolgáltatónk levelek továbbküldéséért felelõs gépe kerül.

Miután ezzel végeztünk, szükségünk lesz még az /etc/mail/mailertable állományra is. Ha a külvilág felé csak egyetlen összeköttetést használunk a levelekhez, akkor az alábbi pontosan megfelel:

```
#
# makemap hash /etc/mail/mailertable.db < /etc/mail/mailertable
.                           uucp-dom:sajat.uucp.relay
```

Egy bonyolultabb példa pedig így néz ki:

```
#
# makemap hash /etc/mail/mailertable.db < /etc/mail/mailertable
#
horus.interface-business.de   uucp-dom:horus
.interface-business.de        uucp-dom:if-bus
interface-business.de         uucp-dom:if-bus
.heep.sax.de                  smtp8:%1
horus.UUCP                    uucp-dom:horus
if-bus.UUCP                   uucp-dom:if-bus
.                             uucp-dom:
```

Az elsõ három sor azokat a speciális eseteket kezeli, ahol a tartomány felé küldött levelek nem az alapértelmezett úton visszük tovább, hanem valamelyik UUCP szomszéd felé és így "le tudjuk rövidíteni" a kézbesítés útvonalát. Az ezeket követõ sor dolgozza fel a helyi Ethernet tartomány felé STMP protokollal továbbítható leveleket. Végül az UUCP szomszédokat is felsoroljuk az .UUCP áltartomány jelölése szerint, így megengedjük, hogy a `uucp-szomszéd! címzett` felülbírálja az alapértelmezett szabályokat. Az utolsó sorban mindig egyetlen pont szerepel, ami minden másra illeszkedik, így az UUCP kézbesítés egy olyan UUCP szomszéd felé halad, amely a világ felé egy univerzális levelezõ átjárónak tekinthetõ. A `uucp-dom:` kulcsszó mögött szereplõ összes csomópont nevének érvényes UUCP szomszédra kell utalnia, amelyet a `uuname` paranccsal le is tudunk ellenõrizni.

A feladatból már csak annyi maradt hátra, hogy használat elõtt ezt az állományt át kell alakítani DBM adatbázis formátumba. Az ehhez szükséges parancsot érdemes mailertable állomány elejére bejegyzésben felírni. A mailertable megváltoztatásakor mindig le kell futtatni ezt a parancsot.

Utolsó jótanács: ha nem lennénk biztosak valamelyik kézbesítési útvonal mûködésében, ne felejtsük el a sendmail `-bt` beállítását. Ezzel a sendmail az ún. *címtesztelõ módban* (address test mode) indul el. Gépeljük be, hogy `3,0`, majd írjuk be a tesztelni kívánt címet. Az utolsó sorban láthatjuk a felhasznált belsõ levéltovábbító ügynököt, a célgépet, amellyel ezt meghívjuk, és a (valószínûleg az átfordított) címet. Innen a `Ctrl` + `D` billentyûkombinációval léphetünk ki.

```
% sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 ize@pelda.com
```

```
canonify           input: ize @ pelda . com
...
parse            returns: $# uucp-dom $@ sajat.uucp.relay $: ize < @ pelda . com . >
> ^D
```

28.8. Csak küldés beállítása

Gyakran elõfordulhat, hogy csak leveleket akarunk továbbküldeni. Mint például:

- Asztali számítógépünk van, de használni akarunk olyan programokat, mint például a send-pr(1). Ehhez az internet-szolgáltatón keresztül kell továbbküldeni a levelet.
- A számítógépünk egy olyan szerver, amely nem helyben kezeli a leveleket, ezért az összeset átküldi feldolgozásra.

Szinte bármelyik levélküldõ ügynök képes betölteni ezt az ûrt. Sajnos eléggé bonyolult helyesen beállítani úgy egy bármire képes levélküldõt, hogy egyszerûen csak szabaduljon meg a levelektõl. Ilyenkor a sendmail vagy a postfix használatával tulajdonképpen ágyúval lövünk verébre.

Továbbá, ha egy átlagos internet-hozzáféréssel rendelkezünk, adódhat, hogy a szerzõdés egyszerûen tiltja a "levelezõ szerver" futtatását.

Legegyszerûbben úgy tudjuk kielégíteni az ilyen jellegû igényeket, ha feltelepítjük a mail/ssmtp portot. A `root` felhasználóval adjuk ki a következõ parancsokat:

```
# cd /usr/ports/mail/ssmtp
# make install replace clean
```

Telepítése után a mail/ssmtp portot a mindössze négysoros /usr/local/etc/ssmtp/ssmtp.conf állománnyal állíthatjuk be:

```
root=valodiemail@minta.com
mailhub=level.minta.com
rewriteDomain=minta.com
hostname=_GEPNEV_
```

A `root` felhasználó számára feltétlenül egy valódi e-mail címet adjuk meg. A `level.minta.com` helyére az internet-szolgáltatónk kimenõ leveleket továbbító szerverét adjuk meg (bizonyos szolgáltatók ezt "kimenõ levelezõ szervernek" vagy "SMTP szervernek" nevezik).

Ne felejtsük el sendmail démont sem letiltani, beleértve a kimenõ levelek kezelését. Ennek részleteit lásd a A sendmail letiltásaban.

A mail/ssmtp használatánál még adhatunk meg további beállításokat is. A /usr/local/etc/ssmtp állományban vagy az ssmtp man oldalán találhatunk példákat és olvashatunk bõvebben a témáról.

Az ssmtp ilyen fajta beállításával a számítógépünkön levõ szoftverek is helyesen fognak mûködni, miközben nem sértjük meg az internet-szolgáltató elõírásait és nem tesszük lehetõvé, hogy a

számítógépünkrõl levélszemetet küldhessenek.

28.9. Levelezés betárcsázós kapcsolattal

Ha statikus IP-címünk van, akkor az alapértelmezett beállítások tökéletesen megfelelõek számunkra. Csupán a gépünkhöz tartozó internetes címet kell megadnunk a gépünk nevének és a sendmail elvégzi a többit.

Ha viszont dinamikusan kiosztott IP-címmel rendelkezünk és betárcsázós PPP kapcsolaton keresztül csatlakozunk az internethez, akkor valószínûleg az internet-szolgáltató levelezõ szerverén van egy postaládánk. Most tegyük fel, hogy a internet-szolgáltató tartománya a `szolgaltato.net` és a felhasználói név a `felhasznalo`, a gépünk neve pedig `otthoni.bsdm`, valamint az internet-szolgáltató részérõl levelezésre a `relay.szolgaltato.net` gépet használhatjuk.

A postaládánkból úgy tudjuk letölteni a leveleket, ha telepítünk hozzá egy programot. Erre a feladatra a fetchmail hibátlanul alkalmas, mivel több különbözõ protokollt ismer. Ez a program csomagként vagy a Portgyûjteménybõl (mail/fetchmail) is elérhetõ. Az internet-szolgáltatók erre általában a POP protokollt ajánlják fel. Ha a felhasználói PPP alkalmazást használjuk, állítsuk be az /etc/ppp/ppp.linkup állományt a következõ módon és így a csatlakozáskor maguktól letöltõdnek a leveleink:

```
MYADDR:
  !bg su felhasznalo -c fetchmail
```

Ha a sendmail segítségével küldjük tovább a leveleket a nem helyi hozzáférések felé (ahogy azt lentebb is láthatjuk), akkor minden bizonnyal a csatlakozáskor arra is szükségünk lesz, hogy a leveleket tároló sor is feldolgozódjon. Ezt úgy oldhatjuk meg, ha az /etc/ppp/ppp.linkup állományba a `fetchmail` parancs után a következõt tesszük:

```
  !bg su felhasznalo -c "sendmail -q"
```

Ez a példa feltételezi, hogy az `otthoni.bsdm` gépen van egy `felhasznalo` nevû felhasználónk. Az `otthoni.bsdm` gépen a `felhasznalo` felhasználói könyvtárában hozzunk létre egy .fetchmailrc állományt:

```
poll szolgaltato.net protocol pop3 fetchall pass TitkosJelszo
```

Ezt az állományt csak és kizárólag a `felhasznalo` olvashatja, mivel szerepel benne a hozzá tartozó `TitkosJelszo`.

Úgy tudunk a megfelelõ `from:` fejléccel küldeni, ha felvilágosítjuk a sendmail programot, hogy ne az felhasznalo@otthoni.bsdm címet, hanem a felhasznalo@szolgaltato.net címet használja. Sõt, a gyorsítás kedvéért a sendmail számára érdemes elárulni, hogy a `relay.szolgaltato.net` címen keresztül küldjön.

A munka elvégzéséhez elegendõ az alábbi .mc állomány:

```
VERSIONID(`otthoni.bsdm.mc 1.0')
OSTYPE(bsd4.4)dnl
FEATURE(nouucp)dnl
MAILER(local)dnl
MAILER(smtp)dnl
Cwlocalhost
Cwotthoni.bsdm
MASQUERADE_AS(`szolgaltato.net')dnl
FEATURE(allmasquerade)dnl
FEATURE(masquerade_envelope)dnl
FEATURE(nocanonify)dnl
FEATURE(nodns)dnl
define(`SMART_HOST', `relay.szolgaltato.net')
Dmotthoni.bsdm
define(`confDOMAIN_NAME',`otthoni.bsdm')dnl
define(`confDELIVERY_MODE',`deferred')dnl
```

Az elõzõ szakaszban találhatjuk meg annak a módját, hogy miként varázsoljunk ebbõl az .mc állományból egy sendmail.cf állományt. A sendmail.cf frissítése után pedig ne felejtsük el a sendmail újraindítását!

28.10. Az SMTP hitelesítése

Levelezõ szerverünkön az SMTP protokoll hitelesítésének (SMTP Authentication) engedélyezése több szempontból is elõnyökkel bír. Az SMTP hitelesítésének bekapcsolása egy újabb réteget képez a sendmail védelmében, és az olyan állandóan mozgásban levõ felhasználók számára is megoldást nyújt, akik anélkül képesek használni ugyanazt a levelezõ szervert, hogy minden alkalommal újrakonfigurálnák a levelezõ kliensüket.

1. Telepítsük fel a security/cyrus-sasl2 portot. A security/cyrus-sasl2 port több fordítási idejû beállítást támogat. Itt most az SMTP hitelesítését fogjuk használni, ezért gondoskodjunk a `LOGIN` opció engedélyezésérõl.
2. A security/cyrus-sasl2 telepítés után nyissuk meg szerkesztésre a /usr/local/lib/sasl2/Sendmail.conf állományt (vagy ha még nem létezne, hozzuk létre), és benne vegyük fel a következõ sort:

   ```
   pwcheck_method: saslauthd
   ```

3. Ezt követõen telepítsük a security/cyrus-sasl2-saslauthd portot, és tegyük bele az /etc/rc.conf állományba ezt a sort:

   ```
   saslauthd_enable="YES"
   ```

   Végezetül indítsuk el a saslauthd démont:

```
# /usr/local/etc/rc.d/saslauthd start
```

Ez a démon fog közvetíteni a sendmail és a FreeBSD passwd adatbázisa közti hitelesítésben. Ezzel elkerülhetjük az új felhasználói nevek és jelszavak felvételét az SMTP hitelesítés használatához, így a hozzáférések és a levelezés jelszava ugyanaz marad.

4. Most pedig írjuk hozzá az alábbi sorokat az /etc/make.conf állományhoz:

```
SENDMAIL_CFLAGS=-I/usr/local/include/sasl -DSASL
SENDMAIL_LDFLAGS=-L/usr/local/lib
SENDMAIL_LDADD=-lsasl2
```

Ezek a sorok állítják be a sendmail számára, hogy fordítás közben a cyrus-sasl2 függvényeit használja. A sendmail újrafordítása elõtt mindenképpen legyen fenn a cyrus-sasl2 port.

5. A sendmail újrafordítását a következõ parancsok végrehajtásával intézhetjük el:

```
# cd /usr/src/lib/libsmutil
# make cleandir && make obj && make
# cd /usr/src/lib/libsm
# make cleandir && make obj && make
# cd /usr/src/usr.sbin/sendmail
# make cleandir && make obj && make && make install
```

A sendmail fordítása esetén semmilyen problémának nem szabadna elõfordulnia, kivéve ha a /usr/src könyvtárat és a szükséges osztott könyvtárakat nem változtatjuk idõközben túlságosan gyakran.

6. A sendmail lefordítása és újratelepítése után szerkesszük át az /etc/mail/freebsd.mc állományt (vagy azt az .mc állományt, amelyet éppen használunk). Sok rendszergazda a hostname(1) parancs válaszát használja fel az .mc típusú állományok egyedi elnevezéséhez). Írjuk bele a következõ sorokat:

```
dnl set SASL options
TRUST_AUTH_MECH(`GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN')dnl
define(`confAUTH_MECHANISMS', `GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN')dnl
```

Ezek állítják be a sendmail számára a felhasználók hitelesítésére alkalmas különbözõ módszereket. Ha a pwcheck módszer helyett valami mást akarunk használni, akkor járjunk utána a dokumentációban.

7. Zárásul futassuk le a make(1) parancsot az /etc/mail könyvtárban. Így lefut az új .mc állományunk és létrejön egy freebsd.cf (vagy amilyen nevet az .mc állománynak megadtunk) .cf állomány. Ezután a `make install restart` parancs kiadásával másoltassuk át ezt a sendmail.cf helyére és szabályosan indítassuk újra a sendmail szolgáltatást. A

folyamatról részletesebb tájékoztatást az /etc/mail/Makefile állomány tud nyújtani.

Ha eddig minden a legnagyobb rendben történt, akkor most már képesek vagyunk bejelentkezési információt is átadni a levelezõ kliensnek és elküldeni egy tesztüzenetet. A hibák kiszûréséhez állítsuk a sendmail `LogLevel` opcióját az 13 értékre és figyeljük a /var/log/maillog állományt.

További felvilágosításért olvassuk el a sendmail SMTP hitelesítéssel foglalkozó oldalát (angolul).

28.11. Levelezõ kliensek

A levelezõ kliens (Mail User Agent, MUA) egy olyan alkalmazás, amelyik elektronikus levelek küldésére és fogadására használható. Azonkívül, ahogy az e-mail "fejlõdik" és egyre bonyolultabbá válik, a levelezõ kliensek is egyre inkább erõsebbé válnak abban a tekintetben, ahogy az e-maileket kezelik. Ezzel együtt a felhasználók is egyre több lehetõséget és rugalmasságot kapnak. A FreeBSD számos levelezõ klienst támogat, mindegyikük könnyedén telepíthetõ a FreeBSD Portgyûjteménye segítségével. A felhasználók választhatnak a grafikus kliensek, mint például az evolution vagy a balsa és a konzolos kliensek, például a mutt, pine vagy `mail` között, esetleg használhatják a nagyobb szervezetek részérõl felkínált webes felületeket is.

28.11.1. mail

A mail(1) a FreeBSD alapértelmezett levelezõ kliense. Egy olyan konzolos alkalmazás, amelyben elérhetjük az e-mailek küldéséhez és fogadásához szükséges összes alapvetõ funkciót, habár a csatolmányokat csak korlátozottan képes kezelni és csak a helyi postaládákat kezeli.

Annak ellenére, hogy a `mail` önmaga nem képes kommunikálni POP vagy IMAP szerverekkel, az ilyen postaládák tartalmát egy fetchmail-szerû alkalmazással (lásd A fetchmail használata) le tudjuk tölteni a számára is elérhetõ helyi mbox állományba.

A levelek küldéséhez és fogadásához egyszerûen hívjuk be a `mail` programot a következõ módon:

```
% mail
```

Ezután a /var/mail könyvtárban található felhasználói postaládánk tartalmát automatikusan beolvassa a `mail` segédprogram. Ha a postaláda üres, akkor a program egyből befejezi futását és közli, hogy nem talált levelet. Amikor viszont tudott beolvasni leveleket, megjelenik egy felület, ahol a beérkezett üzenetek listáját láthatjuk. Az üzenetek automatikusan sorszámozódnak, ahogy ezt az alábbi példa is szemlélteti:

```
Mail version 8.1 6/6/93.  Type ? for help.
"/var/mail/marcs": 3 messages 3 new
>N  1 root@localhost        Mon Mar  8 14:05  14/510   "proba"
 N  2 root@localhost        Mon Mar  8 14:05  14/509   "felhasznaloi hozzaferes"
 N  3 root@localhost        Mon Mar  8 14:05  14/509   "minta"
```

Az üzenetek olvasásának a `t` paranccsal kezdhetünk neki, amelyet az elolvasandó üzenet sorszáma

követ. Ebben a példában az elsõ e-mailt nyitjuk meg:

```
& t 1
Message 1:
From root@localhost  Mon Mar  8 14:05:52 2004
X-Original-To: marcs@localhost
Delivered-To: marcs@localhost
To: marcs@localhost
Subject: proba
Date: Mon,  8 Mar 2004 14:05:52 +0200 (SAST)
From: root@localhost (Charlie Root)

Ezt az uzenetet probabol kuldom, valaszolj ra, ha megkaptad.
```

Ahogy az a fenti példából is látszik, a `t` billentyû hatására az üzenet a teljes fejlécével együtt jelenik meg. Az üzenetek listáját a `h` billentyûvel hozhatjuk vissza.

Ha egy levélre válaszolni szeretnénk, akkor ezt a `mail` paranccsal is megtehetjük, vagy az `R` vagy az `r` parancsokkal. Az `R` arra utasítja a `mail` programot, hogy csak az üzenet küldõjének válaszoljon, míg az `r` hatására nem csupán a küldõ, hanem az üzenet összes címzettje megkapja a válaszunkat. A parancshoz hozzátûzhetjük egy levél sorszámát is, ekkor az adott levélre fogunk válaszolni. Miután kiadtuk a parancsot, írjuk meg a válaszunkat és új sorban kezdve zárjuk le az üzenetet egyetlen `.` beírásával. Valahogy így:

```
& R 1
To: root@localhost
Subject: Re: proba

Koszonom, megkaptam a leveledet.
.
EOT
```

Új levelet az `m` segítségével tudunk küldeni, ami után meg kell adnunk a címzettet. Egyszerre több címzettet is meg tudunk adni, ha a címzett helyén címeiket egy `,` karakterrel elválasztva soroljuk fel. Ezután a levél témája is megadható, amit végül a levél szövege követ. Az üzenetet egy új sorban megadott egyetlen `.` segítségével zárhatjuk le.

```
& mail root@localhost
Subject: Elsajatitottam a mail hasznalatat

Most mar en is tudok levelet irni es fogadni a mail hasznalataval... :)
.
EOT
```

Amikor a `mail` segédprogramban vagyunk, a `?` használatával bármikor segítséget kérhetünk, valamint a `mail` mûködésével kapcsolatban a mail(1) man oldalát érdemes felkeresni.

Ahogy azt már korábban is említettük, a mail(1) parancsot eredetileg nem készítették fel az csatolt állományok kezelésére, ezért igen gyengén bánik velük. Az újabb levelezõ kliensek, mint például a mutt, a csatolt állományokat sokkal intelligensebb módon kezelik. Ha viszont ragaszkodunk a `mail` használatához, akkor a converters/mpack port használatát érdemes megfontolnunk.

28.11.2. mutt

A mutt apró mérete ellenére egy igen komoly levelezõ kliens és remek lehetõségeket ajánl fel. Íme ízelítésképpen közülük néhány:

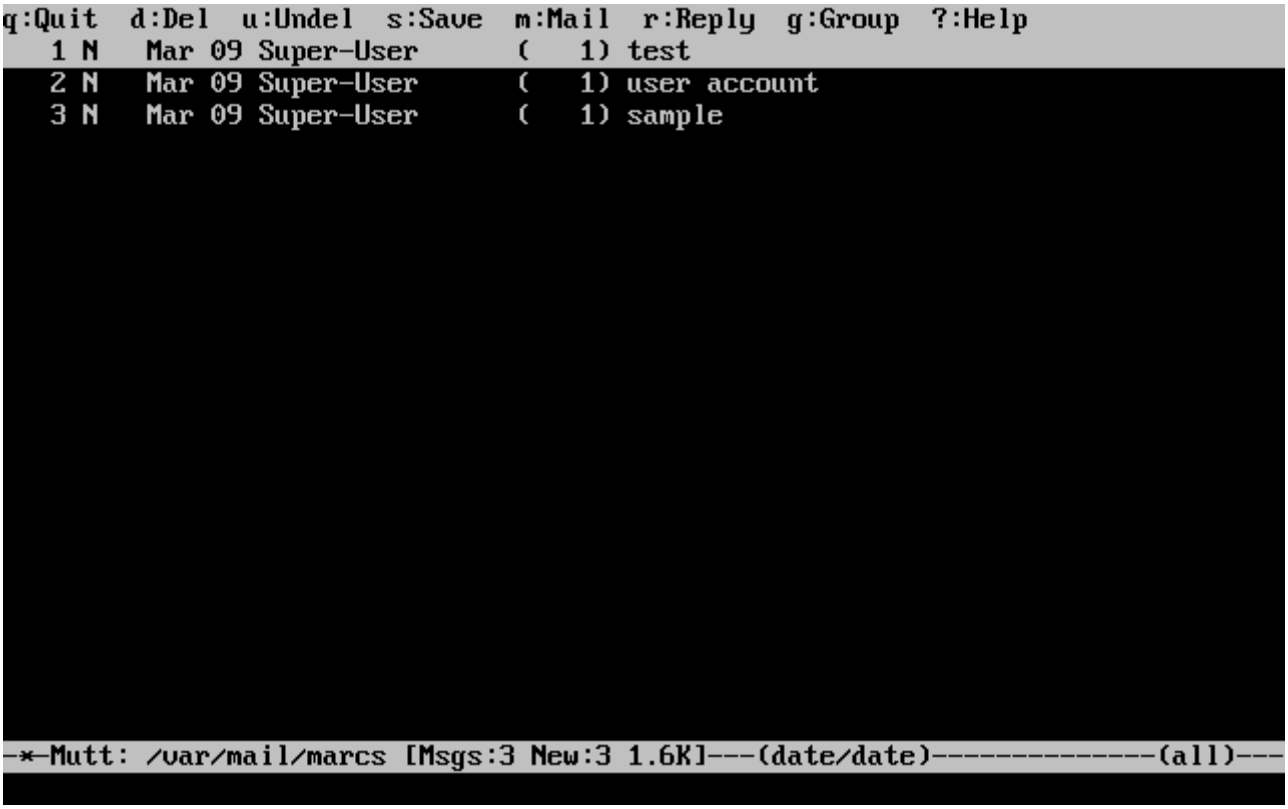
- Képes az üzeneteket szálakba rendezni
- Az e-mailek titkosítására és elektronikus aláírására támogatja a PGP használatát
- MIME támogatás
- Maildir támogatás
- Nagyfokú testreszabhatóság

Ezen lehetõségei révén a mutt ez egyik legfejlettebb levelezõ kliens. A mutt részletesebb bemutatását a http://www.mutt.org címen találjuk (angolul).

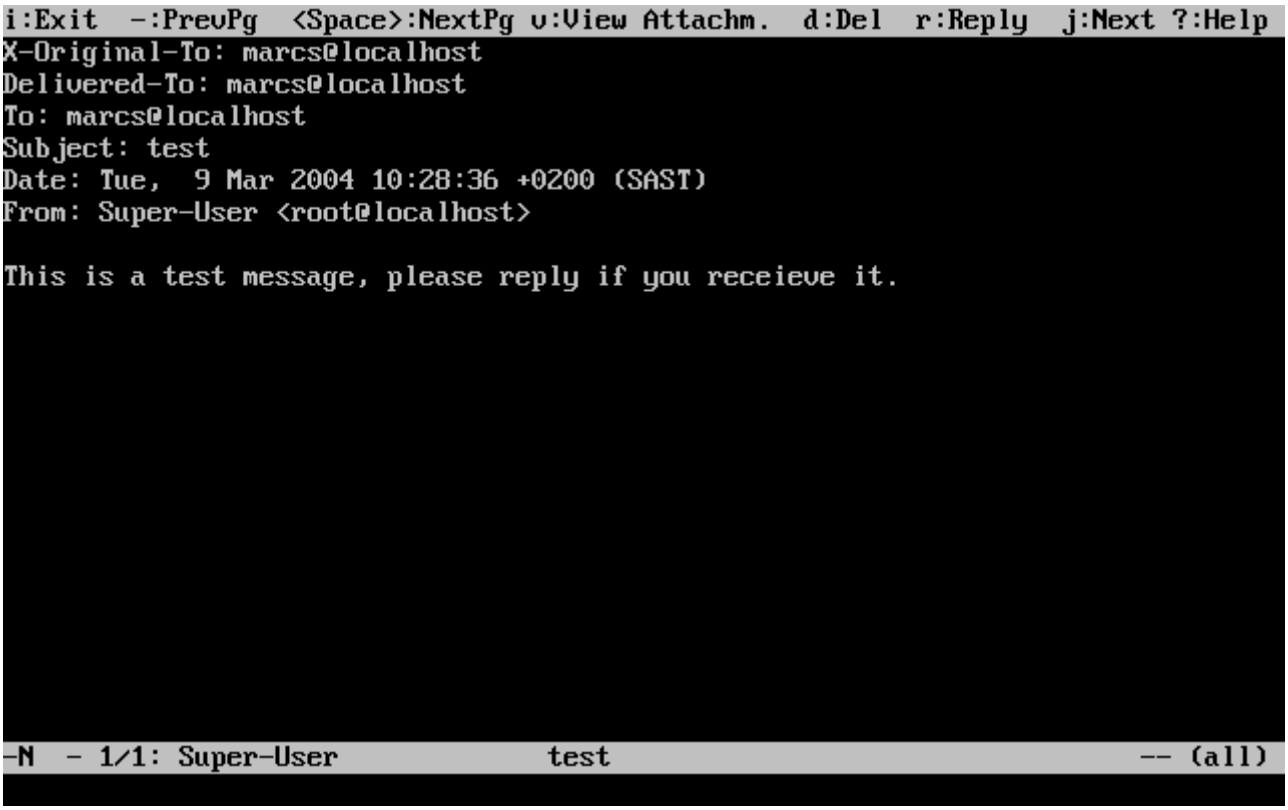
A mutt stabil változata a mail/mutt port használatával telepíthetõ fel, miközben a fejlesztés alatt levõ változatot a mail/mutt-devel port telepíti. Miután a portot sikerült felraknunk, a mutt az alábbi parancs begépelésével indítható el:

```
% mutt
```

A mutt indulása után automatikusan beolvassa a /var/mail könyvtárban megtalálható felhasználói postaládát és ha lehetséges, akkor megjeleníti a tartalmát. Ha nincsen levél a felhasználó postaládájában, akkor a mutt a felhasználó parancsaira vár. Ezen a képen a mutt üzenetlistája látható:



A levelek elolvasásához egyszerûen csak válasszuk ki a kurzorral és nyomjuk meg az `Enter` billentyût. Ezután a mutt így mutatja a levelet:



Ahogy azt már a mail(1) parancsnál is megszokhattuk, a mutt is lehetõvé teszi, hogy vagy csak a küldõnek, vagy pedig rajta kívül még az összes címzettnek is válaszoljunk. A levél küldõjének az `r` lenyomásával tudunk válaszolni. A csoportos válaszadáshoz pedig, ahol tehát a küldõn kívül a címzettek is megkapják a levelünket, a `g` billentyût kell használni.

A mutt az e-mailek létrehozásához és megválaszolásához a vi(1) szövegszerkesztõt használja. Ezt úgy tudjuk átállítani, ha a könyvtárunkban található .muttrc állományban átírjuk az `editor` változót, vagy értéket adunk az `EDITOR` környezeti változónak. A mutt beállításáról többet a http://www.mutt.org címen tudhatunk meg.

Egy új levél megírásához nyomjuk le az `m` gombot. Miután elláttuk érvényes témával a levelet, a mutt elindítja a vi(1) szövegszerkesztõt és nekiláthatunk a levél szövegének. Amint befejeztük, mentsük el és lépjünk ki a `vi` szerkesztõbõl. Ezután visszakapjuk a mutt felületét, ahol a küldendõ e-mail összefoglalását láthatjuk. A levelet végül az `y` lenyomásával küldhetjük el. Erre a következõ képen láthatunk egy példát:

```
y:Send  q:Abort  t:To  c:CC  s:Subj  a:Attach file  d:Descrip  ?:Help
    From: Marc Silver <marcs@localhost>
      To: Super-User <root@localhost>
      Cc:
     Bcc:
 Subject: Re: test
Reply-To:
     Fcc:
Security: Clear


-- Attachments
- I     1 /tmp/mutt-bsd-c0hobscQ              [text/plain, 7bit, us-ascii, 1.1K]




-- Mutt: Compose  [Approx. msg size: 1.1K   Atts: 1]----------------------------
```

A mutt ezenkívül még rengeteg segítséget is tartalmaz, amelyet a legtöbb menübõl a `?` gomb lenyomásával érhetünk el. A felsõ sorban mindig láthatjuk a kiadható parancsok rövid összefoglalását.

28.11.3. pine

A pine alapvetõen a kezdõ felhasználók számára íródott, de számos komolyabb lehetõséget is támogat.

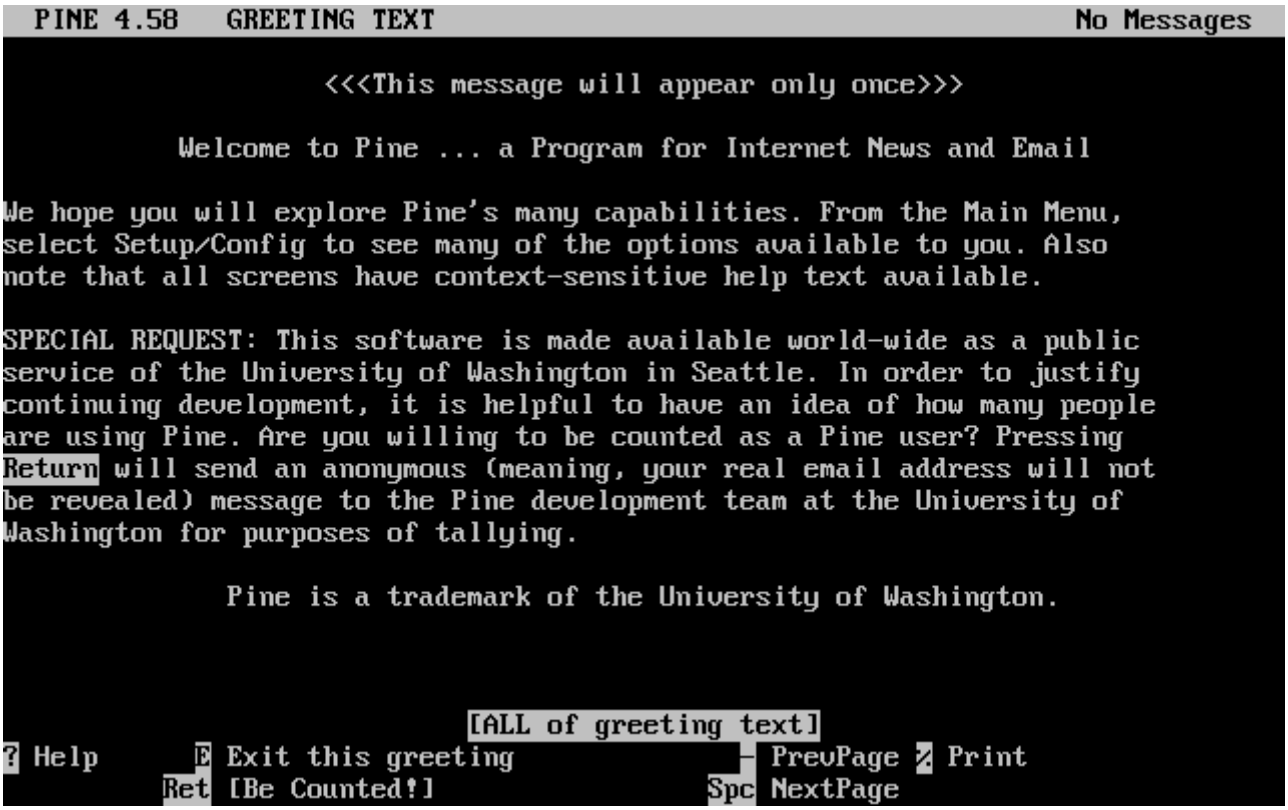
A pine szoftverrel kapcsolatban a múltban már rengeteg távolról kihasználható sebezhetõség látott napvilágot, és ennek köszönhetõen a támadók megfelelõen elõkészített e-mailek segítségével tetszõleges kódot tudnak futtatni a rendszeren levõ helyi felhasználókon keresztül. Noha az összes ilyen *ismert* hibát javították, de a FreeBSD biztonsági tisztje szerint a pine kódját biztonság szempontjából annyira hanyag módon írták, hogy további, eddig még felfedezetlen sebezhetõségeket is magában rejt. Ennek megfelelõen tehát a pine használata mindenkinek csak saját

> felelõsségre javasolt.

A pine jelenlegi verziója a mail/pine4 porton keresztül telepíthetõ. A telepítés lezajlása után a pine a következõ paranccsal indítható:

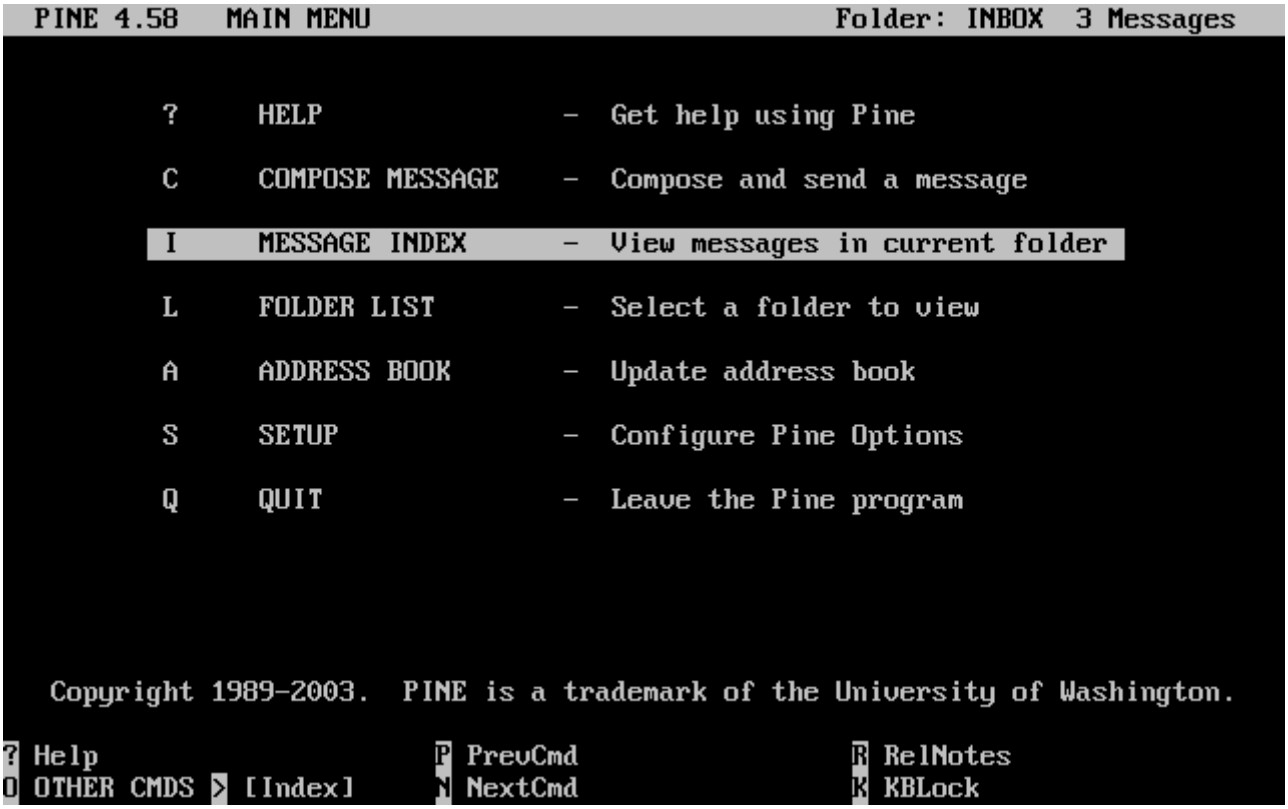
```
% pine
```

A pine elsõ futtatása során egy üdvözlõ üzenetet és egy rövid bemutatkozást jelenít meg, valamint a pine fejlesztõi arra kérik a felhasználókat, hogy küldjenek nekik egy névtelen üzenetet, amibõl le tudják szûrni mennyien használják a kliensüket. A névtelen üzenet elküldéséhez a `Enter` lenyomásával járulhatunk hozzá vagy az `E` használatával enélkül tudunk kilépni a képernyõrõl. Ezt az üdvözlõ képernyõt itt láthatjuk:



A felhasználó ezután a fõmenübe kerül, ahol a kurzorbillentyûkkel minden gond nélkül tudunk mozogni. Ebben a fõmenüben a levelek megírására, a leveleket tároló könyvtárak tallózására vagy éppen a címjegyzék karbantartására gyorsbillentyûket is használhatuk. A fõmenü alatt szerepel az adott menüben végrehajtható feladatokhoz tartozó gyorsbillentyûk rövid felsorolása.

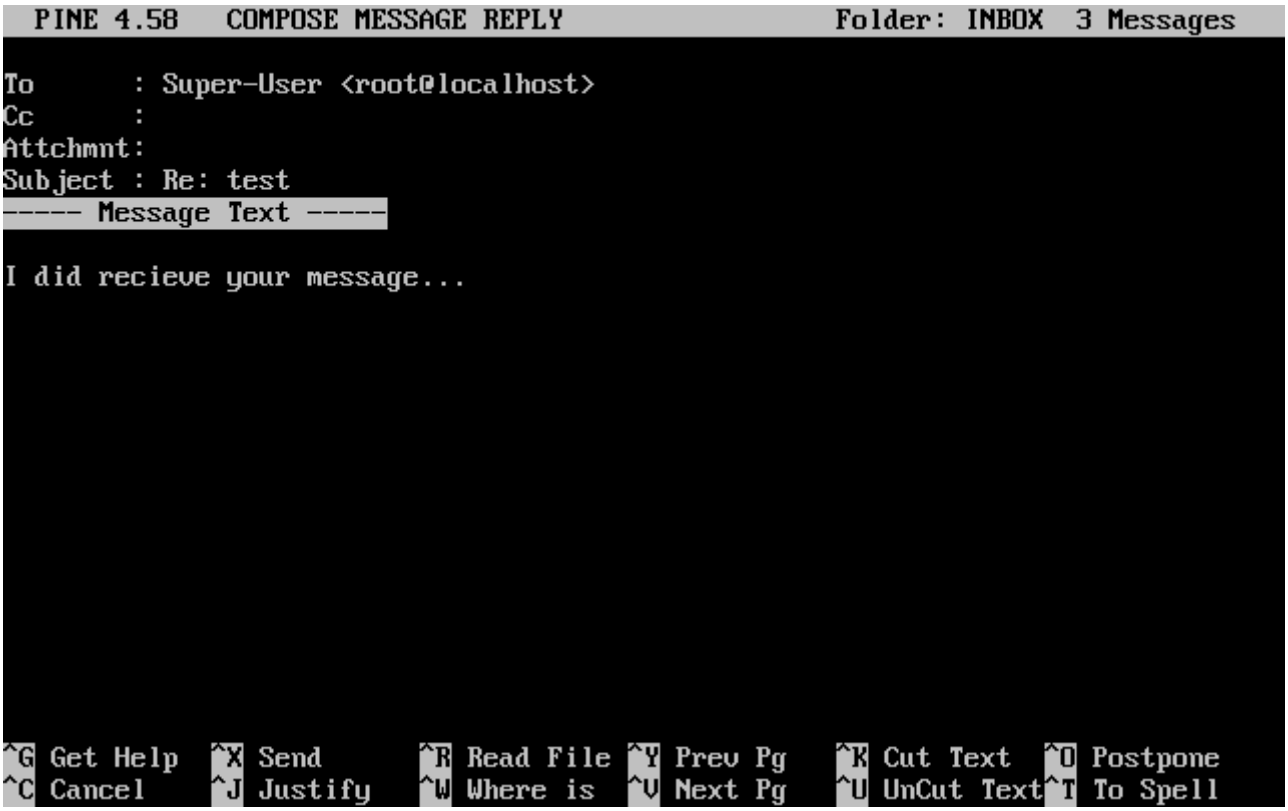
A pine alapértelmezés szerint az inbox könyvtárat nyitja meg. A bennelévõ üzenetek listájának megtekintéséhez nyomjuk a `I` gombot vagy válasszuk ki a lentihez hasonló módon a MESSAGE INDEX menüpontot:



Az üzenetek listájában az adott könyvtárban található üzenetek láthatjuk, és köztük a kurzorbillentyûkkel mozoghatunk. A kiemelt üzenet az `Enter` lenyomásával olvasható el.

A lenti képen egy ilyen példa üzenetet láthatunk a pine programban. A rendelkezésünkre álló gyorsbillentyûk ilyenkor is a képernyõ alján megjelennek referenciaként. Ilyen gyorsbillentyû többek közt az `r` gomb, amelynek hatására a klienssel megválaszolhatjuk a éppen látható üzenetet.

A pine kliensen belül a pico szövegszerkesztõ segítségével tudunk megválaszolni egy e-mailt, amely alapból a pine mellé települ. A pico megkönnyíti a navigációt az üzenetekben és sokkal elnézõbb a kezdõ felhasználókkal, mint például a vi(1) vagy a mail(1). Ha befejeztük a választ, az üzenetet a `Ctrl` + `X` billentyûkombinációval tudjuk elküldeni. A pine erre megerõsítést fog kérni.



A pine alkalmazás a fõmenübõl elérhetõ SETUP menüpont meghívásával szabható testre. A további részleteket a http://www.washington.edu/pine oldalon találhatjuk (angolul).

28.12. A fetchmail használata

A fetchmail egy mindentudó IMAP és POP kliens, amely lehetõvé teszi a felhasználók számára, hogy automatikusan töltsenek le leveleket távoli IMAP és POP szerverekrõl és lementsék azokat a helyi postaládáikba. Így a levelek sokkal könnyebben elérhetõek. A fetchmail a mail/fetchmail port segítségével telepíthetõ, és számos lehetõséget ajánl fel, többek közt:

- A POP3, APOP, KPOP, IMAP, ETRN és az ODMR protokollok ismerete.
- Képes SMTP használatával levelet továbbítani, és ennek révén a szûrés, továbbküldés és az álnevek használata a megszokott módon mûködik.
- Démonként futtatva képes adott idõközönként ellenõrizni a frissen érkezõ üzeneteket.
- Képes egyszerre több postaládát is kezelni, majd ezek tartalmát a beállításainak megfelelõen továbbküldeni a különbözõ helyi felhasználóknak.

Noha a fetchmail összes lehetõségének aprólékos bemutatása meghaladná ennek a leírásnak a kereteit, azért szót kerítünk néhány alapvetõ funkciójára. A fetchmail segédprogramnak a megfelelõ mûködéshez egy .fetchmailrc nevû konfigurációs állományra van szüksége. Ez az állomány tárolja a szerverekre vonatkozó, valamint a bejelentkezéshez szükséges információkat. Az állomány kényes tartalmára tekintettel azt javasoljuk, hogy csak a tulajdonosának engedélyezzük az olvasását:

```
% chmod 600 .fetchmailrc
```

Az alább ismertetésre kerülõ .fetchmailrc állományban azt láthatjuk, ahogy egyetlen felhasználó postaládáját érjük el a POP protokoll használatával. Arra utasítja a fetchmail programot, hogy csatlakozzon a `levelezes.com` címre a `joska` felhasználóval és az `XXX` jelszóval. Ebben a példában feltételezzük, hogy a `joska` nevû felhasználó létezik a rendszerünkben is.

```
poll levelezes.com protocol pop3 username "joska" password "XXX"
```

A következõ példában több POP és IMAP szerverhez csatlakozunk és ahol lehet, több helyi felhasználónak irányítjuk át a leveleket:

```
poll levelezes.com proto pop3:
user "joska", with password "XXX", is "jozsi" here;
user "andrea", with password "XXXX";
poll levelezes2.net proto imap:
user "jani", with password "XXXXX", is "hardstuff" here;
```

A fetchmail program a `-d` beállítás megadásával démonként is elindítható, amely után meg kell adni (másodpercekben) azt az idõközt, aminek elteltével a fetchmail lekérdi a .fetchmailrc állományban felsorolt szervereket. Az alábbi példában a fetchmail 600 másodpercenként kéri el a leveleket:

```
% fetchmail -d 600
```

A fetchmail további lehetõségeirõl és mûködésérõl a http://fetchmail.berlios.de/ oldalon olvashatunk (angolul).

28.13. A procmail használata

A procmail segédprogram egy hihetetlenül erõs alkalmazás, mellyel a beérkezõ leveleinket tudjuk szûrni. A felhasználók számára olyan "szabályok" megadását teszi lehetõvé, amelyekre aztán a rendszer illeszti a bejövõ leveleket, és az eredménynek megfelelõen elvégez bizonyos feladatokat vagy átirányítja a levelet más postaladákba és/vagy e-mail címekre. A procmail a mail/procmail porttal telepíthetõ fel. Miután ez sikerült, akár közvetlenül be is építhetjük a legtöbb levelezõ kliensbe. Errõl az adott levelezõ kliens dokumentációjában olvashatunk többet. A procmail úgy is integrálható, ha a felvesszük a következõ sort a procmail szolgáltatára igényt tartó felhasználó könyvtárában található .forward állományba:

```
"|exec /usr/local/bin/procmail || exit 75"
```

A következõ szakaszban láthatjuk a procmail néhány alapvetõ szabályát, valamint ezek rövid leírását. Ezeket a szabályokat a .procmailrc állományba kell beleírni, amely szintén a felhasználó könyvtárában leledzik.

Ezen szabályok többsége a procmailex(5) man oldalon is olvasható.

A felhasznalo@levelezes.com címrõl érkezõ leveleket irányítsuk át a jocim@levelezes2.com külsõ címre:

```
:0
* ^From.*felhasznalo@levelezes.com
! jocim@levelezes2.com
```

Minden 1000 byte-nál kisebb levelet küldjünk át a jocim@levelezes2.com külsõ címre:

```
:0
* < 1000
! jocim@levelezes2.com
```

Küldjük át az összes masik@levelezes.com címre küldött levelet a masik postaládába:

```
:0
* ^TOmasik@levelezes.com
masik
```

Küldjük az összes olyan levelet a /dev/null eszközre, amelyek a témájában szerepel a "Spam" szó:

```
:0
^Subject:.*Spam
/dev/null
```

Egy hasznos szabály, amellyel el tudjuk kapni a `FreeBSD.org` levelezési listáiról érkezõ leveleket és el tudjuk raktározni ezeket a saját postaládájukba:

```
:0
* ^Sender:.owner-freebsd-\/[^@]+@FreeBSD.ORG
{
    LISTNAME=${MATCH}
    :0
    * LISTNAME??^\/[^@]+
    FreeBSD-${MATCH}
}
```

Chapter 29. Hálózati szerverek

29.1. Áttekintés

Ebben a fejezetben a UNIX® típusú rendszerekben leggyakrabban alkalmazott hálózati szolgáltatások közül fogunk néhányat bemutatni. Ennek során megismerjük a hálózati szolgáltatások különbözõ típusainak telepítését, beállítását, tesztelését és karbantartását. A fejezet tartalmát folyamatosan példákkal igyekszünk illusztrálni.

A fejezet elolvasása során megismerjük:

- hogyan dolgozzunk az inetd démonnal;
- hogyan állítsuk be a hálózati állományrendszereket;
- hogyan állítsunk be egy hálózati információs szervert a felhasználói hozzáférések megosztására;
- hogyan állítsuk be automatikusan a hálózati hozzáférésünket a DHCP használatával;
- hogyan állítsunk be névfeloldó szervereket;
- hogyan állítsuk be az Apache webszervert;
- hogyan állítsuk be az állományok átviteléért felelõs (FTP) szervert;
- a Samba használatával hogyan állítsunk be Windows®-os kliensek számára állomány- és nyomtatószervert;
- az NTP protokoll segítségével hogyan egyeztessük az idõt és dátumot, hogyan állítsunk be egy idõszervert;
- a szabványos naplózó démon, a `syslogd` beállítását hálózati keresztüli naplózásra.

A fejezet elolvasásához ajánlott:

- az /etc/rc szkriptek alapjainak ismerete;
- az alapvetõ hálózati fogalmak ismerete;
- a külsõ szoftverek telepítésének ismerete (Alkalmazások telepítése. csomagok és portok).

29.2. Az inetd"szuperszerver"

29.2.1. Áttekintés

Az inetd(8) démont gyakran csak "internet szuperszerverként" nevezik, mivel a helyi szolgáltatások kapcsolatainak kezeléséért felelõs. Amikor az inetd fogad egy csatlakozási kérelmet, akkor eldönti róla, hogy ez melyik programhoz tartozik és elindít egy példányt belõle, majd átadja neki a socketet (az így meghívott program a szabvány bemenetéhez, kimenetéhez és hibajelzési csatornájához kapja meg a socket leíróit). Az inetd használatával úgy tudjuk csökkenteni a rendszerünk terhelését, hogy a csak alkalmanként meghívott szolgáltatásokat nem futtatjuk teljesen független önálló módban.

Az inetd démont elsõsorban más démonok elindítására használjuk, de néhány triviális protokollt közvetlenül is képes kezelni, mint például a chargen, auth és a daytime.

Ebben a fejezetben az inetd beállításának alapjait foglaljuk össze mind parancssoros módban, mind pedig az /etc/inetd.conf konfigurációs állományon keresztül.

29.2.2. Beállítások

Az inetd mûködése az rc(8) rendszeren keresztül inicializálható. Az `inetd_enable` ugyan alapból a `NO` értéket veszi fel, vagyis tiltott, de a sysinstall használatával már akár a telepítés során bekapcsolható attól függõen, hogy a felhasználó milyen konfigurációt választott. Ha tehát a:

```
inetd_enable="YES"
```

vagy

```
inetd_enable="NO"
```

sort tesszük az /etc/rc.conf állományba, akkor azzal az inetd démont indíthatjuk el vagy tilthatjuk le a rendszer indítása során. Az

```
# /etc/rc.d/inetd rcvar
```

paranccsal lekérdezhetjük a pillanatnyilag érvényes beállítást.

Emellett még az inetd démonnak az `inetd_flags` változón keresztül különbözõ parancssori paramétereket is át tudunk adni.

29.2.3. Parancssori paraméterek

Hasonlóan a legtöbb szerverhez, az inetd viselkedését is befolyásolni tudjuk a parancssorban átadható különbözõ paraméterekkel. Ezek teljes listája a következõ:

```
inetd [-d] [-l] [-w] [-W] [-c maximum] [-C arány] [-a cím | név] [-p állomány] [-R arány] [-s maximum] [konfigurációs állomány]
```

Ezek a paraméterek az /etc/rc.conf állományban az `inetd_flags` segítségével adhatóak meg az inetd részére. Alapértelmezés szerint az `inetd_flags` értéke `-wW -C 60`, ami az inetd által biztosított szolgáltatások TCP protokollon keresztüli wrappelését kapcsolja be, illetve egy IP-címrõl nem engedi a felkínált szolgáltatások elérését percenként hatvannál többször.

A kezdõ felhasználók örömmel nyugtázhatják, hogy ezeket az alapbeállításokat nem szükséges módosítaniuk. A késõbbiekben majd fény derül arra, hogy a kiszolgálás gyakoriságának szabályozása remek védekezést nyújthat túlzottan nagy mennyiségû kapcsolódási kérelem ellen. A megadható paraméterek teljes listája az inetd(8) man oldalán olvasható.

-c *maximum*

Az egyes szolgáltatásokhoz egyszerre felépíthetõ kapcsolatok alapértelmezett maximális számát adja meg. Alapból ezt a démont nem korlátozza. A `max-child` beállítással ez akár szolgáltatásonként külön is megadható.

-C *arány*

Korlátozza, hogy egyetlen IP-címrõl alapból hányszor hívhatóak meg az egyes szolgáltatások egy percen belül. Ez az érték alapból korlátlan. A `max-connections-per-ip-per-minute` beállítással ez szolgáltatásonként is definiálható.

-R *arány*

Megadja, hogy egy szolgáltatást egy perc alatt mennyiszer lehet meghívni. Ez az érték alapértelmezés szerint 256. A 0 megadásával eltöröljük ezt a típusú korlátozást.

-s *maximum*

Annak maximumát adja meg, hogy egyetlen IP-címrõl egyszerre az egyes szolgáltatásokat mennyiszer tudjuk elérni. Alapból ez korlátlan. Szolgáltatásonként ezt a `max-child-per-ip` paraméterrel tudjuk felülbírálni.

29.2.4. Az inetd.conf állomány

Az inetd beállítását az /etc/inetd.conf konfigurációs állományon keresztül végezhetjük el.

Amikor az /etc/inetd.conf állományban módosítunk valamit, az inetd démont a következõ paranccsal meg kell kérnünk, hogy olvassa újra:

Példa 33. Az inetd konfigurációs állományának újraolvasása

```
# /etc/rc.d/inetd reload
```

A konfigurációs állomány minden egyes sora egy-egy démont ír le. A megjegyzéseket egy "#" jel vezeti be. Az /etc/inetd.conf állomány bejegyzéseinek formátuma az alábbi:

```
szolgáltatás-neve
socket-típusa
protokoll
{wait|nowait}[/max-child[/max-connections-per-ip-per-minute[/max-child-per-ip]]]
felhasználó[:csoport][/bejelentkezési-osztály]
szerver-program
szerver-program-paraméterei
```

Az IPv4 protokollt használó ftpd(8) démon bejegyzése például így néz ki:

```
ftp     stream  tcp     nowait  root    /usr/libexec/ftpd       ftpd -l
```

szolgáltatás-neve

Ez az adott démon által képviselt szolgáltatást nevezi meg, amelynek szerepelnie kell az /etc/services állományban. Ez határozza meg, hogy az inetd milyen porton figyelje a beérkezõ kapcsolatokat. Ha egy új szolgáltatást hozunk létre, akkor azt elõször az /etc/services állományba kell felvennünk.

csatlakozás-típusa

Ennek az értéke `stream`, `dgram`, `raw`, vagy `seqpacket` lehet. A `stream` típust használja a legtöbb kapcsolat-orientált TCP démon, miközben a `dgram` típus az UDP szállítási protokollt alkalmazó démonok esetében használatos.

protokoll

Valamelyik a következõk közül:

Protokoll	Magyarázat
tcp, tcp4	TCP IPv4
udp, udp4	UDP IPv4
tcp6	TCP IPv6
udp6	UDP IPv6
tcp46	TCP IPv4 és v6
udp46	UDP IPv4 és v6

{wait|nowait}[/max-child[/max-connections-per-ip-per-minute[/max-child-per-ip]]]

A `wait|nowait` beállítás mondja meg, hogy az inetd démonból meghívott démon saját maga képes-e kezelni kapcsolatokat. A `dgram` típusú kapcsolatok esetében egyértelmûen a `wait` beállítást kell használni, miközben a `stream` esetén, ahol általában több szálon dolgozunk, a `nowait` megadása javasolt. A `wait` hatására általában egyetlen démonnak adunk át több socketet, míg a `nowait` minden sockethez egy újabb példányt indít el.

Az inetd által indítható példányokat a `max-child` megadásával korlátozhatjuk. Ha tehát például az adott démon számára legfeljebb példány létrehozását engedélyezzük, akkor a `nowait` után `/10` beállítást kell megadnunk. A `/0` használatával korlátlan mennyiségû példányt engedélyezhetünk.

A `max-child` mellett még további két másik beállítás jöhet számításba az egyes démonok által kezelhetõ kapcsolatok maximális számának korlátozásában. A `max-connections-per-ip-per-minute` az egyes IP-címekrõl befutó lekezelhetõ kapcsolatok percenkénti számát szabályozza, így például ha itt a tizes értéket adjuk meg, akkor az adott szolgáltatáshoz egy IP-címrõl percenként csak tízszer férhetünk hozzá. A `max-child-per-ip` az egyes IP-címekhez egyszerre elindítható példányok számára ír elõ egy korlátot. Ezek a paraméterek segítenek megóvni rendszerünket az erõforrások akaratos vagy akaratlan kimerítésétõl és a DoS (Denial of Service) típusú támadásoktól.

Ebben a mezõben a `wait` vagy `nowait` valamelyikét kötelezõ megadni. A `max-child`, `max-connections-per-ip-per-minute` és `max-child-per-ip` paraméterek ellenben elhagyhatóak.

A `stream` típusú több szálon futó démonok a `max-child`, `max-connections-per-ip-per-minute` vagy

`max-child-per-ip` korlátozása nélkül egyszerûen csak így adhatóak meg: `nowait`.

Ha ugyanezt a démont tíz kapcsolatra lekorlátozzuk, akkor a következõt kell megadnunk: `nowait/10`.

Amikor pedig IP-címenként 20 kapcsolatot engedélyezünk percenként és mindössze 10 példányt, akkor: `nowait/10/20`.

Az iménti beállítások a fingerd(8) démon alapértelmezett paramétereinél is megtalálhatóak:

```
finger stream  tcp     nowait/3/10 nobody /usr/libexec/fingerd fingerd -s
```

Végezetül engedélyezzük 100 példányt, melyek közül IP-címenként 5 használható: `nowait/100/0/5`.

felhasználó

Ezzel azt a felhasználót adjuk meg, akinek a nevében az adott démon futni fog. Az esetek túlnyomó részében a démonokat a `root` felhasználó futtatja. Láthatjuk azonban, hogy biztonsági okokból bizonyos démonok a `daemon` vagy a legkevesebb joggal rendelkezõ `nobody` felhasználóval futnak.

szerver-program

A kapcsolat felépülésekor az itt teljes elérési úttal megadott démon indul el. Ha ezt a szolgáltatást maga az inetd belsõleg valósítja meg, akkor ebben a mezõben az `internal` értéket adjuk meg.

szerver-program-paraméterei

Ez a `szerver-program` beállítással együtt mûködik, és ebben a mezõben a démon meghívásakor alkalmazandó paramétereket tudjuk rögzíteni, amelyet a démon nevével kezdünk. Ha a démont a parancssorból a `sajátdémon -d` paranccsal hívnánk meg, akkor a `sajátdémon -d` lesz `szerver-program-paraméterei` beállítás helyes értéke is. Természetesen, ha a démon egy belsõleg megvalósított szolgáltatás, akkor ebben a mezõben is az `internal` fog megjelenni.

29.2.5. Védelem

Attól függõen, hogy a telepítés során mit választottunk, az inetd által támogatott szolgáltatások egyes része talán alapból engedélyezett is. Amennyiben egy adott démont konkrétan nem használunk, akkor érdemes megfontolni a letiltását. A kérdéses démon sorába tegyünk egy "#" jelet az /etc/inetd.conf állományba, majd olvastassuk újra az inetd beállításait. Egyes démonok, mint például az fingerd használata egyáltalán nem ajánlott, mivel a támadók számára hasznos információkat tudnak kiszivárogtatni.

Más démonok nem ügyelnek a védelemre, és a kapcsolatokhoz rendelt lejárati idejük túlságosan hosszú vagy éppen nincs is. Ezzel a támadónak lehetõsége van lassú kapcsolatokkal leterhelni az adott démont, ezáltal kimeríteni a rendszer erõforrásait. Ha úgy találjuk, hogy túlságosan sok az ilyen kapcsolat, akkor jó ötletnek bizonyulhat a démonok számára a `max-connections-per-ip-per-minute`, `max-child` vagy `max-child-per-ip` korlátozások elrendelése.

Alapértelmezés szerint a TCP kapcsolatok wrappelése engedélyezett. A hosts_access(5) man oldalon találhatjuk meg az inetd által meghívható különféle démonok TCP-alapú korlátozásainak lehetõségeit.

29.2.6. Egyéb lehetõségek

A daytime, time, echo, discard, chargen és auth szolgáltatások feladatainak mindegyikét maga az inetd is képes ellátni.

Az auth szolgáltatás a hálózati keresztül azonosítást teszi lehetõvé és bizonyos mértékig beállítható. A többit egyszerûen csak kapcsoljuk ki vagy be.

A témában az inetd(8) man oldalán tudunk még jobban elmerülni.

29.3. A hálózati állományrendszer (NFS)

A FreeBSD több állományrendszert ismer, köztük a hálózati állományrendszert (Network File System, NFS) is. Az NFS állományok és könyvtárak megosztását teszi lehetõvé a hálózaton keresztül. Az NFS használatával a felhasználók és a programok képesek majdnem úgy elérni a távoli rendszereken található állományokat, mintha helyben léteznének.

Íme az NFS néhány legjelentõsebb elõnye:

- A helyi munkaállomások kevesebb tárterületet használnak, mivel a közös adatokat csak egyetlen számítógépen tároljuk és megosztjuk mindenki között.
- A felhasználóknak nem kell a hálózat minden egyes gépén külön felhasználói könyvtárral rendelkezniük. Ezek ugyanis az NFS segítségével akár egy szerveren is beállíthatóak és elérhetõvé tehetõek a hálózaton keresztül.
- A különbözõ háttértárak, mint például a floppy lemezek, CD-meghajtók és Zip® meghajtók a hálózaton több számítógép között megoszthatóak. Ezzel csökkenteni tudjuk a hálózatunkban szükséges cserélhetõ lemezes eszközök számát.

29.3.1. Ahogy az NFS mûködik

Az NFS legalább két fõ részbõl rakható össze: egy szerverbõl és egy vagy több kliensbõl. A kliensek a szerver által megosztott adatokhoz képesek távolról hozzáférni. A megfelelõ mûködéshez mindössze csak néhány programot kell beállítani és futtatni.

A szervernek a következõ démonokat kell mûködtetnie:

Démon	Leírás
nfsd	Az NFS démon, amely kiszolgálja az NFS kliensektõl érkezõ kéréseket.
mountd	Az NFS csatlakoztató démonja, amely végrehajtja az nfsd(8) által átküldött kéréseket.

Démon	Leírás
rpcbind	Ez a démon lehetõvé teszi az NFS kliensek számára, hogy fel tudják deríteni az NFS szerver által használt portot.

A kliensen is futnia kell egy démonnak, amelynek a neve nfsiod. Az nfsiod démon az NFS szerver felõl érkezõ kéréseket szolgálja ki. A használata teljesen opcionális, csupán a teljesítményt hívatott javítani, de a normális és helyes mûködéshez nincs rá szükségünk. Az nfsiod(8) man oldalán errõl többet is megtudhatunk.

29.3.2. Az NFS beállítása

Az NFS beállítása viszonylag egyértelmûen adja magát. A mûködéséhez szükséges programok automatikus elindítása csupán néhány apró módosítást igényel az /etc/rc.conf állományban.

Az NFS szerveren gondoskodjunk róla, hogy az alábbi beállítások szerepeljenek az /etc/rc.conf állományban:

```
rpcbind_enable="YES"
nfs_server_enable="YES"
mountd_flags="-r"
```

A mountd magától el fog indulni, ha az NFS szervert engedélyezzük.

A kliensen a következõ beállítást kell felvennünk az /etc/rc.conf állományba:

```
nfs_client_enable="YES"
```

Az /etc/exports állomány adja meg, hogy az NFS milyen állományrendszereket exportáljon (vagy másképpen szólva "osszon meg"). Az /etc/exports állományban tehát a megosztani kívánt állományrendszereket kell szerepeltetnünk, és azt, hogy melyik számítógépekkel tudjuk ezeket elérni. A gépek megnevezése mellett a hozzáférésre további megszorításokat írhatunk fel. Ezek részletes leírását az exports(5) man oldalon találjuk meg.

Lássunk néhány példát az /etc/exports állományban megjelenõ bejegyzésekre:

A most következõ példákban az állományrendszerek exportálásának finomságait igyekszünk érzékeltetni, noha a konkrét beállítások gyakran a rendszerünktõl és a hálózati konfigurációtól függenek. Például, ha a /cdrom könytárat akarjuk három gép számára megosztani, akik a szerverrel megegyezõ tartományban találhatóak (ezért nem is kell megadnunk a tartományt) vagy mert egyszerûen megtalálhatók az /etc/hosts állományunkban. Az `-ro` beállítás az exportált állományrendszereket írásvédetté teszi. Ezzel a beállítással a távoli rendszerek nem lesznek képesek módosítani az exportált állományrendszer tartalmát.

```
/cdrom -ro gép1 gép2 gép3
```

A következõ sorban a /home könyvtárat három gép számára osztjuk meg, melyeket IP-címekkel adtunk meg. Ez olyan helyi hálózat esetén hasznos, ahol nem állítottunk be névfeloldást. Esetleg a belsõ hálózati neveket az /etc/hosts állományban is tárolhatjuk. Ezzel utóbbival kapcsolatban a hosts(5) man oldalt érdemes fellapoznunk. Az `-alldirs` beállítás lehetõvé teszi, hogy az alkönyvtárak is csatlakozási pontok lehessenek. Más szóval, nem fogja csatlakoztatni az alkönyvtárakat, de megengedi a kliensek számára, hogy csak azokat a könyvtárakat csatlakoztassák, amelyeket kell vagy amelyekre szükségünk van.

```
/home  -alldirs  10.0.0.2 10.0.0.3 10.0.0.4
```

A következõ sorban az /a könyvtárat úgy exportáljuk, hogy az állományrendszerhez két különbözõ tartományból is hozzá lehessen férni. A `-maproot=root` beállítás hatására a távoli rendszer `root` felhasználója az exportált állományrendszeren szintén `root` felhasználóként fogja írni az adatokat. Amennyiben a `-maproot=root` beállítást nem adjuk meg, akkor a távoli rendszeren hiába `root` az adott felhasználó, az exportált állományrendszeren nem lesz képes egyetlen állományt sem módosítani.

```
/a  -maproot=root  gep.minta.com doboz.haz.org
```

A kliensek is csak a megfelelõ engedélyek birtokában képesek elérni a megosztott állományrendszereket. Ezért a klienst ne felejtsük el felvenni a szerver /etc/exports állományába.

Az /etc/exports állományban az egyes sorok az egyes állományrendszerekre és az egyes gépekre vonatkoznak. A távoli gépek állományrendszerenként csak egyszer adhatóak meg, és csak egy alapértelmezett bejegyzésük lehet. Például tegyük fel, hogy a /usr egy önálló állományrendszer. Ennek megfelelõen az alábbi bejegyzések az /etc/exports állományban érvénytelenek:

```
# Nem használható, ha a /usr egy állományrendszer:
/usr/src   kliens
/usr/ports kliens
```

Egy állományrendszerhez, vagyis itt a /usr partícióhoz, két export sort is megadtunk ugyanahhoz a `kliens` nevû géphez. Helyesen így kell megoldani az ilyen helyzeteket:

```
/usr/src /usr/ports  kliens
```

Az adott géphez tartozó egy állományrendszerre vonatkozó exportoknak mindig egy sorban kell szerepelniük. A kliens nélkül felírt sorok egyetlen géphez tartozónak fognak számítani. Ezzel az állományrendszerek megosztását tudjuk szabályozni, de legtöbbek számára nem jelent gondot.

Most egy érvényes exportlista következik, ahol a /usr és az /exports mind helyi állományrendszerek:

```
# Osszuk meg az src és ports könyvtárakat a kliens01 és kliens02 részére, de csak a
# kliens01 férhessen hozzá rendszeradminisztrátori jogokkal:
/usr/src /usr/ports -maproot=root    kliens01
```

```
/usr/src /usr/ports                kliens02
# A kliensek az /exports könyvtárban teljes joggal rendelkeznek és azon belül
# bármit tudnak csatlakoztatni. Rajtuk kívül mindenki csak írásvédetten képes
# elérni az /exports/obj könyvtárat:
/exports -alldirs -maproot=root      kliens01 kliens02
/exports/obj -ro
```

A mountd démonnal az /etc/exports állományt minden egyes módosítása után újra be kell olvastatni, mivel a változtatásaink csak így fognak érvényesülni. Ezt megcsinálhatjuk úgy is, hogy küldünk egy HUP (hangup, avagy felfüggesztés) jelzést a már futó démonnak:

```
# kill -HUP `cat /var/run/mountd.pid`
```

vagy meghívjuk a `mountd` rc(8) szkriptet a megfelelõ paraméterrel:

```
# /etc/rc.d/mountd onereload
```

Az Az rc használata FreeBSD alattban tudhatunk meg részleteket az rc szkriptek használatáról.

Ezek után akár a FreeBSD újraindításával is aktiválhatjuk a megosztásokat, habár ez nem feltétlenül szükséges. Ha `root` felhasználónként kiadjuk a következõ parancsokat, akkor azzal minden szükséges programot elindítunk.

Az NFS szerveren tehát:

```
# rpcbind
# nfsd -u -t -n 4
# mountd -r
```

Az NFS kliensen pedig:

```
# nfsiod -n 4
```

Ezzel most már minden készen áll a távoli állományrendszer csatlakoztatására. A példákban a szerver neve `szerver` lesz, valamint a kliens neve `kliens`. Ha csak ideiglenesen akarunk csatlakoztatni egy állományrendszert vagy egyszerûen csak ki akarjuk próbálni a beállításainkat, a kliensen `root` felhasználóként az alábbi parancsot hajtsuk végre:

```
# mount szerver:/home /mnt
```

Ezzel a szerveren található /home könyvtárat fogjuk a kliens /mnt könyvtárába csatlakoztatni. Ha mindent jól beállítottunk, akkor a kliensen most már be tudunk lépni az /mnt könyvtárba és láthatjuk a szerveren található állományokat.

Ha a számítógép indításával automatikusan akarunk hálózati állományrendszereket csatlakoztatni, akkor vegyük fel ezeket az /etc/fstab állományba. Erre íme egy példa:

```
szerver:/home   /mnt    nfs rw  0   0
```

Az fstab(5) man megtalálhatjuk az összes többi beállítást.

29.3.3. Zárolások

Bizonyos alkalmazások (például a mutt) csak akkor mûködnek megfelelõen, ha az állományokat a megfelelõ módon zárolják. Az NFS esetében az rpc.lockd használható az ilyen zárolások megvalósítására. Az engedélyezéséhez mind a szerveren és a kliensen vegyük fel a következõ sort az /etc/rc.conf állományba (itt már feltételezzük, hogy az NFS szervert és klienst korábban beállítottuk):

```
rpc_lockd_enable="YES"
rpc_statd_enable="YES"
```

A következõ módon indíthatjuk el:

```
# /etc/rc.d/lockd start
# /etc/rc.d/statd start
```

Ha nincs szükségünk valódi zárolásra az NFS kliensek és az NFS szerver között, akkor megcsinálhatjuk azt is, hogy az NFS kliensen a mount_nfs(8) programnak az `-L` paraméter átadásával csak helyileg végzünk zárolást. Ennek további részleterõl a mount_nfs(8) man oldalon kaphatunk felvilágosítást.

29.3.4. Gyakori felhasználási módok

Az NFS megoldását a gyakorlatban rengeteg esetben alkalmazzák. Ezek közül most felsoroljuk a legelterjedtebbeket:

- Több gép között megosztunk egy telepítõlemezt vagy más telepítõeszközt. Ez így sokkal olcsóbb és gyakorta kényelmes megoldás abban az esetben, ha egyszerre több gépre akarjuk ugyanazt a szoftvert telepíteni.
- Nagyobb hálózatokon sokkal kényelmesebb lehet egy központi NFS szerver használata, ahol a felhasználók könyvtárait tároljuk. Ezek a felhasználói könyvtárak aztán megoszthatóak a hálózaton keresztül, így a felhasználók mindig ugyanazt a könyvárat kapják függetlenül attól, hogy milyen munkaállomásról is jelentkeztek be.
- Több géppel is képes így osztozni az /usr/ports/distfiles könyvtáron. Ezen a módon sokkal gyorsabban tudunk portokat telepíteni a gépekre, mivel nem kell külön mindegyikre letölteni az ehhez szükséges forrásokat.

29.3.5. Automatikus csatlakoztatás az amd használatával

Az amd(8) (automatikus csatlakoztató démon, az automatic mounter daemon) önmûködõen csatlakoztatja a távoli állományrendszereket, amikor azokon belül valamelyik állományhoz vagy könyvtárhoz próbálunk hozzáférni. Emellett az amd az egy ideje már inaktív állományrendszereket is automatikusan leválasztja. Az amd használata egy remek alternatívát kínál az általában az /etc/fstab állományban megjelenõ állandóan csatlakoztatott állományrendszerekkel szemben.

Az amd úgy mûködik, hogy kapcsolódik egy NFS szerver /host és /net könyvtáraihoz. Amikor egy állományt akarunk elérni ezeken a könyvtárakon belül, az amd kikeresi a megfelelõ távoli csatlakoztatást és magától csatlakoztatja. A /net segítségével egy IP-címrõl tudunk exportált állományrendszereket csatlakoztatni, miközben a /host a távoli gép hálózati neve esetében használatos.

Ha tehát a /host/izemize/usr könyvtárban akarunk elérni egy állományt, akkor az amd démonnak ahhoz elõször az `izemize` nevû géprõl exportált /usr könyvtárat kell csatlakoztatnia.

Példa 34. Egy exportált állományrendszer csatlakoztatása az amd használatával

Egy távoli számítógép által rendelkezésre bocsátott megosztásokat a `showmount` paranccsal tudjuk lekérdezni. Például az `izemize` gépen elérhetõ exportált állományrendszereket így láthatjuk:

```
% showmount -e izemize
Exports list on izemize:
/usr                               10.10.10.0
/a                                 10.10.10.0
% cd /host/izemize/usr
```

Ahogy a példában látjuk is, a `showmount` parancs a /usr könyvtárat mutatja megosztásként. Amikor tehát belépünk a /host/izemize/usr könyvtárba, akkor amd magától megpróbálja feloldani az `izemize` hálózati nevet és csatlakoztatni az elérni kívánt exportált állományrendszert.

Az amd az indító szkripteken keresztül az /etc/rc.conf alábbi beállításával engedélyezhetõ:

```
amd_enable="YES"
```

Emellett még az `amd_flags` használatával további paraméterek is átadható az amd felé. Alapértelmezés szerint az `amd_flags` tartalmaz az alábbi:

```
amd_flags="-a /.amd_mnt -l syslog /host /etc/amd.map /net /etc/amd.map"
```

Az /etc/amd.map állomány adja meg az exportált állományrendszerek alapértelmezett beállításait. Az /etc/amd.conf állományban az amd további lehetõségeit konfigurálhatjuk..

Ha többet is szeretnénk tudni a témáról, akkor az amd(8) és az amd.conf(8) man oldalakat javasolt

elolvasnunk.

29.3.6. Problémák más rendszerek használatakor

Némely PC-s ISA buszos Ethernet kártyákra olyan korlátozások érvényesek, melyek komoly hálózati problémák keletkezéséhez vezethetnek, különösen az NFS esetében. Ez a nehézség nem FreeBSD-függõ, de a FreeBSD rendszereket is érinti.

Ez gond általában majdnem mindig akkor merül fel, amikor egy (FreeBSD-s) PC egy hálózatba kerül többek közt a Silicon Graphic és a Sun Microsystems által gyártott nagyteljesítményû munkaállomásokkal. Az NFS csatlakoztatása és bizonyos mûveletek még hibátlanul végrehajtódnak, azonban hirtelen a szerver látszólag nem válaszol többet a kliens felé úgy, hogy a többi rendszertõl folyamatosan dolgozza felfele a kéréseket. Ez a kliens rendszeren tapasztalható csak, amikor a kliens FreeBSD vagy egy munkaállomás. Sok rendszeren egyszerûen rendesen le sem lehet állítani a klienst, ha a probléma egyszer már felütötte a fejét. Egyedüli megoldás gyakran csak a kliens újraindítása marad, mivel az NFS-ben kialakult helyzetet máshogy nem lehet megoldani.

Noha a "helyes" megoldás az lenne, ha beszereznénk egy nagyobb teljesítményû és kapacitású kártyát a FreeBSD rendszer számára, azonban egy jóval egyszerûbb kerülõút is található a kielégítõ mûködés eléréséhez. Ha a FreeBSD rendszer képviseli a *szervert*, akkor a kliensnél adjuk meg a `-w=1024` beállítást is a csatlakoztatásnál. Ha a FreeBSD rendszer a *kliens* szerepét tölti be, akkor az NFS állományrendszert az `-r=1024` beállítással csatlakoztassuk róla. Ezek a beállítások az fstab állomány negyedik mezõjében is megadhatóak az automatikus csatlakoztatáshoz, vagy manuális esetben a mount(8) parancsnak a `-o` paraméterrel.

Hozzá kell azonban tennünk, hogy létezik egy másik probléma, amit gyakran ezzel tévesztenek össze, amikor az NFS szerverek és kliensek nem ugyanabban a hálózatban találhatóak. Ilyen esetekben mindenképpen *gyõzõdjünk meg róla*, hogy az útválasztók rendesen továbbküldik a mûködéshez szükséges UDP információkat, különben nem sokat tudunk tenni a megoldás érdekében.

A most következõ példákban a `gyorsvonat` lesz a nagyteljesítményû munkaállomás (felület) neve, illetve a `freebsd` pedig a gyengébb teljesítményû Ethernet kártyával rendelkezõ FreeBSD rendszer (felület) neve. A szerveren az /osztott nevû könyvtárat fogjuk NFS állományrendszerként exportálni (lásd exports(5)), amelyet majd a /projekt könyvtárba fogunk csatlakoztatni a kliensen. Minden esetben érdemes lehet még megadnunk a `hard` vagy `soft`, illetve `bg` opciókat is.

Ebben a példában a FreeBSD rendszer (`freebsd`) lesz a kliens, és az /etc/fstab állományában így szerepel az exportált állományrendszer:

```
gyorsvonat:/osztott /projekt nfs rw,-r=1024 0 0
```

És így tudjuk manuálisan csatlakoztatni:

```
# mount -t nfs -o -r=1024 gyorsvonat:/osztott /projekt
```

Itt a FreeBSD rendszer lesz a szerver, és a `gyorsvonat`/etc/fstab állománya így fog kinézni:

```
freebsd:/osztott /projekt nfs rw,-w=1024 0 0
```

Manuálisan így csatlakoztathatjuk az állományrendszert:

```
# mount -t nfs -o -w=1024 freebsd:/osztott /projekt
```

Szinte az összes 16 bites Ethernet kártya képes mûködni a fenti írási vagy olvasási korlátozások nélkül is.

A kíváncsibb olvasók számára eláruljuk, hogy pontosan miért is következik be ez a hiba, ami egyben arra is magyarázatot ad, hogy miért nem tudjuk helyrehozni. Az NFS általában 8 kilobyte-os "blokkokkal" dolgozik (habár kisebb méretû darabkákat is tud készíteni). Mivel az Ethernet által kezelt legnagyobb méret nagyjából 1500 byte, ezért az NFS "blokkokat" több Ethernet csomagra kell osztani - még olyankor is, ha ez a program felsõbb rétegeiben osztatlan egységként látszik - ezt aztán fogadni kell, összerakni és *nyugtázni* mint egységet. A nagyteljesítményû munkaállomások a szabvány által még éppen megengedett szorossággal képesek ontani magukból az egy egységhez tartozó csomagokat, közvetlenül egymás után. A kisebb, gyengébb teljesítményû kártyák esetében azonban az egymáshoz tartozó, késõbb érkezõ csomagok ráfutnak a korábban megkapott csomagokra még pontosan azelõtt, hogy elérnék a gépet, így az egységek nem állíthatóak össze vagy nem nyugtázhatóak. Ennek eredményeképpen a munkaállomás egy adott idõ múlva megint próbálkozik, de ismét az egész 8 kilobyte-os blokkot küldi el, ezért ez a folyamat a végtelenségig ismétlõdik.

Ha a küldendõ egységek méretét az Ethernet által kezelt csomagok maximális mérete alá csökkentjük, akkor biztosak lehetünk benne, hogy a teljes Ethernet csomag egyben megérkezik és nyugtázódik, így elkerüljük a holtpontot.

A nagyteljesítményû munkaállomások természetesen továbbra is küldhetnek a PC-s rendszerek felé túlfutó csomagokat, de egy jobb kártyával az ilyen túlfutások nem érintik az NFS által használt "egységeket". Amikor egy ilyen túlfutás bekövetkezik, az érintett egységet egyszerûen újra elküldik, amelyet a rákövetkezõ alkalommal nagy valószínûséggel már tudunk rendesen fogadni, összerakni és nyugtázni.

29.4. Hálózati információs rendszer (NIS/YP)

29.4.1. Mi ez?

A hálózati információs szolgáltatást (Network Information Service, avagy NIS) a Sun Microsystems fejlesztette ki a UNIX® (eredetileg SunOS™) rendszerek központosított karbantartásához. Mostanra már lényegében ipari szabvánnyá nõtte ki magát, hiszen az összes nagyobb UNIX®-szerû rendszer (a Solaris™, HP-UX, AIX®, Linux, NetBSD, OpenBSD, FreeBSD stb.) támogatja a NIS használatát.

A NIS régebben sárga oldalak (Yellow Pages) néven volt ismert, de a különbözõ jogi problémák miatt késõbb ezt a Sun megváltoztatta. A régi elnevezést (és a yp rövidítést) azonban még

napjainkban is lehet néhol látni.

Ez egy RPC alapján mûködõ, kliens/szerver felépítésû rendszer, amely az egy NIS tartomány belül levõ számítógépek számára teszi lehetõvé ugyanazon konfigurációs állományok használatát. Segítségével a rendszergazda a NIS klienseket a lehetõ legkevesebb adat hozzáadásával, eltávolításával vagy módosításával képes egyetlen helyrõl beállítani.

Hasonló a Windows NT® tartományaihoz, és habár a belsõ implementációt tekintve már akadnak köztük jelentõs eltérések is, az alapvetõ funkciók szintjén mégis összevethetõek.

29.4.2. A témához tartozó fogalmak és programok

A NIS telepítése számos fogalom és fontos felhasználói program kerül elõ FreeBSD-n, akár egy NIS szervert akarunk beállítani, akár csak egy NIS klienst:

Fogalom	Leírás
NIS tartománynév	A NIS központi szerverei és az összes hozzájuk tartozó kliens (beleértve az alárendelt szervereket) rendelkezik egy NIS tartománynévvel. Hasonló a Windows NT® által használt tartománynevekhez, de a NIS tartománynevei semmilyen kapcsolatban nem állnak a névfeloldással.
rpcbind	Az RPC (Remote Procedure Call, a NIS által használt egyik hálózati protokoll) engedélyezéséhez lesz rá szükségünk. Ha az rpcbind nem fut, akkor sem NIS szervert, sem pedig NIS klienst nem tudunk mûködtetni.
ypbind	A NIS klienst "köti össze" a hozzá tartozó NIS szerverrel. A NIS tartománynevet a rendszertõl veszi, és az RPC használatával csatlakozik a szerverhez. Az ypbind a NIS környezet kliens és szerver közti kommunikációjának magját alkotja. Ha az ypbind leáll a kliens gépén, akkor nem tudjuk elérni a NIS szervert.

Fogalom	Leírás
ypserv	Csak a NIS szervereken szabad futnia, mivel ez maga a NIS szerver programja. Ha az ypserv(8) leáll, akkor a szerver nem lesz képes tovább kiszolgálni a NIS kéréseket (szerencsére az alárendelt szerverek képesek átvenni ezeket). A NIS bizonyos változatai (de nem az, amelyik a FreeBSD-ben is megjelenik) nem próbálnak meg más szerverekhez csatlakozni, ha bedöglik az aktuális használt szerver. Ezen gyakran egyedül csak a szervert képviselõ program (vagy akár az egész szerver) újraindítása segíthet, illetve az ypbind újraindítása a kliensen.
rpc.yppasswdd	Ez egy olyan program, amelyet csak a NIS központi szerverein kell csak futtatni. Ez a démon a NIS kliensek számára a NIS jelszavaik megváltoztatását teszi lehetõvé. Ha ez a démon nem fut, akkor a felhasználók csak úgy tudják megváltoztatni a jelszavukat, ha bejelentkeznek a központi NIS szerverre.

29.4.3. Hogyan mûködik?

A NIS környezetekben háromféle gép létezik: a központi szerverek, az alárendelt szerverek és a kliensek. A szerverek képezik a gépek konfigurációs információinak központi tárhelyét. A központi szerverek tárolják ezen információk hiteles másolatát, míg ezt az alárendelt szerverek redundánsan tükrözik. A kliensek a szerverekre támaszkodnak ezen információk beszerzéséhez.

Sok állomány tartalma megosztható ezen a módon. Például a master.passwd, a group és hosts állományokat meg szokták osztani NFS-en. Amikor a kliensen futó valamelyik programnak olyan információra lenne szüksége, amely általában ezekben az állományokban nála megtalálható lenne, akkor helyette a NIS szerverhez fordul.

A gépek típusai

- A *központi NIS szerver*. Ez a szerver, amely leginkább a Windows NT® elsõdleges tartományvezérlõjéhez hasonlítható tartja karban az összes, NIS kliensek által használt állományt. A passwd, group, és összes többi ehhez hasonló állomány ezen a központi szerveren található meg.

Egy gép akár több NIS tartományban is lehet központi szerver. Ezzel a lehetõséggel viszont itt most nem foglalkozunk, mivel most csak egy viszonylag kis méretû NIS környezetet feltételezünk.

- Az *alárendelt NIS szerverek*. A Windows NT® tartalék tartományvezérlõihez hasonlítanak, és az alárendelt NIS szerverek feladata a központi NIS szerveren tárolt adatok másolatainak karbantartása. Az alárendelt NIS szerverek a redundancia megvalósításában segítenek, aminek

leginkább a fontosabb környezetekben van szerepe. Emellett a központi szerver terhelésének kiegyenlítését is elvégzik. A NIS kliensek elsõként mindig ahhoz a NIS szerverhez csatlakoznak, amelytõl elõször választ kapnak, legyen akár az egy alárendelt szerver.

- A *NIS kliensek*. A NIS kliensek, hasonlóan a Windows NT® munkaállomásokhoz, a NIS szerveren (amely a Windows NT® munkaállomások esetében a tartományvezérlõ) keresztül jelentkeznek be.

29.4.4. A NIS/YP használata

Ebben a szakaszban egy példa NIS környezetet állítunk be.

Tervezés

Tegyük fel, hogy egy aprócska egyetemi labor rendszergazdái vagyunk. A labor, mely 15 FreeBSD-s gépet tudhat magáénak, jelen pillanatban még semmilyen központosított adminisztráció nem létezik. Mindegyik gép saját /etc/passwd és /etc/master.passwd állománnyal rendelkezik. Ezeket az állományokat saját kezûleg kell szinkronban tartani. Tehát ha most felveszünk egy felhasználót a laborhoz, akkor az `adduser` parancsot mind a 15 gépen ki kell adni. Egyértelmû, hogy ez így nem maradhat, ezért úgy döntöttük, hogy a laborban NIS-t fogunk használni, és két gépet kinevezünk szervernek.

Az iméntieknek megfelelõen a labor most valahogy így néz ki:

A gép neve	IP-cím	A gép szerepe
`ellington`	`10.0.0.2`	központi NIS
`coltrane`	`10.0.0.3`	alárendelt NIS
`basie`	`10.0.0.4`	tanszéki munkaállomás
`bird`	`10.0.0.5`	kliensgép
`cli[1-11]`	`10.0.0.[6-17]`	a többi kliensgép

Ha még nincs tapasztalatunk a NIS rendszerek összeállításában, akkor elõször jó ötlet lehet végiggondolni, miként is akarjuk kialakítani. A hálózatunk méretétõl függetlenül is akadnak olyan döntések, amelyeket mindenképpen meg kell hoznunk.

A NIS tartománynév megválasztása

Ez nem az a "tartománynév", amit megszokhattunk. Ennek a pontos neve "NIS tartománynév". Amikor a kliensek kérnek valamilyen információt, akkor megadják annak a NIS tartománynak a nevét is, amelynek részei. Így tud egy hálózaton több szerver arról dönteni, hogy melyikük melyik kérést válaszolja meg. A NIS által használt tartománynévre tehát inkább úgy érdemes gondolni, mint egy valamilyen módon összetartozó gépek közös nevére.

Elõfordul, hogy egyes szervezetek az interneten is nyilvántartott tartománynevüket választják NIS tartománynévnek. Ez alapvetõen nem ajánlott, mivel a hálózati problémák felderítése közben félreértéseket szülhet. A NIS tartománynévnek a hálózatunkon belül egyedinek kell lennie, és lehetõleg minél jobban írja le az általa csoportba sorolt gépeket. Például a Kis Kft. üzleti osztályát tegyük a "kis-uzlet" NIS tartományba. Ebben a példában most a `proba-tartomany` nevet választottuk.

A legtöbb operációs rendszer azonban (köztük a SunOS™) a NIS tartománynevet használja internetes tartománynévként is. Ha a hálózatunkon egy vagy több ilyen gép is található, akkor a NIS tartomány nevének az internetes tartománynevet *kell* megadnunk.

A szerverek fizikai elvárásai

Nem árt néhány dolgot fejben tartani, amikor a NIS szervernek használt gépet kiválasztjuk. Az egyik ilyen szerencsétlen dolog az a szintû függõség, ami a NIS kliensek felõl megfigyelhetõ a szerverek felé. Ha egy kliens nem tudja a NIS tartományon belül felvenni a kapcsolatot valamelyik szerverrel, akkor az a gép könnyen megbízhatatlanná válhat. Felhasználói- és csoportinformációk nélkül a legtöbb rendszer egy idõre le is merevedik. Ennek figyelembevételével tehát olyan gépet kell szervernek választanunk, amelyet nem kell gyakran újraindítani, és nem végzünk rajta semmilyen komoly munkát. A célnak legjobban megfelelõ NIS szerverek valójában olyan gépek, amelyek egyedüli feladata csak a NIS kérések kiszolgálása. Ha a hálózatunk nem annyira leterhelt, akkor még a NIS szerver mellett más programokat is futtathatunk, de ne feledjük, hogy ha a NIS szolgáltatás megszûnik, akkor az az *összes* NIS kliensen éreztetni fogja kedvezõtlen hatását.

A NIS szerverek

A NIS rendszerben tárolt összes információ általános példánya egyetlen gépen található meg, amelyet a központi NIS szervernek hívunk. Az információk tárolására szánt adatbázis pedig NIS táblázatoknak (NIS map) nevezzük. FreeBSD alatt ezek a táblázatok a /var/yp/tartománynév könyvtárban találhatóak, ahol a tartománynév a kiszolgált NIS tartományt nevezi meg. Egyetlen NIS szerver egyszerre akár több tartományt is kiszolgálhat, így itt több könyvtár is található, minden támogatott tartományhoz egy. Minden tartomány saját, egymástól független táblázatokkal rendelkezik.

A központi és alárendelt NIS szerverek az `ypserv` démon segítségével dolgozzák fel a NIS kéréseket. Az `ypserv` felelõs a NIS kliensektõl befutó kérések fogadásáért, és a kért tartomány valamint táblázat nevébõl meghatározza az adatbázisban tárolt állományt, majd innen visszaküldi a hozzá tartozó adatot a kliensnek.

A központi NIS szerver beállítása

A központi NIS szerver beállítása viszonylag magától értetõdõ, de a nehézségét az igényeink szabják meg. A FreeBSD alapból támogatja a NIS használatát. Ezért mindössze annyit kell tennünk, hogy a következõ sorokat betesszük az /etc/rc.conf állományba, és a FreeBSD gondoskodik a többirõl.

```
nisdomainname="proba-tartomany"
```

1. Ez a sor adja meg a hálózati beállítások (vagy például az újraindítás) során a NIS tartomány nevét, amely a korábbiak szerint itt most a `proba-tartomany`.

```
nis_server_enable="YES"
```

1. Ezzel utasítjuk a FreeBSD-t, hogy a hálózati alkalmazások következõ indításakor a NIS szervert is aktiválja.

```
nis_yppasswdd_enable="YES"
```

Ezzel engedélyezzük az `rpc.yppasswdd` démont, amely a korábban említettek szerint lehetõvé teszi a felhasználók számára, hogy a közvetlenül a kliensekrõl változtassák meg a NIS jelszavukat.

A konkrét NIS beállításainktól függõen további bejegyzések felvételére is szükségünk lehet. Erre késõbb még "az olyan NIS szervereknél, amelyek egyben NIS kliensek", vissza fogunk térni.

Miután ezeket beállítottuk, rendszeradminisztrátorként adjuk ki az `/etc/netstart` parancsot. Az /etc/rc.conf állományban szereplõ adatok alapján mindent beállít magától. Még mielõtt inicializálnánk a NIS táblázatokat, indítsuk el manuálisan az ypserv démont:

```
# /etc/rc.d/ypserv start
```

A NIS táblázatok inicializálása

A *NIS táblázatok* lényegében a /var/yp könyvtárban tárolt adatbázisok. A központi NIS szerver /etc könyvtárában található konfigurációs állományokból állítódnak elõ, egyetlen kivétellel: ez az /etc/master.passwd állomány. Ennek megvan a maga oka, hiszen nem akarjuk a `root` és az összes többi fontosabb felhasználóhoz tartozó jelszót az egész NIS tartománnyal megosztani. Ennek megfelelõen a NIS táblázatok inicializálásához a következõt kell tennünk:

```
# cp /etc/master.passwd /var/yp/master.passwd
# cd /var/yp
# vi master.passwd
```

El kell távolítanunk az összes rendszerszintû (`bin`, `tty`, `kmem`, `games`, stb), és minden olyan egyéb hozzáférést, amelyeket nem akarjuk közvetíteni a NIS kliensek felé (például a `root` és minden más nullás, vagyis rendszeradminisztrátori azonosítóval ellátott hozzáférést).

Gondoskodjunk róla, hogy az /var/yp/master.passwd állomány sem a csoport, sem pedig bárki más számára nem olvasható (600-as engedély)! Ennek beállításához használjuk az `chmod` parancsot, ha szükséges.

Ha végeztünk, akkor már tényleg itt az ideje inicializálni NIS táblázatainkat. A FreeBSD erre egy `ypinit` nevû szkriptet ajánl fel (errõl a saját man oldalán tudhatunk meg többet). Ez a szkript egyébként a legtöbb UNIX® típusú operációs rendszeren megtalálható, de nem az összesen. A Digital UNIX/Compaq Tru64 UNIX rendszereken ennek a neve `ypsetup`. Mivel most a központi NIS szerver táblázatait hozzuk létre, azért az `ypinit` szkriptnek át kell adnunk a `-m` opciót is. A NIS táblázatok elõállításánál feltételezzük, hogy a fentebb ismertetett lépéseket már megtettük, majd kiadjuk ezt a parancsot:

```
ellington# ypinit -m proba-tartomany
Server Type: MASTER Domain: proba-tartomany
Creating an YP server will require that you answer a few questions.
Questions will all be asked at the beginning of the procedure.
Do you want this procedure to quit on non-fatal errors? [y/n: n] n
Ok, please remember to go back and redo manually whatever fails.
If you don't, something might not work.
At this point, we have to construct a list of this domains YP servers.
rod.darktech.org is already known as master server.
Please continue to add any slave servers, one per line. When you are
done with the list, type a <control D>.
master server   :  ellington
next host to add:  coltrane
next host to add:  ^D
The current list of NIS servers looks like this:
ellington
coltrane
Is this correct?  [y/n: y] y

[ .. a táblázatok generálása .. ]

NIS Map update completed.
ellington has been setup as an YP master server without any errors.
```

Az üzenetek fordítása:

```
A szerver típusa: KÖZPONTI, tartomány: proba-tartomany
Az YP szerver létrehozásához meg kell válaszolni néhány kérdést az
eljárás megkezdése előtt.
Szeretnénk, ha az eljárás megszakadna a nem végzetes hibák esetén is? [i/n: n] n
Rendben, akkor ne felejtsük el manuálisan kijavítani a hibát, ha
valamivel gond lenne.  Ha nem tesszük meg, akkor előfordulhat, hogy
valami nem fog rendesen működni.  Most össze kell állítanunk egy listát
a tartomány YP szervereiről.
Jelenleg a rod.darktech.org a központi szerver.
Kérjünk, adjon meg további alárendelt szervereket, soronként egyet.
Amikor ezt befejeztük, a <control D> lenyomásával tudunk
kilépni.
központi szerver : ellington
következő gép    : coltrane
következő gép    : ^D
A NIS szerverek listája jelenleg a következő:
ellington
coltrane
Ez megfelelő?  [i/n: i] i

[ .. a táblázatok generálása .. ]

A NIS táblázatok sikeressen frissültek.
```

```
Az elligon szervert minden hiba nélkül sikerült központi szerverként
beállítani.
```

Az `ypinit` a /var/yp/Makefile.dist állományból létrehozza a /var/yp/Makefile állományt. Amennyiben ez létrejött, az állomány feltételezi, hogy csak FreeBSD-s gépek részvételével akarunk kialakítani egy egyszerveres NIS környezetet. Mivel a `proba-tartomany` még egy alárendelt szervert is tartalmaz, ezért át kell írnunk a /var/yp/Makefile állományt:

```
ellington# vi /var/yp/Makefile
```

Ezt a sort kell megjegyzésbe tennünk:

```
NOPUSH = "True"
```

(ha még nem lenne úgy).

Az alárendelt NIS szerverek beállítása

Az alárendelt NIS szerverek beállítása még a központinál is egyszerûbb. Jelentkezzünk be az alárendelt szerverre és az eddigieknek megfelelõen írjuk át az /etc/rc.conf állományt. Az egyetlen különbség ezúttal csupán annyi lesz, hogy az `ypinit` lefuttatásakor a `-s` opciót kell megadnunk (mint slave, vagyis alárendelt). A `-s` opció használatához a központi NIS szerver nevét is át kell adnunk, ezért a konkrét parancs valahogy így fog kinézni:

```
coltrane# ypinit -s ellington proba-tartomany

Server Type: SLAVE Domain: test-domain Master: ellington

Creating an YP server will require that you answer a few questions.
Questions will all be asked at the beginning of the procedure.

Do you want this procedure to quit on non-fatal errors? [y/n: n]  n

Ok, please remember to go back and redo manually whatever fails.
If you don't, something might not work.
There will be no further questions. The remainder of the procedure
should take a few minutes, to copy the databases from ellington.
Transferring netgroup...
ypxfr: Exiting: Map successfully transferred
Transferring netgroup.byuser...
ypxfr: Exiting: Map successfully transferred
Transferring netgroup.byhost...
ypxfr: Exiting: Map successfully transferred
Transferring master.passwd.byuid...
ypxfr: Exiting: Map successfully transferred
Transferring passwd.byuid...
ypxfr: Exiting: Map successfully transferred
```

```
Transferring passwd.byname...
ypxfr: Exiting: Map successfully transferred
Transferring group.bygid...
ypxfr: Exiting: Map successfully transferred
Transferring group.byname...
ypxfr: Exiting: Map successfully transferred
Transferring services.byname...
ypxfr: Exiting: Map successfully transferred
Transferring rpc.bynumber...
ypxfr: Exiting: Map successfully transferred
Transferring rpc.byname...
ypxfr: Exiting: Map successfully transferred
Transferring protocols.byname...
ypxfr: Exiting: Map successfully transferred
Transferring master.passwd.byname...
ypxfr: Exiting: Map successfully transferred
Transferring networks.byname...
ypxfr: Exiting: Map successfully transferred
Transferring networks.byaddr...
ypxfr: Exiting: Map successfully transferred
Transferring netid.byname...
ypxfr: Exiting: Map successfully transferred
Transferring hosts.byaddr...
ypxfr: Exiting: Map successfully transferred
Transferring protocols.bynumber...
ypxfr: Exiting: Map successfully transferred
Transferring ypservers...
ypxfr: Exiting: Map successfully transferred
Transferring hosts.byname...
ypxfr: Exiting: Map successfully transferred

coltrane has been setup as an YP slave server without any errors.
Don't forget to update map ypservers on ellington.
```

Most már lennie kell egy /var/yp/proba-tartomany nevû könyvtárunknak is. A központi NIS szerver táblázatainak másolata itt fognak tárolódni. Ezeket soha ne felejtsük el frissen tartani. Az alárendelt szervereken a következõ /etc/crontab bejegyzések pontosan ezt a feladatot látják el:

```
20      *       *       *       *       root    /usr/libexec/ypxfr passwd.byname
21      *       *       *       *       root    /usr/libexec/ypxfr passwd.byuid
```

Ez a két sor gondoskodik róla, hogy az alárendelt szerverek ne felejtsék el egyeztetni a táblázataikat a központi szerver táblázataival. Ezek a bejegyzések nem nélkülözhetetlenek a megfelelõ mûködéshez, mivel a központi szerver automatikusan feltölti az alárendelt szerverekre a létrejött változásokat. Mivel azonban a jelszavak létfontosságúak a szervertõl függõ rendszerek számára, ezért ajánlott explicit módon is elõírni a frissítést. Ez a forgalmasabb hálózatokon nagyobb jelentõséggel bír, mivel ott a táblázatok frissítése nem mindig fejezõdik be rendesen.

Most pedig futassuk le a `/etc/netstart` parancsot az alárendelt szervereken is, amivel így elindul a

NIS szerver.

A NIS kliensek

A NIS kliens az `ypbind` démon segítségével egy kötésnek (bind) nevezett kapcsolatot épít ki egy adott NIS szerverrel. Az `ypbind` ellenõrzi a rendszer alapértelmezett tartományát (ezt a `domainname` paranccsal állítottunk be), majd RPC kéréseket kezd szórni a helyi hálózaton. Ezek a kérések annak a tartománynak a nevét tartalmazzák, amelyhez az `ypbind` megpróbál kötést létrehozni. Ha az adott tartomány kiszolgálására beállított szerver észleli ezeket a kéréseket, akkor válaszol az `ypbind` démonnak, amely pedig feljegyzi a szerver címét. Ha több szerver is elérhetõ (például egy központi és több alárendelt), akkor az `ypbind` az elsõként válaszoló címét fogja rögzíteni. Innentõl kezdve a kliens közvetlenül ennek a szervernek fogja küldeni a NIS kéréseit. Az `ypbind` idõnként "megpingeli" a szervert, hogy meggyõzõdjön az elérhetõségérõl. Ha az `ypbind` egy adott idõn belül nem kap választ a ping kéréseire, akkor megszünteti a kötést a tartományhoz és nekilát keresni egy másik szervert.

A NIS kliensek beállítása

Egy FreeBSD-s gépet NIS kliensként meglehetõsen egyszerûen lehet beállítani.

1. Nyissuk meg az /etc/rc.conf állományt és a NIS tartománynév beállításához, valamint az `ypbind` elindításához a következõket írjuk bele:

   ```
   nisdomainname="proba-tartomany"
   nis_client_enable="YES"
   ```

2. A NIS szerveren található jelszavak importálásához távolítsuk el az összes felhasználói hozzáférést az /etc/master.passwd állományunkból és a `vipw` segítségével adjuk hozzá az alábbi sort az állomány végéhez:

   ```
   +:::::::::
   ```

   Ez a sor beenged bárkit a rendszerünkre, akinek a NIS szervereken van érvényes hozzáférése. A NIS klienseket ezzel a sorral sokféle módon tudjuk állítani. A hálózati csoportokról szóló szakaszban találunk majd errõl több információt. A téma mélyebb megismeréséhez az O'Reilly `Managing NFS and NIS` címû könyvét ajánljuk.

   Legalább helyi hozzáférést (vagyis amit nem NIS-en keresztül importálunk) azonban mindenképpen hagyjunk meg az /etc/master.passwd állományunkban, és ez a hozzáférés legyen a `wheel` csoport tagja. Ha valami gond lenne a NIS használatával, akkor ezen a hozzáférésen keresztül tudunk a gépre távolról bejelentkezni, majd innen `root` felhasználóra váltva megoldani a felmerült problémákat.

3. A NIS szerverrõl az összes lehetséges csoport-bejegyzést az /etc/group állományban így

tudjuk importálni:

```
+:*::
```

Miután elvégeztük ezeket a lépéseket, képesek leszünk futtatni az `ypcat passwd` parancsot, és látni a NIS szerver jelszavakat tartalmazó táblázatát.

29.4.5. A NIS biztonsága

Általában tetszõleges távoli felhasználó küldhet RPC kéréseket az ypserv(8) számára és kérheti le a NIS táblázatok tartalmát, feltéve, hogy ismeri a tartomány nevét. Az ilyen hitelesítés nélküli mûveletek ellen az ypserv(8) úgy védekezik, hogy tartalmaz egy "securenets" nevû lehetõséget, amellyel az elérhetõségüket tudjuk leszûkíteni gépek egy csoportjára. Az ypserv(8) indításakor ezeket az információkat a /var/yp/securenets állományból próbálja meg betölteni.

Az elérési útvonala megadható a `-p` opció használatával. Ez az állomány olyan bejegyzéseket tartalmaz, amelyekben egy hálózati cím és tõle láthatatlan karakterekkel elválasztva egy hálózati maszk szerepel. A "#" karakterrel kezdõdõ sorokat megjegyzésnek nyilvánítjuk. Egy minta securenets állomány valahogy így nézne ki:

```
# Engedélyezzük önmagunkról a csatlakozást -- kell!
127.0.0.1     255.255.255.255
# Engedélyezzük a 192.168.128.0 hálózatról érkezõ csatlakozásokat:
192.168.128.0 255.255.255.0
# Engedélyezzük a laborban található 10.0.0.0 és 10.0.15.255 közti
# címekkel rendelkezõ gépek csatlakozását:
10.0.0.0      255.255.240.0
```

Ha az ypserv(8) olyan címrõl kap kérést, amely illeszkedik az elõírt címek valamelyikére, akkor a szokásos módon feldolgozza azt. Ellenkezõ esetben a kérést figyelmen kívül hagyja és egy figyelmeztetést vesz fel hozzá a naplóba. Ha a /var/yp/securenets állomány nem létezik, akkor az `ypserv` tetszõleges géprõl engedélyezi a csatlakozást.

Az `ypserv` lehetõséget ad a Wietse Venema által fejlesztett TCP Wrapper csomag használatára is. Ezzel a rendszergazda a /var/yp/securenets állomány helyett a TCP Wrapper konfigurációs állományai alapján képes szabályozni az elérhetõséget.

Miközben mind a két módszer nyújt valamilyen fajta védelmet, de a privilegizált portok teszteléséhez hasonlóan az "IP álcázásával" (IP spoofing) sebezhetõek. Ezért az összes NIS-hez tartozó forgalmat tûzfallal kell blokkolnunk.

Az /var/yp/securenets állományt használó szerverek nem képesek az elavult TCP/IP implementációkat használó érvényes klienseket rendesen kiszolgálni. Egyes ilyen implementációk a címben a géphez tartozó biteket nullára állítják az üzenetszóráshoz, és/vagy ezért az üzenetszóráshoz használt cím kiszámításakor

nem tudja észleli a hálózati maszkot. A legtöbb ilyen probléma megoldható a kliens konfigurációjának megváltoztatásával, míg más problémák megoldása a kérdéses kliensek nyugdíjazását kívánják meg, vagy a /var/yp/securenets használatának elhagyását.

Egy régebbi TCP/IP implementációval üzemelõ szerveren pedig a /var/yp/securenets állomány használata kifejezetten rossz ötlet, és a hálózatunk nagy részében képes használhatatlanná tenni a NIS funkcióit.

A TCP Wrapper csomag alkalmazása a NIS szerverünk válaszadáshoz szükséges idejét is segít csökkenteni. Az ilyenkor jelentkezõ plusz késlekedés mellesleg elég nagy lehet ahhoz, hogy a klienseknél idõtúllépés következzen be, különösen a terheltebb hálózatokon vagy a lassú NIS szerverek esetében. Ha egy vagy több kliensünk is ilyen tüneteket mutat, akkor érdemes a kérdéses kliens rendszereket alárendelt NIS szerverekké alakítani és önmagukhoz rendelni.

29.4.6. Egyes felhasználók bejelentkezésének megakadályozása

A laborunkban van egy `basie` nevû gép, amely a tanszék egyetlen munkaállomása. Ezt a gépet nem akarjuk kivenni a NIS tartományból, de a központi NIS szerver passwd állománya mégis egyaránt tartalmazza a hallgatók és az oktatók eléréseit. Mit lehet ilyenkor tenni?

Adott felhasználók esetében le tudjuk tiltani a bejelentkezést a gépen még olyankor is, ha léteznek a NIS adatbázisában. Ehhez mindössze a kliensen az /etc/master.passwd állomány végére be kell tennünk egy `-felhasználónév` sort, ahol a *felhasználónév* annak a felhasználónak a neve, akit nem akarunk beengedni a gépre. Ezt leginkább a `vipw` használatán keresztül érdemes megtennünk, mivel a `vipw` az /etc/master.passwd állomány alapján végez némi ellenõrzést, valamint a szerkesztés befejeztével magától újragenerálja a jelszavakat tároló adatbázist. Például, ha a `bill` nevû felhasználót ki akarjuk tiltani a `basie` nevû géprõl, akkor:

```
basie# vipw
[vegyük fel a -bill sort a végére, majd lépjünk ki]
vipw: rebuilding the database...
vipw: done

basie# cat /etc/master.passwd

root:[jelszó]:0:0::0:0:The super-user:/root:/bin/csh
toor:[jelszó]:0:0::0:0:The other super-user:/root:/bin/sh
daemon:*:1:1::0:0:Owner of many system processes:/root:/sbin/nologin
operator:*:2:5::0:0:System &:/:/sbin/nologin
bin:*:3:7::0:0:Binaries Commands and Source,,,:/:/sbin/nologin
tty:*:4:65533::0:0:Tty Sandbox:/:/sbin/nologin
kmem:*:5:65533::0:0:KMem Sandbox:/:/sbin/nologin
games:*:7:13::0:0:Games pseudo-user:/usr/games:/sbin/nologin
news:*:8:8::0:0:News Subsystem:/:/sbin/nologin
man:*:9:9::0:0:Mister Man Pages:/usr/shared/man:/sbin/nologin
bind:*:53:53::0:0:Bind Sandbox:/:/sbin/nologin
uucp:*:66:66::0:0:UUCP pseudo-user:/var/spool/uucppublic:/usr/libexec/uucp/uucico
```

```
xten:*:67:67::0:0:X-10 daemon:/usr/local/xten:/sbin/nologin
pop:*:68:6::0:0:Post Office Owner:/nonexistent:/sbin/nologin
nobody:*:65534:65534::0:0:Unprivileged user:/nonexistent:/sbin/nologin
+:::::::::
-bill

basie#
```

29.4.7. A hálózati csoportok alkalmazása

Az elõzõ szakaszban ismertetett módszer viszonylag jól mûködik olyan esetekben, amikor nagyon kevés felhasználóra és/vagy számítógépre kell alkalmaznunk speciális megszorításokat. A nagyobb hálózatokban szinte *biztos*, hogy elfelejtünk kizárni egyes felhasználókat az érzékeny gépekrõl, vagy az összes gépen egyenként kell ehhez a megfelelõ beállításokat elvégezni, és ezzel lényegében elvesztjük a NIS legfontosabb elõnyét, vagyis a *központosított* karbantarthatóságot.

A NIS fejlesztõi erre a problémára a *hálózati csoportokat* létrehozásával válaszoltak. A céljuk és mûködésük szempontjából leginkább a UNIX®-os állományrendszerekben található csoportokhoz mérhetõek. A legnagyobb eltérés a numerikus azonosítók hiányában mutatkozik meg, valamint a hálózati csoportokat a felhasználókon kívül további hálózati csoportok megadásával is ki lehet alakítani.

A hálózati csoportok a nagyobb, bonyolultabb, többszáz felhasználós hálózatok számára jöttek létre. Egy részrõl ez nagyon jó dolog, különösen akkor, ha egy ilyen helyzettel kell szembenéznünk. Másrészrõl ez a mértékû bonyolultság szinte teljesen lehetetlenné teszi a hálózati csoportok egyszerû bemutatását. A szakasz további részében használt példa is ezt a problémát igyekszik illusztrálni.

Tételezzük fel, hogy laborunkban a NIS sikeres bevezetése felkeltette a fõnökeink figyelmét. Így a következõ feladatunk az lett, hogy terjesszük ki a NIS tartományt az egyetemen található néhány másik gépre is. Az alábbi két táblázatban az új felhasználók és az új számítógép neveit találjuk, valamint a rövid leírásukat.

Felhasználók nevei	Leírás
alpha, beta	az IT tanszék hétköznapi dolgozói
charlie, delta	az IT tanszék újdonsült dolgozói
echo, foxtrott, golf, ...	átlagos dolgozók
able, baker, ...	ösztöndíjasok

Gépek nevei	Leírás
haboru, halal, ehseg, szennyezes	A legfontosabb szervereink. Csak az IT tanszék dolgozói férhetnek hozzájuk.
buszkeseg, kapzsisag, irigyseg, harag, bujasag, lustasag	Kevésbé fontos szerverek. Az IT tankszék összes tagja el tudja érni ezeket a gépeket.

Gépek nevei	Leírás
egy, ketto, harom, negy, ...	Átlagos munkaállomások. Egyedül csak a *valódi* dolgozók jelentkezhetnek be ezekre a gépekre.
szemetes	Egy nagyon régi gép, semmi értékes adat nincs rajta. Akár még az ösztöndíjasok is nyúzhatják.

Ha ezeket az igényeket úgy próbáljuk meg teljesíteni, hogy a felhasználókat egyenként blokkoljuk, akkor minden rendszer passwd állományába külön fel kell vennünk a `-felhasználó` sorokat a letiltott felhasználókhoz. Ha csak egyetlen bejegyzést is kihagyunk, akkor könnyen bajunk származhat belõle. Ez a rendszer kezdeti beállítása során még talán nem okoz gondot, de az új felhasználókat *biztosan* el fogjuk felejteni felvenni a megfelelõ csoportokba. Elvégre Murphy is optimista volt.

A hálózati csoportok használata ilyen helyzetekben számos elõnyt rejt. Nem kell az egyes felhasználókat külön felvenni, egy felhasználót felveszünk valamelyik csoportba vagy csoportokba, és a csoportok összes tagjának egyszerre tudjuk tiltani vagy engedélyezni a hozzáféréseket. Ha hozzáadunk egy új gépet a hálózatunkhoz, akkor mindössze a hálózati csoportok bejelentkezési korlátozásait kell beállítani. Ha új felhasználót veszünk fel, akkor a felhasználót kell vennünk egy vagy több hálózati csoportba. Ezek a változtatások függetlenek egymástól, és nincs szükség "minden felhasználó és minden gép összes kombinációjára". Ha a NIS beállításainkat elõzetesen körültekintõen megterveztük, akkor egyetlen központi konfigurációs állományt kell módosítani a gépek elérésének engedélyezéséhez vagy tiltásához.

Az elsõ lépés a hálózati csoportokat tartalmazó NIS táblázat inicializálása. A FreeBSD ypinit(8) programja alapértelmezés szerint nem hozza létre ezt a táblázatot, de ha készítünk egy ilyet, akkor a NIS implementációja képes kezelni. Egy ilyen üres táblázat elkészítéséhez ennyit kell begépelni:

```
ellington# vi /var/yp/netgroup
```

Ezután elkezdhetjük felvenni a tartalmát. A példánk szerint legalább négy hálózati csoportot kell csinálnunk: az IT dolgozóinak, az IT új dolgozóinak, a normál dolgozóknak és az ösztöndíjasoknak.

```
IT_DOLG       (,alpha,proba-tartomany)    (,beta,proba-tartomany)
IT_UJDOLG     (,charlie,proba-tartomany)  (,delta,proba-tartomany)
FELHASZNALO   (,echo,proba-tartomany)     (,foxtrott,proba-tartomany) \
              (,golf,proba-tartomany)
OSZTONDIJAS   (,able,proba-tartomany)     (,baker,proba-tartomany)
```

Az `IT_DOLG`, `IT_UJDOLG` stb. a hálózati csoportok nevei lesznek. Minden egyes zárójelezett csoport egy vagy több felhasználói hozzáférést tartalmaz. A csoportokban szereplõ három mezõ a következõ:

1. Azon gépek neve, amelykre a következõ elemek érvényesek. Ha itt nem adunk meg neveket, akkor a bejegyzés az összes gépre vonatkozik. Ha megadjuk egy gép nevét, akkor jutalmunk a teljes sötétség, a rettegetés és totális megtébolyodás.
2. A csoporthoz tartozó hozzáférés neve.

3. A hozzáféréshez kapcsolódó NIS tartomány. A csoportba más NIS tartományokból is át tudunk hozni hozzáféréseket, ha netalán éppen olyan szerencsétlenek lennénk, hogy több NIS tartományt is felügyelnünk kell.

A mezõk mindegyike tartalmazhat dzsókerkaraktereket. Errõl részletesebben a netgroup(5) man oldalon olvashatunk.

A hálózati csoportoknak lehetõleg ne adjunk 8 karakternél hosszabb nevet, különösen abban az esetben, ha a NIS tartományban más operációs rendszereket is használunk. A nevekben eltérnek a kis- és nagybetûk. Ha a hálózati csoportokat nevét nagybetûkkel írjuk, akkor könnyen különbséget tudunk tenni a felhasználók, gépek és hálózati csoportok nevei között.

Egyes (nem FreeBSD alapú) NIS kliensek nem képesek kezelni a nagyon sok bejegyzést tartalmazó hálózati csoportokat. Például a SunOS™ néhány korábbi verziója fennakad rajta, ha egy hálózati csoport 15 *bejegyzésnél* többet tartalmaz. Az ilyen korlátozások alól úgy tudunk kibújni, ha 15 felhasználónként újabb hálózati csoportokat hozunk létre, amelyekkel az eredeti hálózati csoportot építjük fel:

```
NAGYCSP1  (,joe1,tartomany)  (,joe2,tartomany)  (,joe3,tartomany) [...]
NAGYCSP2  (,joe16,tartomany)  (,joe17,tartomany) [...]
NAGYCSP3  (,joe31,tartomany)  (,joe32,tartomany)
NAGYCSOPORT  NAGYCSP1 NAGYCSP2 NAGYCSP3
```

Ugyanez a folyamat javasolt olyan esetekben is, ahol 225 felhasználónál többre lenne szükség egyetlen hálózati csoporton belül.

Az így létrehozott új NIS táblázat szétküldése meglehetõsen könnyû feladat:

```
ellington# cd /var/yp
ellington# make
```

Ez a parancs létrehoz három NIS táblázatot: netgroup, netgroup.byhost és netgroup.byuser. Az ypcat(1) paranccsal ellenõrizni is tudjuk az új NIS táblázatainkat:

```
ellington% ypcat -k netgroup
ellington% ypcat -k netgroup.byhost
ellington% ypcat -k netgroup.byuser
```

Az elsõ parancs kimenete a /var/yp/netgroup állomány tartalmára emlékeztethet minket. A második parancsnak nincs semmilyen kimenete, hacsak nem adtunk meg valamilyen gépfüggõ hálózati csoportot. A harmadik parancs a hálózati csoportokat listázza ki a felhasználókhoz.

A kliensek beállítása tehát nagyon egyszerû. A `haboru` nevû szerver beállításához indítsuk el a vipw(8) programot, és cseréljük a

```
+:::::::::
```

sort erre:

```
+@IT_DOLG:::::::::
```

Innentõl kezdve kizárólag csak az `IT_DOLG` csoportban található felhasználók fognak bekerülni a `haboru` jelszó adatbázisába, és csak ezek a felhasználók tudnak ide bejelentkezni.

Sajnos ez a korlátozás a parancsértelmezõ `~` funkciójára és összes olyan rutinra is vonatkozik, amelyet a felhasználói nevek és azok numerikus azonosító között képez le. Más szóval a `cd ~felhasználó` parancs nem fog mûködni, és az `ls -l` parancs kimenetében a felhasználói nevek helyett csak numerikus azonosítók jelennek meg, továbbá a `find . -user joe -print No such user` (`Nincs ilyen felhasználó`) hibát fog visszaadni. Ez úgy tudjuk megjavítani, ha úgy importáljuk a szerverre az összes felhasználó bejegyzését, hogy *közben tiltjuk a hozzáférésüket*.

Ehhez vegyünk fel egy újabb sort az /etc/master.passwd állományba. A sor valahogy így fog kinézni:

`+:::::::::/sbin/nologin`, amely annyit tesz, hogy "importáljuk az összes bejegyzést, de a hozzájuk tartozó parancsértelmezõ a /sbin/nologin legyen". A `passwd` állományban tetszõleges mezõ tartalmát le tudjuk úgy cserélni, ha megadunk neki egy alapértelmezett értéket az /etc/master.passwd állományban.

Vigyázzunk, hogy a `:::::::::/sbin/nologin` sort az `@IT_DOLG:::::::::` sor után írjuk. Ha nem így teszünk, akkor a NIS-bõl importált összes felhasználói hozzáférés a /sbin/nologin parancsértelmezõt kapja.

Miután elvégeztük ezt a változtatást, minden újabb dolgozó felvétele után csupán egyetlen táblázatot kell megváltoztatnunk. Ugyanezt a taktikát követhetjük a kevésbé fontosabb szerverek esetében is, hogy ha a helyi /etc/master.passwd állományukban a korábbi `+:::::::::` bejegyzést valami ilyesmivel helyettesítjük:

```
+@IT_DOLG:::::::::
+@IT_UJDOLG:::::::::
+:::::::::/sbin/nologin
```

Az egyszerû munkaállomások esetében pedig ezekre a sorokra lesz szükségünk:

```
+@IT_DOLG:::::::::
+@FELHASZNALOK:::::::::
+:::::::::/sbin/nologin
```

Minden remekül üzemel egészen addig, amíg néhány hét múlva ismét változik a házirend: az IT tanszékre ösztöndíjasok érkeznek. Az IT ösztöndíjasai a munkaállomásokat és a kevésbé fontosabb szervereket tudják használni. Az új IT dolgozók már a központi szerverekre is bejelentkezhetnek.

Így tehát létrehozunk egy új hálózati csoportot IT_OSZTONDIJAS néven, majd felvesszük ide az új IT ösztöndíjasokat, és nekilátunk végigzongorázni az összes gép összes konfigurációs állományát... Ahogy azonban egy régi mondás is tartja: "A központosított tervezésben ejtett hibák teljes káoszhoz vezetnek".

A NIS az ilyen helyzeteket úgy igyekszik elkerülni, hogy megengedi újabb hálózati csoportok létrehozását más hálózati csoportokból. Egyik ilyen lehetõség a szerep alapú hálózati csoportok kialakítása. Például, ha a fontosabb szerverek bejelentkezési korlátozásai számára hozzunk létre egy NAGYSRV nevû csoportot, valamint egy másik hálózati csoportot KISSRV néven a kevésbé fontosabb szerverekhez, végül MUNKA néven egy harmadik hálózati csoportot a munkaállomásokhoz. Mindegyik ilyen hálózati csoport tartalmazza azokat a csoportokat, amelyek engedélyezik a gépek elérését. A hálózati csoportok leírását tartalmazó NIS táblázat most valahogy így fog kinézni:

```
NAGYSRV  IT_DOLG IT_UJDOLG
KISSRV   IT_DOLG IT_UJDOLG IT_OSZTONDIJAS
MUNKA    IT_DOLG IT_OSZTONDIJAS FELHASZNALOK
```

A bejelentkezési megszorítások ilyen típusú megadása viszonylag jól mûködik, hogy ha azonos korlátozások alá esõ gépek csoportjait akarjuk felírni. Bánatunk ez a kivétel, és nem a szabály. Az esetek nagy többségében ugyanis a bejelentkezésre vonatkozó korlátozásokat gépenként kell egyesével megadni.

A hálózati csoportok gépfüggõ megadása tehát az iménti házirendhez társuló igények kielégítésének egyik módja. Ebben a forgatókönyvben az /etc/master.passwd állomány minden számítógépen két "+"-os sorral kezdõdik. Közülük az elsõ a gépen engedélyezett hozzáféréseket tartalmazó hálózati csoportra vonatkozik, a második pedig az összes többi hozzáféréshez az /sbin/nologin parancsértelmezõt kapcsolja hozzá. Itt jó ötlet, ha a gép nevének "VÉGIG-NAGYBETûS" változatát adjuk meg a hozzá tartozó hálózati csoport nevének:

```
+@GÉPNÉV:::::::::
+::::::::::/sbin/nologin
```

Miután elvégeztük ezt a feladatot minden egyes gépen, az /etc/master.passwd állomány helyi változatait soha többé nem kell módosítanunk. Az összes többi változtatást a NIS táblázaton keresztül tudjuk keresztül vinni. Íme a felvázolt forgatókönyvhöz tartozó hálózati csoportok kiépítésének egyik lehetséges változata, egy-két finomsággal kiegészítve:

```
# Elõször a felhasználók csoportjait adjuk meg:
IT_DOLG         (,alpha,proba-tartomany)    (,beta,proba-tartomany)
IT_UJDOLG       (,charlie,proba-tartomany)  (,delta,proba-tartomany)
TANSZ1          (,echo,proba-tartomany)     (,foxtrott,proba-tartomany)
TANSZ2          (,golf,proba-taromany)      (,hotel,proba-tartomany)
TANSZ3          (,india,proba-taromany)     (,juliet,proba-tartomany)
IT_OSZTONDIJAS  (,kilo,proba-tartomany)     (,lima,proba-tartomany)
D_OSZTONDIJAS   (,able,proba-tartomany)     (,baker,proba-tartomany)
#
# Most pedig hozzunk létre csoportokat szerepek szerint:
```

```
FELHASZNALOK     TANSZ1   TANSZ2          TANSZ3
NAGYSRV          IT_DOLG  IT_UJDOLG
KISSRV           IT_DOLG  IT_UJDOLG       IT_OSZTONDIJAS
MUNKA            IT_DOLG  IT_OSZTONDIJAS  FELHASZNALOK
#
# Következzenek a speciális feladatokhoz tartozó csoportok:
# Az echo és a golf tudja elérni a vírusvédelemért felelõs gépet:
VEDELEM          IT_DOLG  (,echo,proba-tartomany)  (,golf,proba-tartomany)
#
# Gép alapú hálózati csoportok
# A fõ szervereink:
HABORU      NAGYSRV
EHSEG       NAGYSRV
# Az india nevû felhasználó hozzá szeretné ehhez férni:
SZENNYEZES  NAGYSRV  (,india,proba-tartomany)
#
# Ez valóban fontos és komolyan szabályoznunk kell:
HALAL       IT_DOLG
#
# Az elõbb említett vírusvédelmi gép:
EGY         VEDELEM
#
# Egyetlen felhasználóra korlátozzuk le ezt a gépet:
KETTO       (,hotel,proba-tartomany)
# [...és itt folytatódik a többi csoporttal]
```

Ha a felhasználói hozzáféréseinket valamilyen adatbázisban tároljuk, akkor a táblázat elsõ részét akár az adatbázis lekérdezésein keresztül is elõ tudjuk állítani. Ezzel a módszerrel az új felhasználók automatikusan hozzáférnek a gépekhez.

Legyünk viszont óvatosak: nem mindig javasolt gépeken alapuló hálózati csoportokat készíteni. Ha a hallgatói laborokba egyszerre több tucat vagy akár több száz azonos konfigurációjú gépet telepítünk, akkor a gép alapú csoportok helyett inkább szerep alapú csoportokat építsünk fel, mivel így a NIS táblázatok méretét egy elfogadható méreten tudjuk tartani.

29.4.8. Amit feltétlenül észben kell tartanunk

Még mindig akad néhány olyan dolog, amit másképpen kell csinálnunk azután, hogy most már NIS környezetben vagyunk.

- Amikor egy új felhasználót akarunk felvenni a laborba, akkor *csak* a központi NIS szerverre kell felvennünk, és *újra kell generáltatnunk a NIS táblázatokat*. Ha ezt elfelejtjük megtenni, akkor az új felhasználó a központi NIS szerveren kívül sehova sem lesz képes bejelentkezni. Például, ha fel akarjuk venni a `jsmith` nevû felhasználót a laborba, akkor ezt kell tennünk:

  ```
  # pw useradd jsmith
  # cd /var/yp
  # make proba-tartomany
  ```

Vagy a `pw useradd jsmith` parancs helyett az `adduser jsmith` parancsot is használhatjuk.

- *A rendszergazdai szintû hozzáféréseket ne tároljuk a NIS táblázatokban.* Olyan gépekre egyáltalán ne is küldjünk olyan karbantartáshoz használt hozzáféréseket, amelynek a felhasználói hivatalosan nem is férhetnének hozzájuk.
- *A központi NIS szervert és az alárendelt szervereket óvjuk minél jobban, és igyekezzünk minimalizálni a kieséseiket.* Ha valaki feltöri vagy egyszerûen csak kikapcsolja ezeket a gépeket, akkor ezzel lényegében mindenkit megakadályoz abban, hogy be tudjon jelentkezni a laborban.

  Ezek a központosított vezérlésû rendszerek legfõbb gyengeségei. Ha nem védjük kellõen a NIS szervereinket, akkor azzal nagyon ellenséget szerezhetünk magunknak!

29.4.9. Kompatibilitás a NIS elsõ változatával

A FreeBSD-ben megtalálható ypserv szolgáltatás valamennyire képes ellátni a NIS elsõ változatát használó klienseket is. A FreeBSD NIS implementációja csak a NIS v2 protokollt használja, azonban mivel más implementációk kompatibilisek kívánnak maradni a régebbi rendszerekkel, ismerik a v1 protokollt is. Az ilyen rendszerekhez tartozó ypbind démonok még olyankor is megpróbálnak v1-es NIS szerverekhez kötést létrehozni, amikor valójában nincs is rá szükségük (és gyakran még akkor is ilyet keresnek, amikor az üzenetükre már válaszolt egy v2-es szerver). Hozzátennénk, hogy bár az ypserver ezen változata a normál klienshívásokat képes feldolgozni, a táblázatokat már nem tudja átküldeni a v1-es klienseknek. Ebbõl következik, hogy a központi vagy alárendelt szerverek nem tudnak együttmûködni olyan NIS szerverekkel, amelyek csak a v1-es protokollt beszélik. Szerencsére ilyen szervereket manapság már alig használnak.

29.4.10. NIS szerverek, melyek egyben NIS kliensek

Óvatosan kell bánnunk az ypserv elindításával olyan többszerveres tartományokban, ahol a szerverek maguk is NIS kliensek. Alapvetõen nincs abban semmi kivetnivaló, ha a szervereket saját magukhoz kötjük ahelyett, hogy engednénk nekik a kötési kérések küldését és így egymáshoz kötnénk ezeket. Különös hibák tudnak származni olyan helyzetekben, amikor az egyik szerver leáll, miközben a többiek pedig függenek tõle. Végül is ilyenkor minden kliens szépen kivárja a szükséges idõt, aztán megpróbál más szerverekhez kötõdni, de az itt fellépõ késlekedés jelentõs mennyiségû lehet, és ez a hibajelenség ismét fennállhat, mivel elõfordulhat, hogy a szerverek megint egymáshoz kapcsolódnak.

A klienst úgy tudjuk egy adott szerverhez kötni, ha az `ypbind` parancsot a `-S` beállítással indítjuk. Ha mindezt nem akarjuk manuálisan megtenni a NIS szerver minden egyes újraindításakor, akkor vegyük fel a következõ sorokat az /etc/rc.conf állományba:

```
nis_client_enable="YES" # elindítjuk a klienst is
nis_client_flags="-S NIS tartomány,szerver"
```

Részletesebb lásd az ypbind(8) man oldalát.

29.4.11. A jelszavak formátuma

A NIS rendszerek kiépítése során az emberek leggyakrabban a jelszavak formátumával kapcsolatban tapasztalnak nehézségeket. Ha a szerverünk DES titkosítású jelszavakat használ, akkor csak olyan klienseket fog tudni támogatni, amelyek szintén így kódolják ezeket. Például, ha a hálózaton vannak Solaris™ rendszerû NIS klienseink, akkor szinte biztos, hogy DES titkosítást kell használnunk.

A szerverek és a kliensek által használt formátumokat az /etc/login.conf állományba tekintve deríthetjük ki. Ha a gépek többségén a DES titkosítást látjuk, akkor a `default` osztálynak egy ilyen bejegyzést kell tartalmaznia:

```
default:\
    :passwd_format=des:\
    [a többit most nem mutatjuk]
```

A `passwd_format` tulajdonság további lehetséges értékei lehetnek a `blf` és az `md5` (melyek rendre a Blowfish és MD5 titkosítású jelszavakat adják meg).

Ha változtattunk valamit az /etc/login.conf állományban, akkor a bejelentkezési tulajdonságok adatbázisát is újra kell generálni, melyet `root` felhasználóként a következõ módon tehetünk meg:

```
# cap_mkdb /etc/login.conf
```

Az /etc/master.passwd állományban jelenlevõ jelszavak formátuma azonban nem frissítõdik egészen addig, amíg a felhasználók a bejelentkezési adatbázis újragenerálása *után* meg nem változtatják a jelszavaikat.

Úgy tudjuk még biztosítani, hogy a jelszavak megfelelõ formátumban kódolódjanak, ha az /etc/auth.conf állományban megkeressük a `crypt_default` sort, amelyben a választható jelszóformátumok felhasználási sorrendjét találhatjuk meg. Itt tehát mindössze annyit kell tennünk, hogy a kiszemelt formátumot a lista elejére tesszük. Például, ha a DES titkosítású jelszavakat akarunk használni, akkor ez a bejegyzés így fog kinézni:

```
crypt_default   =   des blf md5
```

Ha a fenti lépéseket követjük az összes FreeBSD alapú NIS szervernél és kliensnél, akkor biztosra mehetünk abban, hogy a hálózatunkon belül ugyanazt a jelszóformátumot fogják használni. Ha gondunk akadna a NIS kliensek hitelesítésével, akkor itt érdemes kezdeni a hiba felderítését. Ne felejtsük: ha egy NIS szervert egy heterogén hálózatba akarunk telepíteni, akkor valószínûleg az összes rendszeren a DES titkosítást kell választani, mivel általában ez a közös nevezõ ebben a tekintetben.

29.5. A hálózat automatikus beállítása (DHCP)

29.5.1. Mi az a DHCP?

A Dinamikus állomáskonfigurációs protokoll, avagy Dynamic Host Configuration Protocol (DHCP) annak eszközeit írja le, hogy egy rendszer miként tud csatlakozni egy hálózathoz és miként tudja azon belül megszerezni a kommunikációhoz szükséges információkat. A FreeBSD 6.0 elõtti változatai az ISC (Internet Systems Consortium, vagyis az internetes rendszerkonzorcium) által kidolgozott DHCP kliens (dhclient(8)) implementációját tartalmazzák. A késõbbi verziókban pedig az OpenBSD 3.7 verziójából átvett `dhclient` paranccsal dolgozhatunk. Ebben a szakaszban a `dhclient` parancsra vonatkozó összes információ egyaránt érvényes az ISC és az OpenBSD által fejlesztett DHCP kliensekre. A DHCP szerver az ISC-tõl származik.

29.5.2. Mivel foglalkozik ez a szakasz

Ebben a szakaszban az ISC és az OpenBSD DHCP klienseinek kliens- és szerver oldali komponsenseit mutatjuk be. A kliens oldali program neve a `dhclient`, amely a FreeBSD részeként érkezik, és a szerver oldali elem pedig a net/isc-dhcp31-server porton keresztül érhetõ el. A lentebb említett hivatkozások mellett a témában még a dhclient(8), dhcp-options(5) és a dhclient.conf(5) man adhatnak bõvebb felvilágosítást a témában.

29.5.3. Ahogyan mûködik

Amikor a `dhclient`, vagyis a DHCP kliens elindul egy kliensgépen, akkor a hálózaton üzenetszórással próbálja meg elkérni a konfigurációjához szükséges adatokat. Alapértelmezés szerint ezek a kérések a 68-as UDP porton keresztül mennek. A szerver ezekre a 67-es UDP porton válaszol, ahol visszaad a kliensnek egy IP-címet és a hálózat használatához szükséges további információkat, mint például a hálózati maszkot, az alapértelmezett átjáró és a névfeloldásért felelõs szerverek címét. Az összes ilyen jellegû adat egy DHCP "bérlet" (lease) formájában érkezik meg, amely csak egy adott ideig érvényes (ezt a DHCP szerver karbantartója állítja be). Így a hálózaton a kliens nélküli IP-címeket egy idõ után automatikusan visszanyerjük.

A DHCP kliensek rengeteg információt képes elkérni a szervertõl. Ezek teljes listáját a dhcp-options(5) man oldalán olvashatjuk el.

29.5.4. Használat a FreeBSD-n belül

A FreeBSD teljes egészében tartalmazza az ISC vagy az OpenBSD DHCP kliensét, a `dhclient` programot (attól függõen, hogy a FreeBSD melyik változatát használjuk). A DHCP kliensek támogatása a telepítõben és az alaprendszerben is megtalálható, és ezzel mentesülünk minden konkrét hálózati beállítás alól a DHCP szervereket alkalmazó hálózatokon. A `dhclient` a FreeBSD 3.2 változata óta megtalálható a rendszerben.

DHCP használatát a sysinstall is lehetõvé teszi. Amikor egy hálózati felületet a sysinstall programon belül állítunk be, akkor a második kérdés mindig ez szokott lenni: "Do you want to try DHCP configuration of the interface?" ("Megpróbáljuk DHCP használatával beállítani a felületet?") Ha erre igennel válaszolunk, akkor azzal lényegében a `dhclient` parancsot indítjuk el, és ha mindez sikerrel zárul, akkor szinte magától kitöltõdik az összes hálózati beállításunk.

A DHCP használatához két dolgot kell beállítanunk a rendszerünkön:

- Gondoskodjunk róla, hogy a bpf eszköz része a rendszermagunknak. Ha még nem lenne benne, akkor a rendszermag beállításait tartalmazó állományba vegyük fel a `device bpf` sort és fordítsuk újra a rendszermagot. A rendszermagok fordításáról a A FreeBSD rendszermag testreszabásaben tudhatunk meg többet.

  A bpf eszköz alapból megtalálható a GENERIC rendszermagokban, így ha ezt használjuk, akkor nem kell saját verziót készítenünk a DHCP használatához.

  Azok számára viszont, akik biztonsági szempontból aggódnak a rendszerük miatt, meg kell említenünk, hogy a bpf egyben az az eszköz, amely a csomagok lehallgatását is lehetővé teszi (habár az ilyeneket `root` felhasználóként lehet csak elindítani). A bpf*kell* a DHCP használatához, azonban ha nagyon fontos nekünk a rendszerünk biztonsága, akkor a bpf eszközt érdemes kivennünk a rendszermagból, ha még pillanatnyilag nem használunk ilyet.

- Az /etc/rc.conf állományunkat az alábbiak szerint kell módosítani:

  ```
  ifconfig_fxp0="DHCP"
  ```

  Az `fxp0` eszközt ne felejtsük el kicserélni arra a felületre, amelyet automatikusan akarunk beállítani. Ennek mikéntje a A hálózati kártyák beállításaban olvasható.

  Ha a `dhclient` a rendszerünkben máshol található, vagy egyszerűen csak további beállításokat akarunk átadni a `dhclient` parancsnak, akkor adjuk meg a következőt is (változtassuk meg igényeink szerint):

  ```
  dhclient_program="/sbin/dhclient"
  dhclient_flags=""
  ```

A DHCP szerver, a dhcpd a net/isc-dhcp31-server port részeként érhető el. Az a port tartalmazza az ISC DHCP szerverét és a hozzá tartozó dokumentációt.

29.5.5. Állományok

- /etc/dhclient.conf

  A `dhclient` működéséhez szükség lesz egy konfigurációs állományra, aminek a neve /etc/dhclient.conf. Ez az állomány általában csak megjegyzéseket tartalmaz, mivel az alapértelmezett értékek többnyire megfelelőek. Ezt a konfigurációs állományt a dhclient.conf(5) man oldal írja le.

- /sbin/dhclient

  A `dhclient` statikusan linkelt és az /sbin könyvtárban található. A dhclient(8) man oldal tud róla

részletesebb felvilágosítást adni.

- /sbin/dhclient-script

  A `dhclient-script` a FreeBSD-ben levõ DHCP kliens konfigurációs szkriptje. Mûködését a dhclient-script(8) man oldal írja le, de a felhasználók részérõl semmilyen módosítást nem igényel.

- /var/db/dhclient.leases

  A DHCP kliens az érvényes bérleteket tartja nyilván ezekben az állományban és naplóként használja. A dhclient.leases(5) man oldal ezt valamivel bõvebben kifejti.

29.5.6. További olvasnivalók

A DHCP protokoll mûködését az RFC 2131 mutatja be. A témához kapcsolódóan itt tudunk még leírásokat találni.

29.5.7. A DHCP szerverek telepítése és beállítása

Mirõl szól ez a szakasz

Ebben a szakaszban arról olvashatunk, hogy miként kell egy FreeBSD típusú rendszert DHCP szervernek beállítani, ha az ISC (internetes rendszerkonzorcium) DHCP szerverét használjuk.

Ez a szerver nem része a FreeBSD-nek, ezért a szolgáltatás elindításához elõször fel kell raknunk a net/isc-dhcp31-server portot. A Portgyûjtemény használatára vonatkozóan a Alkalmazások telepítése. csomagok és portok lehet segítségünkre.

A DHCP szerver telepítése

Ha a FreeBSD rendszerünket DHCP szerverként akarjuk beállítani, akkor ehhez elsõként a bpf(4) eszköz jelenlétét kell biztosítani a rendszermagban. Ehhez vegyük fel a `device bpf` sort a rendszermagunk beállításait tartalmazó állományba, majd fordítsuk újra a rendszermagot. A rendszermag lefordításáról a A FreeBSD rendszermag testreszabásaben olvashatunk.

A bpf eszköz a FreeBSD-hez alapból adott GENERIC rendszermag része, ezért a DHCP használatához nem kell feltétlenül újat fordítanunk.

A biztonsági szempontok miatt aggódó felhasználók részére megjegyezzük, hogy a bpf eszköz egyben a csomagok lehallgatását is lehetõvé teszi (habár az ilyen témájú programok futtatásához megfelelõ jogokra is szükség van). A bpf használata *kötelezõ* a DHCP mûködtetéséhez, de ha nagyon kényesek vagyunk a biztonságot illetõen, akkor minden olyan esetben, amikor nem használjuk ki ezt a lehetõséget, távolítsuk el a rendszermagból.

A következõ lépésben át kell szerkesztenünk a mintaként mellékelt dhcpd.conf állományt, amelyet a net/isc-dhcp31-server port rakott fel. Ez alapértelmezés szerint a /usr/local/etc/dhcpd.conf.sample néven található meg, és mielõtt bármit is változtatnánk rajta, másoljuk le /usr/local/etc/dhcpd.conf néven.

A DHCP szerver beállítása

A dhcpd.conf az alhálózatokat illetve a gépeket érintõ deklarációkat tartalmazza, és talán a legkönnyebben a következõ példa alapján mutatható be:

```
option domain-name "minta.com";①
option domain-name-servers 192.168.4.100;②
option subnet-mask 255.255.255.0;③

default-lease-time 3600;④
max-lease-time 86400;⑤
ddns-update-style none;⑥

subnet 192.168.4.0 netmask 255.255.255.0 {
  range 192.168.4.129 192.168.4.254;⑦
  option routers 192.168.4.1;⑧
}

host mailhost {
  hardware ethernet 02:03:04:05:06:07;⑨
  fixed-address levelezes.minta.com;⑩
}
```

① Ez a beállítás adja meg a kliensek számára az alapértelmezett keresési tartományt (search domain). A resolv.conf(5) tud ezzel kapcsolatban részletesebb információkat adni.

② Ez a beállítás adja meg a kliensek által használt névfeloldó szerverek vesszõvel elválasztott felsorolását.

③ A kliensekhez tartozó hálózati maszk.

④ A kliens egy adott idõre kérhet bérleti jogot, egyébként a szerver dönt a bérlet lejárati idejérõl (másodpercekben).

⑤ Ez az a maximális idõ, amennyire a szerver hajlandó bérbe adni IP-címet. A kliens ugyan hosszabb idõre is kérheti és meg is kapja, de legfeljebb csak `max-lease-time` másodpercig lesz érvényes.

⑥ Ez a beállítás határozza meg, hogy a DHCP szervernek frissítse-e a névoldási információkat a bérlések elfogadásánál vagy visszamondásánál. Az ISC implementációjánál ez a beállítás *kötelezõ*.

⑦ Ezzel adjuk meg milyen tartományból tudunk IP-címeket kiosztani a kliensek számára. A kezdõ címet is beleértve, innen fogunk kiutalni egyet a klienseknek.

⑧ A kliensek felé elküldött alapértelmezett átjáró címe.

⑨ A gép hardveres MAC-címe (így a DHCP szerver képes felismerni a kérés küldõjét).

⑩ Ennek megadásával a gépek mindig ugyanazt az IP-címet kapják. Itt már megadhatunk egy hálózati nevet, mivel a bérlethez tartozó információk visszaküldése elõtt maga a DHCP szerver fogja feloldani a gép nevét.

Miután befejeztük a dhcpd.conf módosítását, a DHCP szerver az /etc/rc.conf állományban tudjuk

engedélyezni, vagyis tegyük bele a következõt:

```
dhcpd_enable="YES"
dhcpd_ifaces="dc0"
```

A `dc0` felület nevét helyettesítsük annak a felületnek (vagy whitespace karakterekkel elválasztott felületeknek) a nevével, amelyen keresztül a DHCP szerver várni fogja a kliensek kéréseit.

Ezután a következõ parancs kiadásával indítsuk el a szervert:

```
# /usr/local/etc/rc.d/isc-dhcpd start
```

Amikor a jövõben valamit változtatunk a konfigurációs állományon, akkor ezzel kapcsolatban fontos megemlíteni, hogy ha csak egy `SIGHUP` jelzést küldünk a dhcpd démonnak, akkor az a többi démontól eltérõen önmagában még *nem* eredményezi a konfigurációs adatok újraolvasását. Helyette a `SIGTERM` jelzéssel kell leállítani a programot, majd újraindítani a fenti paranccsal.

Állományok

- /usr/local/sbin/dhcpd

  A dhcpd statikusan linkelt és a /usr/local/sbin könyvtárban található. A porttal együtt felkerülõ dhcpd(8) man oldal ad részletesebb útmutatást dhcpd használatáról.

- /usr/local/etc/dhcpd.conf

  Mielõtt a dhcpd megkezdhetné mûködését, egy konfigurációs állományra is szükségünk lesz, amely a /usr/local/etc/dhcpd.conf. Ez az állomány tartalmazza az összes olyan információt, ami kell a kliensek megfelelõ kiszolgálásához valamint a szerver mûködéséhez. Ez a konfigurációs állomány porthoz tartozó dhcpd.conf(5) man oldalon kerül ismertetésre.

- /var/db/dhcpd.leases

  A DHCP szerver ebben az állományba tartja nyilván a kiadott bérleteket, egy napló formájában. A porthoz kapcsolódó dhcpd.leases(5) man oldalon errõl többet is megtudhatunk.

- /usr/local/sbin/dhcrelay

  A dhcrelay állománynak olyan komolyabb környezetekben van szerepe, ahol a DHCP szerver a kliensektõl érkezõ kéréseket egy másik hálózaton található DHCP szerverhez továbbítja. Ha szükség lenne erre a lehetõségre, akkor telepítsük fel a net/isc-dhcp31-relay portot. A porthoz tartozó dhcrelay(8) man oldal ennek részleteit taglalja.

29.6. Névfeloldás (DNS)

29.6.1. Áttekintés

A FreeBSD alapértelmezés szerint a BIND (Berkeley Internet Name Domain) egyik verzióját

tartalmazza, amely a névfeloldási (Domain Name System, DNS) protokoll egyik elterjedt implementációja. A DNS protokollon keresztül tudunk az IP-címekhez neveket rendelni és fordítva. Például a `www.FreeBSD.org` névre a FreeBSD Projekt webszerverének IP-címét kapjuk meg, miközben a `ftp.FreeBSD.org` pedig a hozzá tartozó FTP szerver IP-címét fogja visszaadni. Ehhez hasonlóan a fordítottja is megtörténhet, vagyis egy IP-címhez is kérhetjük a hálózati név feloldását. A névfeloldási kérések kiszolgálásához nem feltétlenül szükséges névszervert futtatni a rendszerünkön.

A FreeBSD jelen pillanatban alapból a BIND9 névszervert tartalmazza. A benne szereplõ változata több biztonsági javítást, új állományrendszeri kiosztást és automatizált chroot(8) beállítást is magában foglal.

Az interneten keresztüli névfeloldást legfelsõ szintû tartományoknak (Top Level Domain, TLD) nevezett hitelesített tövek némileg bonyolult rendszerén alapszik, valamint más egyéb olyan névszervereken, amelyek további egyéni információkat tárolnak és táraznak.

A BIND fejlesztését jelenleg az Internet Systems Consortium (http://www.isc.org/) felügyeli.

29.6.2. Alapfogalmak

A leírás megértéséhez be kell mutatnunk néhány névfeloldással kapcsolatos fogalmat.

Fogalom	Meghatározás
Közvetlen névfeloldás (forward DNS)	A hálózati nevek leképezése IP-címekre.
õs (origin)	Egy adott zóna állományban szereplõ tartományra vonatkozik.
named, BIND	A FreeBSD-n belüli BIND névszerver különbözõ megnevezései.
Névfeloldó (resolver)	Az a program a rendszerben, amelyhez a hálózaton levõ gépek a zónák adatainak elérésével kapcsolatban fordulnak.
Inverz névfeloldás (reverse DNS)	Az IP-címek leképzése hálózati nevekre.
Gyökérzóna (root zone)	Az interneten található zónák hierarchiájának töve. Minden zóna ebbe a gyökérzónába esik, ahhoz hasonlóan, ahogy egy állományrendszerben az állományok a gyökérkönyvtárba.
Zóna (zone)	Egy különálló tartomány, altartomány vagy a névfeloldás azon része, amelyet egyazon fennhatóság alatt tartanak karban.

Példák zónákra:

- A gyökérzónára a leírásokban általában `.` néven szoktak hivatkozni.
- A `org.` egy legfelsõ szintû tartomány (TLD) a gyökérzónán belül.

- A `minta.org.` a `org.` TLD tartomány alatti zóna.
- A `1.168.192.in-addr.arpa` egy olyan zóna, amelyek a `192.168.1.*` IP-címtartományban szereplõ összes címet jelöli.

Mint láthatjuk, a hálózati nevek balról kiegészülve pontosodnak. Tehát például a `minta.org.` sokkal pontosabb meghatározás, mint a `org.`, ahogy az `org.` magánál a gyökérzónánál jelent többet. A hálózati nevek felosztása leginkább egy állományrendszerhez hasonlítható, például a /dev könyvtár a gyökéren belül található, és így tovább.

29.6.3. Miért érdemes névszervert futtatni

A névszerverek általában két alakban jelennek meg. Egyikük a hitelesített névszerver, a másikuk a gyorsítótárazó névszerver.

Egy hitelesített névszerverre akkor van szükségünk, ha:

- a világ többi része felé akarunk hiteles névfeloldási információkat szolgáltatni;
- regisztráltunk egy tartományt (például `minta.org`) és az alatta levõ hálózati nevekhez is szeretnénk IP-címeket rendeltetni;
- a IP-címtartományunkban szükség van inverz névfeloldási bejegyzésekre (amely IP-címbõl ad meg hálózati nevet) is;
- a kérések teljesítéséhez egy tartalék avagy második, alárendelt (slave) névszerver kell.

A gyorsítótárazó névszerverre akkor van szükségünk, ha:

- egy helyi névfeloldó szerver felhasználásával fel akarjuk gyorsítani az egyébként a külsõ névszerver felé irányuló kérések kiszolgálását.

Amikor valaki lekérdezi a `www.FreeBSD.org` címét, akkor a névfeloldó elõször általában a kapcsolatot rendelkezésre bocsátó internet-szolgáltató névszerverét kérdezi meg és onnan kapja meg a választ. Egy helyi, gyorsítótárazó névszerver használata esetén azonban egy ilyen kérést csak egyszer kell kiadni a külsõ névszervernek. Ezután már minden további ilyen kérés el sem hagyja a belsõ hálózatunkat, mivel a válasz szerepel a gyorsítótárban.

29.6.4. Ahogyan mûködik

FreeBSD alatt a BIND démon nyilvánvaló okokból named néven érhetõ el.

Állomány	Leírás
named(8)	A BIND démon.
rndc(8)	A névszervert vezérlõ segédprogram.
/etc/namedb	A BIND által kezelt zónák adatait tároló könyvtár.
/etc/namedb/named.conf	A démon konfigurációs állománya.

Attól függõen, hogy miként állítjuk be az adott zónát a szerveren, a hozzá tartozó állományok a

/etc/namedb könyvtáron belül a master, slave vagy dynamic alkönyvtárban foglalnak helyet. Az itt tárolt állományokban levõ névfeloldási információk alapján válaszol a névszerver a felé intézett kérésekre.

29.6.5. A BIND elindítása

Mivel a BIND alapból elérhetõ a rendszerben, viszonylag könnyen be tudjuk állítani.

A named alapértelmezett beállítása szerint egy chroot(8) környezetben futó egyszerû névfeloldást végzõ szerver, amely a helyi IPv4 interfészen (127.0.0.1) fogadja a kéréseket. Ezzel a beállítással a következõ parancson keresztül tudjuk elindítani:

```
# /etc/rc.d/named onestart
```

Ha engedélyezni akarjuk a named démont minden egyes rendszerindításkor, tegyük a következõ sort az /etc/rc.conf állományba:

```
named_enable="YES"
```

Értelemszerûen az /etc/namedb/named.conf tele van olyan beállítási lehetõségekkel, amelyek meghaladják ennek a leírásnak a kereteit. Ha viszont kíváncsiak vagyunk a FreeBSD-ben a named indításához használt beállításokra, akkor az /etc/defaults/rc.conf állományban nézzük meg `named_*` változókat és olvassuk át az rc.conf(5) man oldalt. Emellett még a Az rc használata FreeBSD alattt is hasznos lehet elolvasni.

29.6.6. A konfigurációs állományok

A named beállításait tartalmazó állományok pillanatnyilag az /etc/namedb könyvtárban találhatóak és hacsak nem egy egyszerû névfeloldóra tartunk igényt, akkor a használata elõtt módosítanunk is kell. Itt ejtjük meg a beállítások nagy részét.

/etc/namedb/named.conf

```
// $FreeBSD$
//
// Részletesebb leírást a named.conf(5) és named(8) man oldalakon, valamint
// a /usr/shared/doc/bind9 könyvtárban találhatunk.
//
// Ha egy hitelesített szervert akarunk beállítani, akkor igyekezzünk
// a névfeloldás összes finom részletével pontosan tisztában lenni.
// Ugyanis még a legkisebb hibákkal is egyrészt elvághatunk gépeket az
// internet-lérésétõl, vagy másrészt felesleges forgalmat tudunk
// generálni
//

options {
    // A chroot könyvtárhoz relatív elérési út, amennyiben létezik
```

```
    directory   "/etc/namedb";
    pid-file    "/var/run/named/pid";
    dump-file   "/var/dump/named_dump.db";
    statistics-file "/var/stats/named.stats";

// Ha a named démont csak helyi névfeloldóként használjuk, akkor ez
// egy biztonságos alapbeállítás. Ha viszont a named démon az egész
// hálózatunkat is kiszolgálja, akkor ezt a beállítást tegyük
// megjegyzésbe, vagy adjunk meg egy rendes IP-címet, esetleg
// töröljük ki.
    listen-on   { 127.0.0.1; };

// Ha rendszerünkön engedélyezett az IPv6 használata, akkor a helyi
// névfeloldó használatához ezt a sort vegyük ki a megjegyzésbõl.
// A hálózatunk többi részérõl pedig úgy lehet elérni, ha itt megadunk
// egy IPv6 címet, vagy az "any" kulcsszót.
//  listen-on-v6    { ::1; };

// Az alábbi zónákat már a lentebb található üres zónák eleve lefedik.
// Ha tehát a lenti üres zónákat kivesszük a konfigurációból, akkor
// ezeket a sorokat is tegyük megjegyzésbe.
    disable-empty-zone "255.255.255.255.IN-ADDR.ARPA";
    disable-empty-zone
"0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.IP6.ARPA";
    disable-empty-zone
"1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.IP6.ARPA";

// Ha a szolgáltatónk névszervert is elérhetõvé tett számunkra, akkor
// itt adjuk meg annak az IP-címét és engedélyezzük az alábbi sort.
// Ezzel egyben kihasználjuk a gyorsítótárat is, így mérsékeljük az
// internet felé mozgó névfeloldásokat.
/*
    forwarders {
        127.0.0.1;
    };
*

// Ha a 'forwarders' rész nem üres, akkor alapértelmezés szerint a
// 'forward first' értékkel rendelkezik.  Ekkor a kérést a helyi szerver
// kapja abban az esetben, amikor a 'forwarders' részben megadott
// szerverek nem tudják megválaszolni.  Emellett a névszerverben a
// következõ sor hozzáadásával letilthatjuk, hogy önmagától ne
// kezdeményezzen kéréseket:
//     forward only;

// Ha a kérések továbbítását az /etc/resolv.conf állományban megadott
// bejegyzések mentén szeretnénk automatikusan konfigurálni, akkor vegyük
// ki a megjegyzésbõl az alábbi sort és adjuk hozzá az /etc/rc.conf
// állományhoz a name_auto_forward=yes sort.  Emellett használható még a
// named_auto_forward_only beállítás is (amely fentebb leírt funkciót
// valósítja meg).
```

```
//     include "/etc/namedb/auto_forward.conf";
```

Ahogy arról a megjegyzésekben is szó esik, úgy tudjuk aktiválni a gyorsítótárat, ha megadjuk a `forwarders` beállítást. Normális körülmények között a névszerver az interneten az egyes névszervereket rekurzívan fogja keresni egészen addig, amíg meg nem találja a keresett választ. Az iménti beállítás engedélyezésével azonban elõször a szolgáltató névszerverét (vagy az általa kijelölt névszervert) fogjuk megkérdezni, a saját gyorsítótárából. Ha a szolgáltató kérdéses névszervere egy gyakran használt, gyors névszerver, akkor ezt érdemes bekapcsolnunk.

Itt a `127.0.0.1` megadása *nem* mûködik. Mindenképpen írjuk át a szolgáltatónk névszerverének IP-címére.

```
    /*
       A BIND legújabb változataiban alapértelmezés szerint minden egyes
       kimenõ kérésnél más, véletlenszerûen választott UDP portot
       használnak, ezáltal jelentõs mértékben csökkenthetõ a gyorsítótár
       meghamisíthatóságának (cache poisoning) esélye.  Javasoljuk
       mindenkinek, hogy használják ki ezt a lehetõséget és eszerint
       állítsák be a tûzfalakat.

       Ha nem sikerül a tûzfalat hozzáigazítani ehhez a
       viselkedéshez AKKOR ÉS CSAK IS AKKOR engedélyezzük a lenti
       beállítást.  Alkalmazásával sokkal kevésbé lesz ellenálló a
       névszerver a különbözõ hamisítási kísérletekkel szemben,
       ezért lehetõség szerint kerüljük el.

       Az NNNNN helyére egy 49160 és 65530 közti számot kell
       beírnunk.
     */
     // query-source address * port NNNNN;
};

// Ha engedélyezzük a helyi névszervert, akkor az /etc/resolv.conf
// állományban elsõ helyen megadni a 127.0.0.1 címet. Sõt, az
// /etc/rc.conf állományból se felejtsük ki.

// A hagyományos "root-hints" megoldás.  Használjuk ezt VAGY a lentebb
// megadott alárendelt zónákat.
zone "." { type hint; file "named.root"; };

/*  Több szempontból is elõnyös, ha a következõ zónákat alárendeljük a
    gyökér névfeloldó szervereknek:
    1. A helyi felhasználók kéréseit gyorsabban tudjuk feloldalni.
    2. A gyökérszerverek felé nem megy semmilyen hamis forgalom.
    3. A gyökérszerverek meghibásodása vagy elosztott DoS támadás
       esetén rugalmasabban tudunk reagálni.

    Másfelöl azonban ez a módszer a "hints" állomány alkalmazásával
    szemben több felügyeletet igényel, mivel figyelnünk kell, nehogy
```

```
    egy váratlan meghibásodás mûködésképtelenné tegye a
    szerverünket.  Ez a megoldás leginkább a sok klienst kiszolgáló
    névszerverek esetén bizonyulhat jövedelmezõbbnek.  Óvatosan
    bánjunk vele!

    A módszer alkalmazásához vegyük ki a megjegyzésbõl a következõ
    bejegyzéseket és tegyük megjegyzésbe a fenti hint zónát.
*/

zone "." {
    type slave;
    file "slave/root.slave";
    masters {
        192.5.5.241;    // F.ROOT-SERVERS.NET.
    };
    notify no;
};

zone "arpa" {
    type slave;
    file "slave/arpa.slave";
    masters {
        192.5.5.241;    // F.ROOT-SERVERS.NET.
    };
    notify no;
}

zone "in-addr.arpa" {
    type slave;
    file "slave/in-addr.arpa.slave";
    masters {
        192.5.5.241;    // F.ROOT-SERVERS.NET.
    };
    notify no;
};
*/

/*  Az alábbi zónák helyi kiszolgálásával meg tudjuk akadályozni, hogy
    a belõlük indított kérések elhagyják a hálózatunkat és a elérjük
    a gyökér névfeloldó szervereket.  Ez a megközelítés két komoly
    elõnnyel rendelkezik:
    1. A helyi felhasználók kéréseit gyorsabban tudjuk
       megválaszolni.
    2. A gyökérszerverek felé nem továbbítódik semmilyen hamis
       forgalom.
*/
// RFC 1912
zone "localhost"    { type master; file "master/localhost-forward.db"; };
zone "127.in-addr.arpa" { type master; file "master/localhost-reverse.db"; };
zone "255.in-addr.arpa" { type master; file "master/empty.db"; };
```

```
// A helyi IPv6 címek részére létrehozott RFC 1912-szerû zóna
zone "0.ip6.arpa"   { type master; file "master/localhost-reverse.db"; };

// "Ez" a hálózat (RFC 1912 és 3330)
zone "0.in-addr.arpa"       { type master; file "master/empty.db"; };

// Magáncélú hálózatok (RFC 1918)
zone "10.in-addr.arpa"      { type master; file "master/empty.db"; };
zone "16.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "17.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "18.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "19.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "20.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "21.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "22.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "23.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "24.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "25.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "26.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "27.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "28.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "29.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "30.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "31.172.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "168.192.in-addr.arpa" { type master; file "master/empty.db"; };

// Helyi link/APIPA (RFC 3330 és 3927)
zone "254.169.in-addr.arpa" { type master; file "master/empty.db"; };

// Dokumentációs próbahálózat (RFC 3330)
zone "2.0.192.in-addr.arpa" { type master; file "master/empty.db"; };

// Útválasztási teljesítmény tesztelésére (RFC 3330)
zone "18.198.in-addr.arpa"  { type master; file "master/empty.db"; };
zone "19.198.in-addr.arpa"  { type master; file "master/empty.db"; };

// Az IANA részére fentartott - a régi E osztályú címtér
zone "240.in-addr.arpa"     { type master; file "master/empty.db"; };
zone "241.in-addr.arpa"     { type master; file "master/empty.db"; };
zone "242.in-addr.arpa"     { type master; file "master/empty.db"; };
zone "243.in-addr.arpa"     { type master; file "master/empty.db"; };
zone "244.in-addr.arpa"     { type master; file "master/empty.db"; };
zone "245.in-addr.arpa"     { type master; file "master/empty.db"; };
zone "246.in-addr.arpa"     { type master; file "master/empty.db"; };
zone "247.in-addr.arpa"     { type master; file "master/empty.db"; };
zone "248.in-addr.arpa"     { type master; file "master/empty.db"; };
zone "249.in-addr.arpa"     { type master; file "master/empty.db"; };
zone "250.in-addr.arpa"     { type master; file "master/empty.db"; };
zone "251.in-addr.arpa"     { type master; file "master/empty.db"; };
zone "252.in-addr.arpa"     { type master; file "master/empty.db"; };
zone "253.in-addr.arpa"     { type master; file "master/empty.db"; };
```

```
zone "254.in-addr.arpa"     { type master; file "master/empty.db"; };

// Hozzárendelés nélküli IPv6-címek (RFC 4291)
zone "1.ip6.arpa"       { type master; file "master/empty.db"; };
zone "3.ip6.arpa"       { type master; file "master/empty.db"; };
zone "4.ip6.arpa"       { type master; file "master/empty.db"; };
zone "5.ip6.arpa"       { type master; file "master/empty.db"; };
zone "6.ip6.arpa"       { type master; file "master/empty.db"; };
zone "7.ip6.arpa"       { type master; file "master/empty.db"; };
zone "8.ip6.arpa"       { type master; file "master/empty.db"; };
zone "9.ip6.arpa"       { type master; file "master/empty.db"; };
zone "a.ip6.arpa"       { type master; file "master/empty.db"; };
zone "b.ip6.arpa"       { type master; file "master/empty.db"; };
zone "c.ip6.arpa"       { type master; file "master/empty.db"; };
zone "d.ip6.arpa"       { type master; file "master/empty.db"; };
zone "e.ip6.arpa"       { type master; file "master/empty.db"; };
zone "0.f.ip6.arpa"     { type master; file "master/empty.db"; };
zone "1.f.ip6.arpa"     { type master; file "master/empty.db"; };
zone "2.f.ip6.arpa"     { type master; file "master/empty.db"; };
zone "3.f.ip6.arpa"     { type master; file "master/empty.db"; };
zone "4.f.ip6.arpa"     { type master; file "master/empty.db"; };
zone "5.f.ip6.arpa"     { type master; file "master/empty.db"; };
zone "6.f.ip6.arpa"     { type master; file "master/empty.db"; };
zone "7.f.ip6.arpa"     { type master; file "master/empty.db"; };
zone "8.f.ip6.arpa"     { type master; file "master/empty.db"; };
zone "9.f.ip6.arpa"     { type master; file "master/empty.db"; };
zone "a.f.ip6.arpa"     { type master; file "master/empty.db"; };
zone "b.f.ip6.arpa"     { type master; file "master/empty.db"; };
zone "0.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "1.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "2.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "3.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "4.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "5.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "6.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "7.e.f.ip6.arpa"       { type master; file "master/empty.db"; };

// IPv6 ULA (RFC 4193)
zone "c.f.ip6.arpa"     { type master; file "master/empty.db"; };
zone "d.f.ip6.arpa"     { type master; file "master/empty.db"; };

// IPv6 helyi link (RFC 4291)
zone "8.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "9.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "a.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "b.e.f.ip6.arpa"       { type master; file "master/empty.db"; };

// Elavult IPv6 helyi címek (RFC 3879)
zone "c.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "d.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
zone "e.e.f.ip6.arpa"       { type master; file "master/empty.db"; };
```

```
zone "f.e.f.ip6.arpa"       { type master; file "master/empty.db"; };

// Az IP6.INT már elavult (RFC 4159)
zone "ip6.int"          { type master; file "master/empty.db"; };

// FONTOS: Ne használjuk ezeket az IP-címeket, mert nem valódiak,
// csupán illusztrációs és dokumentációs célokból adtuk meg!
//
// Az alárendelt zónák beállításaira vonatkozó bejegyzések. Érdemes
// ilyet beállítani legalább ahhoz a zónához, amelyhez a tartományunk is
// tartozik. Az elsõdleges névszerverhez tartozó IP-címet érdeklõdjük meg
// az illetékes hálózati rendszergazdától.
//
// Soha ne felejtsünk el megadni zónát az inverz kereséshez!  A neve az IP-cím
// tagjainak fordított sorrendjébõl // származik, amelyhez hozzátoldunk még egy
// ".IN-ADDR.ARPA" (illetve IPv6 esetén ".IP6.ARPA") részt.
//
// Mielõtt nekilátnánk egy elsõdleges zóna beállításának, gondoljuk
// végig, hogy tényleg a megfelelõ szinten ismerjük a névfeloldás és
// a BIND mûködését. Gyakran ugyanis egyáltalán nem nyilvánvaló
// csapdákba tudunk esni.  Egy alárendelt zóna beállítása általában sokkal egyszerûbb
feladat.
//
// FONTOS: Ne kövessük vakon a most következõ példát :-) Helyette inkább
// valódi neveket és címeket adjunk meg.

/* Példa dinamikus zónára
key "mintaorgkulcs" {
    algorithm hmac-md5;
    secret "sf87HJqjkqh8ac87a02lla==";
};
zone "minta.org" {
    type master;
    allow-update {
        key "mintaorgkulcs";
    };
    file "dynamic/minta.org";
};
*/

/* Példa inverz alárendelt zónákra
zone "1.168.192.in-addr.arpa" {
    type slave;
    file "slave/1.168.192.in-addr.arpa";
    masters {
        192.168.1.1;
    };
};
*/
```

A named.conf állományban tehát így adhatunk meg közvetlen és inverz alárendelt zónákat.

Minden egyes újabb kiszolgált zónához az egy új bejegyzést kell felvenni a named.conf állományban.

Például a `minta.org` címhez tartozó legegyszerûbb ilyen bejegyzés így néz ki:

```
zone "minta.org" {
    type master;
    file "master/minta.org";
};
```

Ez egy központi zóna, ahogy arról a `type` mezõ, vagyis a típusa is árulkodik. Továbbá a `file` mezõben láthatjuk, hogy a hozzá tartozó információkat az /etc/namedb/master/minta.org állományban tárolja.

```
zone "minta.org" {
    type slave;
    file "slave/minta.org";
};
```

Az alárendelt esetben a zónához tartozó információkat a zóna központi szerverétõl kapjuk meg és megadott állományban mentjük el. Ha valamiért a központi szerver leáll vagy nem érhetõ el, akkor az alárendelt szerver az átküldött zóna információk alapján képes helyette kiszolgálni a kéréseket.

A zóna állományok

A `minta.org` címhez tartozó példa központi zóna állomány (amely az /etc/namedb/master/néven.org érhetõ el) tartalma az alábbi:

```
$TTL 3600        ; alapértelmezés szerint 1 óra
minta.org.      IN      SOA      ns1.minta.org. admin.minta.org. (
                                 2006051501      ; sorozatszám
                                 10800           ; frissítés
                                 3600            ; ismétlés
                                 604800          ; lejárat
                                 300             ; TTL negatív válasz
                         )

; névszerverek
                IN      NS      ns1.minta.org.
                IN      NS      ns2.minta.org.

; MX rekordok
                IN      MX 10   mx.minta.org.
                IN      MX 20   levelezes.minta.org.

                IN      A       192.168.1.1
```

```
; a gépek nevei
localhost       IN      A       127.0.0.1
ns1             IN      A       192.168.1.2
ns2             IN      A       192.168.1.3
mx              IN      A       192.168.1.4
levelezes       IN      A       192.168.1.5

; álnevek
www             IN      CNAME   minta.org.
```

A "."-ra végzõdõ hálózati nevek abszolút nevek, míg minden más "." nélküli név az õsére vezethetõ vissza (tehát relatív). Például az `ns1` névbõl az `ns1.minta.org` keletkezik.

A zóna állományok felépítése a következõ:

```
rekordnév      IN rekordtípus   érték
```

A névfeloldásban leggyakrabban alkalmazott rekordok típusai:

SOA
: a zóna fennhatóságának kezdete

NS
: egy hitelesített névszerver

A
: egy gép címe

CNAME
: egy álnév kanonikus neve

MX
: levélváltó

PTR
: mutató a tartománynévre (az inverz feloldás használja)

```
minta.org. IN SOA ns1.minta.org. admin.minta.org. (
                        2006051501      ; sorozatszám
                        10800           ; 3 óránként frissítsünk
                        3600            ; 1 óra után próbálkozzunk újra
                        604800          ; 1 hét után jár le
                        300 )           ; TTL negatív válasz
```

`minta.org.`
: a tartomány neve, amely egyben a zóna õse

`ns1.minta.org.`

a zóna elsõdleges/hitelesített névszervere

`admin.minta.org.`

a zónáért felelõs személy neve, akinek az e-mail címét a "@" behelyettesítésével kapjuk meg. (Tehát a admin@example.org címbõl `admin.example.org` lesz.)

`2006051501`

az állomány sorozatszáma. Ezt a zóna állomány módosításakor mindig növelnünk kell. Manapság a rendszergazdák a sorozatszámot `éééévhhnnvv` alakban adják meg. A `2006051501` tehát azt jelenti, hogy az állományt 2006. május 15-én módosították utoljára, és a `01` pedig arra utal, hogy aznap elõször. A sorozatszám megadása fontos az alárendelt névszerverek számára, mivel így tudják megállapítani, hogy a zóna mikor változott utoljára.

```
                IN NS           ns1.minta.org.
```

Ez egy NS bejegyzés. A zónához tartozó minden hitelesített névszervernek lennie kell legalább egy ilyen bejegyzésének.

```
localhost       IN      A       127.0.0.1
ns1             IN      A       192.168.1.2
ns2             IN      A       192.168.1.3
mx              IN      A       192.168.1.4
levelezes       IN      A       192.168.1.5
```

Az A rekord egy gép nevét adja meg. Ahogy a fenti példából is kiderül, az `ns1.minta.org` név a `192.168.1.2` címre képzõdik le.

```
                IN      A       192.168.1.1
```

Ez a sor `192.168.1.1` címet rendeli az aktuális õshöz, amely jelen esetünkben az `example.org`.

```
www             IN CNAME        @
```

A kanonikus neveket tároló rekordokat általában egy gép álneveihez használjuk. Ebben a példában a `www` a "fõgép" egyik álneve, amely itt éppenséggel a `minta.org` (`192.168.1.1`) tartományneve. A CNAME rekordok mellé más típusú rekordokat ugyanarra a hálózati névre soha ne adjunk meg.

```
                IN MX   10      levelezes.minta.org.
```

Az MX rekord adja meg, hogy milyen levelezõ szerverek felelõsek a zónába érkezõ levelek fogadásáért. A `levelezes.minta.org` a levelezõ szerver hálózati neve, ahol a 10 az adott levelezõ szerver prioritása.

Több levelezõ szerver is megadható 10-es, 20-as stb. prioritásokkal. A `minta.org` tartományon belül elõször mindig a legnagyobb MX prioritással rendelkezõ levelezõ szervernek próbáljuk meg továbbítani a leveleket (a legkisebb prioritási értékkel rendelkezõ rekord), majd ezután a második legnagyobbnak stb. egészen addig, amíg a levelet tovább nem küldtük.

Az in-addr.arpa zóna állományok (inverz DNS) esetén ugyanez a felépítés, kivéve, hogy a PTR típusú bejegyzések szerepelnek az A és CNAME helyett.

```
$TTL 3600

1.168.192.in-addr.arpa. IN SOA ns1.minta.org. admin.minta.org. (
                        2006051501      ; sorozatszám
                        10800           ; frissítés
                        3600            ; ismétlés
                        604800          ; lejárat
                        300 )           ; TTL negatív válasz

        IN      NS      ns1.minta.org.
        IN      NS      ns2.minta.org.

1       IN      PTR     minta.org.
2       IN      PTR     ns1.minta.org.
3       IN      PTR     ns2.minta.org.
4       IN      PTR     mx.minta.org.
5       IN      PTR     levelezes.minta.org.
```

Ez az állomány írja le tehát a kitalált tartományunkon belül az IP-címek és hálózati nevek összerendelését.

Érdemes megemlíteni, hogy a PTR rekordok jobb oldalán álló nevek mindegyikének teljes hálózati névnek kell lennie (vagyis "." karakterrel kell végzõdnie).

29.6.7. A gyorsítótárazó névszerver

A gyorsítótárazó névszerver az a névszerver, amely elsõdleges feladata a rekurzív kérések kiszolgálása. Egyszerûen továbbítja a beérkezõ kéréseket, majd megjegyzi azokat, így késõbb közvetlenül tud válaszolni.

29.6.8. Biztonság

Habár a névfeloldás szempontjából a BIND a legelterjedtebb, a biztonságosságával azért akadnak gondok. Gyakran találnak benne potenciális és kihasználható biztonsági réseket.

A FreeBSD azonban a named démont automatikusan egy chroot(8) környezetbe helyezi. Emellett még léteznek további más védelmi mechanizmusok is, amelyek segítségével el tudjuk kerülni a névfeloldást célzó esetleges támadásokat.

Sosem árt olvasgatni a CERT által kiadott biztonsági figyelmeztetéseket és feliratkozni a FreeBSD security notifications levelezési lista címére, hogy folyamatosan értesüljünk az interneten és a

FreeBSD-ben talált különbözõ biztonsági hibákról.

Ha valamilyen gondunk támadna, akkor esetleg próbálkozzunk meg a forrásaink frissítésével és a named újrafordításával.

29.6.9. Egyéb olvasnivalók

A BIND/named man oldalai: rndc(8) named(8) named.conf(8)

- Az ISC BIND hivatalos honlapja (angolul)
- Az ISC BIND hivatalos fóruma (angolul)
- O'Reilly DNS and BIND 5th Edition
- RFC1034 - Domain Names - Concepts and Facilities
- RFC1035 - Domain Names - Implementation and Specification

29.7. Az Apache webszerver

29.7.1. Áttekintés

A FreeBSD szolgálja ki a legforgalmasabb honlapok nagy részét szerte a világban. A mögöttük álló webszerverek általában az Apache webszervert alkalmazzák. Az Apache használatához szükséges csomagok megtalálhatóak a FreeBSD telepítõlemezén is. Ha a FreeBSD elsõ telepítésekor még nem telepítettük volna az Apache szerverét, akkor a www/apache13 vagy www/apache12 portból tudjuk feltenni.

Az Apache szervert sikeres telepítését követõen be kell állítanunk.

Ebben a szakaszban az Apache webszerver 1.3.X változatát mutatjuk be, mivel ezt használják a legtöbben FreeBSD alatt. Az Apache 2.X rengeteg új technológiát vezetett be, de ezekkel itt most nem foglalkozunk. Az Apache 2.X változatával kapcsolatban keressük fel a http://httpd.apache.org/ oldalt.

29.7.2. Beállítás

Az Apache webszerver konfigurációs állománya FreeBSD alatt /usr/local/etc/apache/httpd.conf néven található. Ez az állomány egy szokványos UNIX®-os szöveges konfigurációs állomány, ahol a megjegyzéseket egy # karakterrel vezetjük be. Az itt használható összes lehetséges beállítási lehetõség átfogó ismertetése meghaladná az egész kézikönyv határait, ezért most csak a leggyakrabban módosított direktívákat fogjuk ismertetni.

`ServerRoot "/usr/local"`

Ez adja meg az Apache számára az alapértelmezett könyvtárat. A binárisai ezen belül a bin és sbin alkönyvtárakban, a konfigurációs állományai pedig az etc/apache könyvtárban tárolódnak.

`ServerAdmin saját@címünk.az.interneten`

Erre a címre küldhetik nekünk a szerverrel kapcsolatos hibákat. Ez a cím egyes szerver által

generált oldalakon jelenik meg, például hibák esetében.

`ServerName www.minta.com`

A `ServerName` segítségével meg tudjuk adni, hogy milyen nevet küldjön vissza a szerver a klienseknek olyankor, ha az nem egyezne meg a jelenlegivel (vagyis a `www` nevet használjuk a gépünk valódi neve helyett).

`DocumentRoot "/usr/local/www/data"`

A `DocumentRoot` adja meg azt a könyvtárat, ahonnan kiszolgáljuk a dokumentumokat. Alapértelmezés szerint az összes kérés erre a könyvtárra fog vonatkozni, de a szimbolikus linkek és az álnevek akár más helyekre is mutathatnak.

A változtatások végrehajtása elõtt mindig is jó ötlet biztonsági másolatot készíteni az Apache konfigurációs állományairól. Ahogy sikerült összerakni egy számunkra megfelelõ konfigurációt, készen is állunk az Apache futtatására.

29.7.3. Az Apache futtatása

A többi hálózati szervertõl eltérõen az Apache nem az inetd szuperszerverbõl fut. A kliensektõl érkezõ HTTP kérések minél gyorsabb kiszolgálásának érdekében úgy állítottuk be, hogy önállóan fusson. Ehhez egy szkriptet is mellékeltünk, amellyel igyekeztünk a lehetõ legjobban leegyszerûsíteni a szerver indítását, leállítását és újraindítását. Az Apache elsõ indításához adjuk ki a következõ parancsot:

```
# /usr/local/sbin/apachectl start
```

Így pedig a szervert bármikor leállíthatjuk:

```
# /usr/local/sbin/apachectl stop
```

Ha valamilyen okból megváltoztattuk volna a szerver beállításait, akkor ezen a módon tudjuk újraindítani:

```
# /usr/local/sbin/apachectl restart
```

Ha a jelenleg megnyitott kapcsolatok felbontása nélkül akarjuk újraindítani az Apache szervert, akkor ezt írjuk be:

```
# /usr/local/sbin/apachectl graceful
```

Mindezekrõl az apachectl(8) man oldalon találunk bõvebb leírást.

Amennyiben szükségünk lenne az Apache elindítására a rendszer indításakor, akkor a következõ sort vegyünk fel az /etc/rc.conf állományba:

```
apache_enable="YES"
```

Az Apache 2.2 esetében:

```
apache22_enable="YES"
```

Amikor az Apache `httpd` nevû programjának szeretnénk további paranccsori paramétereket átadni a rendszer indítása során, akkor ezeket így tudjuk megadni az rc.conf állományban:

```
apache_flags=""
```

Most, miután a webszerverünk mûködik, a böngészõnkkel mindezt ellenõrizni is tudjuk a `http://localhost/` cím beírásával. Ilyenkor az alapértelmezés szerinti /usr/local/www/data/index.html állomány tartalmát láthatjuk.

29.7.4. Virtuális nevek

Az Apache a virtuális nevek használatának két különbözõ módját ismeri. Ezek közül az elsõ módszer a név alapú virtualizáció (Name-based Virtual Hosting). Ilyenkor a kliens HTTP/1.1 fejlécébõl próbálja meg a szerver megállapítani a hivatkozási nevet. Segítségével több tartomány is osztozhat egyetlen IP-címen.

Az Apache név alapú virtualizációjának beállításához az alábbi beállítást kell hozzátennünk a httpd.conf állományhoz:

```
NameVirtualHost *
```

Ha a webszerverünk neve `www.tartomany.hu`, és hozzá egy `www.valamilyenmasiktartomany.hu` virtuális nevet akarunk megadni, akkor azt a következõképpen tehetjük meg a httpd.conf állományon belül:

```
<VirtualHost *>
ServerName www.tartomany.hu
DocumentRoot /www/tartomany.hu
</VirtualHost>

<VirtualHost *>
ServerName www.valamilyenmasiktartomany.hu
DocumentRoot /www/valamilyenmasiktartomany.hu
</VirtualHost>
```

A címek és elérési utak helyére helyettesítsük be a használni kívánt címeket és elérési utakat.

A virtuális nevek beállításának további részleteivel kapcsolatosan keressük fel az Apache hivatalos dokumentációját a http://httpd.apache.org/docs/vhosts/ címen (angolul).

29.7.5. Apache-modulok

Az alap szerver képességeinek kiegészítéséhez több különbözõ Apache modul áll rendelkezésünkre. A FreeBSD Portgyûjteménye az Apache telepítése mellett lehetõséget ad a népszerûbb bõvítményeinek telepítésére is.

mod_ssl

A mod_ssl modul az OpenSSL könyvtár használatával valósít meg erõs titkosítást a biztonságos socket réteg második, illetve harmadik verziójával (Secure Sockets Layer, SSL v2/v3) és a biztonságos szállítási rétegbeli (Transport Layer Security v1) protokoll segítségével. Ez a modul mindent biztosít ahhoz, hogy a megfelelõ hatóságok által aláírt tanúsítványokat tudjunk kérni, és ezáltal egy védett webszervert futtassunk FreeBSD-n.

Ha még nem telepítettünk volna fel az Apache szervert, akkor a www/apache13-modssl porton keresztül a mod_ssl modullal együtt is fel tudjuk rakni az Apache 1.3.X változatát. Az SSL támogatása pedig már az Apache 2.X www/apache22 porton keresztül elérhetõ változataiban alapértelmezés szerint engedélyezett.

Kapcsolódás nyelvekhez

Mindegyik nagyobb szkriptnyelvhez létezik egy külön Apache-modul, amelyek segítségével komplett Apache-modulokat tudunk készíteni az adott nyelven. Gyakran a dinamikus honlapok is így próbálják a szerverbe épített belsõ értelmezõn keresztül a külsõ értelmezõ indításából és benne a szkriptek lefuttatásából fakadó költségeket megspórolni, ahogy errõl a következõ szakaszokban olvashatunk.

29.7.6. Dinamikus honlapok

Az utóbbi évtizedben egyre több vállalkozás fordult az internet felé bevételeik és részesedéseinek növelésének reményében, amivel egyre jobban megnõtt az igény a dinamikus honlapokra is. Miközben bizonyos cégek, mint például a Microsoft®, a saját fejlesztésû termékeikbe építettek be ehhez támogatást, addig a nyílt forrásokkal foglalkozó közösség sem maradt tétlen és felvette a kesztyût. A dinamikus tartalom létrehozásához többek közt Django, Ruby on Rails, a mod_perl és a mod_php modulok használhatóak.

Django

A Django egy BSD típusú licensszel rendelkezõ keretrendszer, amelynek használatával nagy teljesítményû és elegáns webes alkalmazásokat tudunk gyorsan kifejleszteni. Tartalmaz egy objektum-relációs leképezõt, így az adattípusokat Python-objektumokként tudjuk leírni, és ezekhez az objektumokhoz egy sokrétû, dinamikus adatbázis hozzáférést nyújtó alkalmazásfejlesztõi felületet, így a fejlesztõknek egyetlen SQL utasítást sem kell megírniuk. Találhatunk még benne továbbá egy bõvíthetõ sablonrendszert, amelynek köszönhetõen az alkalmazás belsõ mûködése elválasztható a HTML-beli megjelenésétõl.

A Django mûködéséhez a mod_python modulra, az Apache szerverre és egy tetszõlegesen választott SQL alapú adatbázisrendszerre van szükség. A hozzá tartozó FreeBSD port mindezeket automatikusan telepíti a megadott beállítások szerint.

Példa 35. A Django telepítése az Apache, mod_python3 és a PostgreSQL használatával

```
# cd /usr/ports/www/py-django; make all install clean -DWITH_MOD_PYTHON3
-DWITH_POSTGRESQL
```

Miután a Django és a hozzá szükséges komponensek felkerültek rendszerünkre, hozzunk létre egy könyvtárat a leendõ Django projektünknek és állítsuk be az Apache szervert, hogy az oldalunk belül a megadott linkekre a saját alkalmazásunkat hívja meg a beágyazott Python-értelmezõn keresztül.

Példa 36. Az Apache beállítása a Django és mod_python használatához

A következõ sort kell hozzátennünk a httpd.conf állományhoz, hogy az Apache bizonyos linkeket a webes alkalmazás felé irányítson át:

```
<Location "/">
    SetHandler python-program
    PythonPath "['/a/django/csomagok/helye/'] + sys.path"
    PythonHandler django.core.handlers.modpython
    SetEnv DJANGO_SETTINGS_MODULE azoldalam.beallitasai
    PythonAutoReload On
    PythonDebug On
</Location>
```

Ruby on Rails

A Ruby on Rails egy olyan másik nyílt forráskódú keretrendszer, amivel lényegében egy teljes fejlesztõi készletet kapunk és amelyet kifejezetten arra élezték ki, hogy segítségével a webfejlesztõk sokkal gyorsabban tudjanak haladni és a komolyabb alkalmazások gyorsabb elkészítése se okozzon nekik gondot. A Portrgyûjteményböl pillanatok alatt telepíthetõ.

```
# cd /usr/ports/www/rubygem-rails; make all install clean
```

mod_perl

Az Apache és Perl egyesítésén fáradozó projekt a Perl programozási nyelv és az Apache webszerver erejének összehangolásán dolgozik. A mod_perl modulon keresztül Perlben vagyunk képesek modulokat készíteni az Apache szerverhez. Ráadásul a szerverben egy belsõ állandó értelmezõ is található hozzá, ezzel igyekeznek megspórolni a külsõ értelmezõ és a Perl indításából keletkezõ többletköltségeket.

A mod_perl több különbözõ módon állítható munkába. A mod_perl használatához nem szabad elfelejtenünk, hogy a mod_perl 1.0-ás verziója csak az Apache 1.3 változatával mûködik, és a mod_perl 2.0-ás változata pedig csak az Apache 2.X változataival. A mod_perl 1.0 a www/mod_perl portból telepíthetõ, valamint a statikusan beépített változata a www/apache13-modperl portban található. A mod_perl 2.0 a www/mod_perl2 portból rakható fel.

mod_php

A PHP, vagy másik nevén "PHP, a hipertext feldolgozó" egy általános célú szkriptnyelv, amelyet kifejezetten honlapok fejlesztéséhez hoztak létre. A szabványos HTML ágyazható nyelv felépítésében a C, Java™ és Perl nyelveket ötvözi annak elérése érdekében, hogy ezzel segítse a fejlesztõket a dinamikusan generált oldalak minél gyorsabb megírásában.

A PHP5 támogatását úgy tudjuk hozzáadni az Apache webszerverhez, ha telepítjük a lang/php5 portot.

Ha a lang/php5 portot most telepítjük elõször, akkor a vele kapcsolatos beállításokat tartalmazó `OPTIONS` menü automatikusan megjelenik. Ha ezzel nem találkoznánk, mert például valamikor korábban már felraktuk volna a lang/php5 portot, akkor a port könyvtárában következõ parancs kiadásával tudjuk újra visszahozni:

```
# make config
```

A beállítások között jelöljük be az `APACHE` opciót, amelynek eredményeképpen létrejön az Apache webszerverhez használható mod_php5 betölthetõ modul.

A PHP4 modult még ma is rengeteg szerver használja több különbözõ okból (például kompatibilitási problémák vagy a már korábban kiadott tartalom miatt). Ha tehát a mod_php5 helyett inkább a mod_php4 modulra lenne szükségünk, akkor a lang/php4 portot használjuk. A lang/php4 portnál is megtalálhatjuk a lang/php5 fordítási idejû beállításainak nagy részét.

Az iméntiek révén települnek és beállítódnak a dinamikus PHP alkalmazások támogatásához szükséges mouldok. Az /usr/local/etc/apache/httpd.conf állományban ellenõrizni is tudjuk, hogy az alábbi részek megjelentek-e:

```
LoadModule php5_module        libexec/apache/libphp5.so
```

```
AddModule mod_php5.c
    <IfModule mod_php5.c>
        DirectoryIndex index.php index.html
    </IfModule>
    <IfModule mod_php5.c>
        AddType application/x-httpd-php .php
        AddType application/x-httpd-php-source .phps
    </IfModule>
```

Ahogy befejezõdött a mûvelet, a PHP modul betöltéséhez mindösszesen az `apachectl` paranccsal kell óvatosan újraindítanunk a webszervert:

```
# apachectl graceful
```

A PHP jövõbeni frissítéseihez már nem lesz szükségünk a `make config` parancsra, mivel a korábban kiválasztott `OPTIONS` menün belüli beállításainkat a FreeBSD Portgyûjteményéhez tartozó keretrendszer automatikusan elmenti.

A PHP FreeBSD-ben megtalálható támogatása kifejezetten moduláris, ezért az alap telepítése igencsak korlátozott. A további elemek hozzáadásához a lang/php5-extensions portot tudjuk használni. A port egy menüvezérelt felületet nyújt a PHP különbözõ bõvítményeinek telepítéséhez. Az egyes bõvítményeket azonban a megfelelõ portok használatával is fel tudjuk rakni.

Például PHP5 modulhoz úgy tudunk támogatást adni a MySQL adatbázis szerverhez, ha telepítjük a databases/php5-mysql portot.

Miután telepítettünk egy bõvítményt, az Apache szerverrel újra be kell töltetnünk a megváltozott beállításokat:

```
# apachectl graceful
```

29.8. Állományok átvitele (FTP)

29.8.1. Áttekintés

Az adatállomány átviteli protokoll (File Transfer Protocol, FTP) a felhasználók számára lehetõséget ad az ún. FTP szerverekre állományokat feltölteni, illetve onnan állományokat letölteni. A FreeBSD alaprendszere is tartalmaz egy ilyen FTP szerverprogramot, ftpd néven. Ezért FreeBSD alatt egy FTP szerver beállítása meglehetõsen egyszerû.

29.8.2. Beállítás

A beállítás legfontosabb lépése, hogy eldöntsük milyen hozzáféréseken át lehet elérni az FTP szervert. Egy hétköznapi FreeBSD rendszerben rengeteg hozzáférés a különbözõ démonokhoz tartozik, de az ismeretlen felhasználók számára nem kellene megengednünk ezek használatát. Az /etc/ftpusers állományban szerepelnek azok a felhasználók, akik semmilyen módon nem érhetik el az FTP szolgáltatást. Alapértelmezés szerint itt találhatjuk az elõbb említett rendszerszintû hozzáféréseket is, de ide minden további nélkül felvehetjük azokat a felhasználókat, akiknél nem akarjuk engedni az FTP elérését.

Más esetekben elõfordulhat, hogy csak korlátozni akarjuk egyes felhasználók FTP elérését. Ezt az /etc/ftpchroot állományon keresztül tehetjük meg. Ebben az állományban a lekorlátozni kívánt felhasználókat és csoportokat írhatjuk bele. Az ftpchroot(5) man oldalán olvashatjuk el ennek részleteit, ezért ennek pontos részleteit itt most nem tárgyaljuk.

Ha az FTP szerverünkhöz névtelen (anonim) hozzáférést is engedélyezni akarunk, akkor ahhoz elõször készítenünk kell egy `ftp` nevû felhasználót a FreeBSD rendszerünkben. A felhasználók ezután az `ftp` vagy `anonymous` nevek, valamint egy tetszõleges jelszó (ez a hagyományok szerint a felhasználó e-mail címe) használatával is képesek lesznek bejelentkezni. Az FTP szerver ezután a névtelen felhasználók esetében meghívja a chroot(2) rendszerhívást, és ezzel lekorlátozza hozzáférésüket az `ftp` felhasználó könyvtárára.

Két szöveges állományban adhatunk meg a becsatlakozó FTP kliensek számára üdvözlõ üzeneteket. Az /etc/ftpwelcome állomány tartalmát még a bejelentkezés elõtt látni fogják a felhasználók, a sikeres bejelentkezést követõen pedig az /etc/ftpmotd állomány tartalmát látják. Vigyázzunk, mert ennek az állománynak már a bejelentkezési környezethez képest relatív az elérése, ezért a névtelen felhasználók esetében ez konkrétan az ~ftp/etc/ftpmotd állomány lesz.

Ahogy beállítottuk az FTP szervert, az /etc/inetd.conf állományban is engedélyeznünk kell. Itt mindössze annyira lesz szükségünk, hogy eltávolítsuk a megjegyzést jelzõ "#" karaktert a már meglevõ ftpd sor elõl:

```
ftp stream  tcp nowait  root    /usr/libexec/ftpd   ftpd -l
```

Ahogy arról már a Az inetd konfigurációs állományának újraolvasása szót ejtett, az inetd beállításait újra be kell olvastatnunk a konfigurációs állomány megváltoztatása után. A Beállítások írja le az inetd engedélyezésének részleteit.

Az ftpd önálló szerverként is elindítható. Ehhez mindössze elegendõ csak a megfelelõ változót beállítani az /etc/rc.conf állományban:

```
ftpd_enable="YES"
```

Miután megadtuk az iménti változót, a szerver el fog indulni a rendszer következõ indítása során. Szükség esetén természetesen `root` felhasználóként a következõ paranccsal is közvetlenül elindítható:

```
# /etc/rc.d/ftpd start
```

Most már be is tudunk jelentkezni az FTP szerverre:

```
% ftp localhost
```

29.8.3. Karbantartás

Az ftpd démon a syslog(3) használatával naplózza az üzeneteket. Alapértelmezés szerint a rendszernaplózó démon az FTP mûködésére vonatkozó üzeneteket az /var/log/xferlog állományba írja. Az FTP naplóinak helyét az /etc/syslog.conf állományban tudjuk módosítani:

```
ftp.info      /var/log/xferlog
```

Legyünk körültekintõek a névtelen FTP szerverek üzemeltetésekor. Azt pedig kétszer is gondoljuk meg, hogy engedélyezzük-e a névtelen felhasználók számára állományok feltöltését, hiszen könnyen azon kaphatjuk magunkat, hogy az FTP oldalunk illegális állománycserék színterévé válik vagy esetleg valami sokkal rosszabb történik. Ha mindenképpen szükségünk lenne erre a lehetõségre, akkor állítsunk be olyan engedélyeket a feltöltött állományokra, hogy a többi névtelen

felhasználó ezeket a tartalmuk tüzetes ellenõrzéséig ne is olvashassa.

29.9. Állomány- és nyomtatási szolgáltatások Microsoft® Windows® kliensek számára (Samba)

29.9.1. Áttekintés

A Samba egy olyan elterjedt nyílt forráskódú szoftver, ami Microsoft® Windows® kliensek számára tesz lehetõvé állomány- és nyomtatási szolgáltatásokat. Az ilyen kliensek általa helyi meghajtóként képesek elérni a FreeBSD állományrendszerét, vagy helyi nyomtatóként a FreeBSD általt kezelt nyomtatókat.

A Samba csomagja általában megtalálható a FreeBSD telepítõeszközén. Ha a FreeBSD-vel együtt nem raktuk fel a Samba csomagját, akkor ezt késõbb net/samba3 port vagy csomag telepítésével pótolhatjuk.

29.9.2. Beállítás

A Samba konfigurációs állománya a telepítés után /usr/local/shared/examples/samba/smb.conf.default néven található meg. Ezt kell lemásolnunk /usr/local/etc/smb.conf néven, amelyet aztán a Samba tényleges használata elõtt módosítanunk kell.

Az smb.conf állomány a Samba futásához használt beállításokat tartalmazza, mint például Windows® kliensek számára felkínált a nyomtatók és "megosztások" adatait. A Samba csomagban ezen kívül találhatunk még egy swat nevû webes eszközt, amellyel egyszerû módon tudjuk az smb.conf állományt állítgatni.

A Samba webes adminisztrációs eszköze (SWAT)

A Samba webes adminisztrációs segédeszköze (Samba Web Administration Tool, SWAT) az inetd démonon keresztül fut démonként. Ennek megfelelõn az /etc/inetd.conf állományban a következõ sort kell kivennünk megjegyzésbõl, mielõtt a swat segítségével megkezdenénk a Samba beállítását:

```
swat   stream  tcp     nowait/400      root    /usr/local/sbin/swat    swat
```

Ahogy azt a Az inetd konfigurációs állományának újraolvasása is mutatja, az inetd démont újra kell indítanunk a megváltozott konfigurációs állományának újbóli beolvasásához.

Miután az inetd.conf állományban a swat engedélyezésre került, a böngészõnk segítségével próbáljunk meg a http://localhost:901 címre csatlakozni. Elõször a rendszer `root` hozzáférésével kell bejelentkeznünk.

Miután sikeresen bejelentkeztünk a Samba beállításait tárgyaló lapra, el tudjuk olvasni a rendszer dokumentációját, vagy a **Globals** fülre kattintva nekiláthatunk a beállítások elvégzésének. A **Globals** részben található opciók az /usr/local/etc/smb.conf állomány `[global]` szekciójában található változókat tükrözik.

Általános beállítások

Akár a swat eszközzel, akár a /usr/local/etc/smb.conf közvetlen módosításával dolgozunk, a Samba beállítása során a következõkkel mindenképpen össze fogunk futni:

`workgroup`

A szervert elérni kívánó számítógépek által használt NT tartomány vagy munkacsoport neve.

`netbios name`

A Samba szerver NetBIOS neve. Alapértelmezés szerint ez a név a gép hálózati nevének elsõ tagja.

`server string`

Ez a szöveg jelenik meg akkor, ha például a `net view` paranccsal vagy valamilyen más hálózati segédprogrammal kérdezzük le a szerver beszédesebb leírását.

Biztonsági beállítások

A /usr/local/etc/smb.conf állományban a két legfontosabb beállítás a választott biztonsági modell és a kliensek felhasználói jelszavainak tárolásához használt formátum. Az alábbi direktívák vezérlik ezeket:

`security`

Itt a két leggyakoribb beállítás a `security = share` és a `security = user`. Ha a kliensek a FreeBSD gépen található felhasználói neveiket használják, akkor felhasználói szintû védelemre van szükségünk (tehát a user beállításra). Ez az alapértelmezett biztonsági házirend és ilyenkor a klienseknek elõször be kell jelentkezniük a megosztott erõforrások eléréséhez.

A megosztás (share) szintû védelem esetében, a klienseknek nem kell a szerveren érvényes felhasználói névvel és jelszóval rendelkezniük a megosztott erõforrások eléréséhez. Ez volt az alapbeállítás a Samba korábbi változataiban.

`passdb backend`

A Samba számos különbözõ hitelesítési modellt ismer. A klienseket LDAP, NIS+, SQL adatbázis vagy esetleg egy módosított jelszó állománnyal is tudjuk hitelesíteni. Az alapértelmezett hitelesítési módszer a `smbpasswd`, így itt most ezzel foglalkozunk.

Ha feltesszük, hogy az alapértelmezett `smbpasswd` formátumot választottuk, akkor a Samba úgy fogja tudni hitelesíteni a klienseket, ha elõtte létrehozzuk a /usr/local/private/smbpasswd állományt. Ha a Windows®-os kliensekkel is el akarjuk érni a UNIX®-os felhasználói hozzáféréseinket, akkor használjuk a következõ parancsot:

```
# smbpasswd -a felhasználónév
```

A Samba a 3.0.23c verziójától kezdõdõen a hitelesítéshez szükséges állományokat a /usr/local/etc/samba könyvtárban tárolja. A felhasználói hozzáférések hozzáadására innentõl már a `tdbsam` parancs használata javasolt:

```
# pdbedit -a -u felhasználónév
```

A hivatalos Samba HOGYAN ezekrõl a beállításokról szolgál további információkkal (angolul). Viszont az itt vázolt alapok viszont már elegendõek a Samba elindításához.

29.9.3. A Samba elindítása

A net/samba3 port a Samba irányítására egy új indító szkriptet tartalmaz. A szkript engedélyezéséhez, tehát általa a Samba elindításának, leállításának és újraindításának lehetõvé tételéhez vegyük fel a következõ sort az /etc/rc.conf állományba:

```
samba_enable="YES"
```

Ha még finomabb irányításra vágyunk:

```
nmbd_enable="YES"
```

```
smbd_enable="YES"
```

Ezzel egyben a rendszer indításakor automatikusan be is indítjuk a Samba szolgáltatást.

A Samba a következõkkel bármikor elindítható:

```
# /usr/local/etc/rc.d/samba start
Starting SAMBA: removing stale tdbs :
Starting nmbd.
Starting smbd.
```

Az rc szkriptekkel kapcsolatban a Az rc használata FreeBSD alattt ajánljuk elolvasásra.

A Samba jelen pillanatban három különálló démonból áll. Láthatjuk is, hogy az nmbd és smbd démonokat elindította a samba szkript. Ha az smb.conf állományban engedélyeztük a winbind névfeloldási szolgáltatást is, akkor láthatjuk, hogy ilyenkor a winbindd démon is elindul.

A Samba így állítható le akármikor:

```
# /usr/local/etc/rc.d/samba stop
```

A Samba egy összetett szoftvercsomag, amely a Microsoft® Windows® hálózatokkal kapcsolatos széles körû együttmûködést tesz lehetõvé. Az általa felkínált alapvetõ lehetõségeken túl a többit a http://www.samba.org honlapon ismerhetjük meg (angolul).

29.10. Az órák egyeztetése az NTP használatával

29.10.1. Áttekintés

Idõvel a számítógép órája hajlamos elmászni. A hálózati idõ protokoll (Network Time Protocol, NTP) az egyik módja az óránk pontosan tartásának.

Rengeteg internetes szolgáltatás elvárja vagy éppen elõnyben részesíti a számítógép órájának pontosságát. Például egy webszervertõl megkérdezhetik, hogy egy állományt adott ideje módosítottak-e. A helyi hálózatban az egyazon állományszerveren megosztott állományok ellentmondásmentes dátumozása érdekében szinte elengedhetetlen az órák szinkronizálása. Az olyan szolgáltatások, mint a cron(8) is komolyan építkeznek a pontosan járó rendszerórára, amikor egy adott pillanatban kell lefuttatniuk parancsokat.

A FreeBSD alapból az ntpd(8) NTP szervert tartalmazza, amellyel más NTP szerverek segítségével tudjuk beállítani gépünk óráját, vagy éppen idõvel kapcsolatos információkat szolgáltatni másoknak.

29.10.2. A megfelelõ NTP szerverek kiválasztása

Az óránk egyeztetéséhez egy vagy több NTP szerverre lesz szükségünk. Elõfordulhat, hogy a hálózati rendszergazdánk vagy az internet-szolgáltatónk már beállított egy ilyen szervert erre a célra. Ezzel kapcsolatban olvassuk el a megfelelõ leírásokat. A nyilvánosan elérhetõ NTP szerverekrõl készült egy lista, ahonnan könnyedén ki tudjuk keresni a számunkra leginkább megfelelõ (hozzánk legközelebbi) szervert. Ne hagyjuk figyelmen kívül a szerverre vonatkozó házirendet és kérjünk engedélyt a használatához, amennyiben ez szükséges.

Több, egymással közvetlen kapcsolatban nem álló NTP szerver választásával járunk jól, ha netalán az egyikük váratlanul elérhetetlenné vagy az órája pontatlanná válna. Az ntpd(8) a visszakapott válaszokat intelligensen használja fel, mivel esetükben a megbízható szervereket részesíti elõnyben.

29.10.3. A gépünk beállítása

Alapvetõ beállítások

Ha a számítógépünk indításakor akarjuk egyeztetni az óránkat, akkor erre az ntpdate(8) nevû programot használhatjuk. Ez olyan asztali gépek számára megfelelõ választás, amelyeket gyakran indítanak újra és csak idõnként kell szinkronizálnunk. A legtöbb gépnek viszont az ntpd(8) használatára van szüksége.

Az ntpdate(8) elindítása olyan esetekben is hasznos, ahol az ntpd(8) is fut. Az ntpd(8) az órát fokozatosan állítja, ellenben az ntpdate(8) az eltérés mértékétõl és irányától függetlenül egyszerûen átállítja a gép óráját a pontos idõre.

Az ntpdate(8) elindítását úgy tudjuk engedélyezni a rendszer indításakor, ha az /etc/rc.conf állományba berakjuk az `ntpdate_enable="YES"` sort. Emellett még `ntpdate_flags` változóban meg kell adnunk az alkalmazott beállítások mellett azokat a szervereket, amelyekkel szinkronizálni akarunk.

Általános beállítások

Az NTP az /etc/ntp.conf állományon keresztül állítható, amelyek felépítését az ntp.conf(5) man oldal tárgyalja. Íme erre egy egyszerû példa:

```
server ntplocal.minta.com prefer
server timeserver.minta.org
server ntp2a.minta.net

driftfile /var/db/ntp.drift
```

A `server` beállítás adja meg az egyeztetéshez használt szervereket, soronként egyet. Ha egy szerver mellett szerepel még a `prefer` paraméter is, ahogy azt a példában a `ntplocal.minta.com` mellett láthattuk, akkor a többivel szemben azt a szervert fogjuk elõnyben részesíteni. Az így kiemelt szervertõl érkezõ választ abban az esetben viszont eldobjuk, hogy a többi szervertõl kapott válasz jelentõs mértékben eltér tõle. Minden más esetben a õ válasza lesz a mérvadó. A `prefer` paramétert általában olyan NTP szerverekhez használják, amelyek közismerten nagy pontosságúak, tehát például külön erre a célra szánt felügyeleti eszközt is tartalmaznak.

A `driftfile` beállítással azt az állományt adjuk meg, amiben a rendszeróra frekvencia eltolódásait tároljuk. Az ntpd(8) program ezzel ellensúlyozza automatikusan az óra természetes elmászását, ezáltal lehetõvé téve, hogy egy viszonylag pontos idõt kapjuk még abban az esetben is, amikor egy kis idõre külsõ idõforrások nélkül maradnánk.

A `driftfile` beállítással egyben azt az állományt jelöljük ki, amely az NTP szervertõl kapott korábbi válaszokat tárolja. Ez az NTP mûködéséhez szükséges belsõ adatokat tartalmaz, ezért semmilyen más programnak nem szabad módosítania.

A szerverünk elérésének szabályozása

Alapértelmezés szerint az NTP szerverünket bárki képes elérni az interneten. Az /etc/ntp.conf állományban szereplõ `restrict` beállítás segítségével azonban meg tudjuk mondani, milyen gépek érhetik el a szerverünket.

Ha az NTP szerverünk felé mindenféle próbálkozást el akarunk utasítani, akkor az /etc/ntp.conf állományba a következõ sort kell felvennünk:

```
restrict default ignore
```

Ezzel egyben azonban a helyi beállításainkban szereplõ szerverek elérését is megakadályozzuk. Ha külsõ NTP szerverekkel is szeretnénk szinkronizálni, akkor itt is engedélyezünk kell ezeket. Errõl bõvebben lásd az ntp.conf(5) man oldalon.

Ha csak a belsõ hálózatunkban levõ gépek számára szeretnénk elérhetõvé tenni az órák egyeztetését, de sem a szerver állapotának módosítását nem engedélyezzük, sem pedig azt, hogy a vele egyenrangú szerverekkel szinkronizáljon, akkor az iménti helyett a

```
restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

sort írjuk bele, ahol a `192.168.1.0` a belsõ hálózatunk IP-címe és a `255.255.255.0` a hozzá tartozó hálózati maszk.

Az /etc/ntp.conf több `restrict` típusú beállítást is tartalmazhat. Ennek részleteirõl az ntp.conf(5) man oldalon, az `Access Control Support` címû szakaszban olvashatunk.

29.10.4. Az NTP futtatása

Úgy tudjuk az NTP szervert elindítani a rendszerünkkel együtt, ha az /etc/rc.conf állományban szerepeltetjük az `ntpd_enable="YES"` sort. Ha az ntpd(8) számára további beállításokat is át akarunk adni, akkor az /etc/rc.conf állományban adjuk meg az `ntpd_flags` paramétert.

Ha a gépünk újraindítása nélkül akarjuk elindítani a szerver, akkor az `ntpd` parancsot adjuk ki az /etc/rc.conf állományban a `ntpd_flags` változóhoz megadott paraméterekkel. Mint például:

```
# ntpd -p /var/run/ntpd.pid
```

29.10.5. Az ntpd használati idõleges internet csatlakozással

Az ntpd(8) program megfelelõ mûködéséhez nem szükséges állandó internet kapcsolat. Ha azonban igény szerinti tárcsázással építjünk fel ideiglenes kapcsolatot, akkor érdemes letiltani az NTP forgalmát, nehogy feleslegesen aktiválja vagy tartsa életben a vonalat. Ha PPP típusú kapcsolatunk van, akkor az /etc/ppp/ppp.conf állományban a `filter` direktívával tudjuk ezt leszabályozni. Például:

```
set filter dial 0 deny udp src eq 123
# Nem engedjük az NTP által küldött adatoknak, hogy tárcsázást
# kezdeményezzenek:
set filter dial 1 permit 0 0
set filter alive 0 deny udp src eq 123
# Nem engedjük az NTP adatainak, hogy fenntartsák a kapcsolatot:
set filter alive 1 deny udp dst eq 123
set filter alive 2 permit 0/0 0/0
```

Mindenezekrõl részletesebb felvilágosítást a ppp(8) man oldal `PACKET FILTERING` címû szakaszában és a /usr/shared/examples/ppp/ könyvtárban található példákban kaphatunk.

Egyes internet-szolgáltatók blokkolják az alacsonyabb portokat, ezáltal az NTP nem használható, mivel a válaszok nem fogják elérni a gépünket.

29.10.6. További olvasnivalók

Az NTP szerver dokumentációja HTML formátumban a /usr/shared/doc/ntp/ könyvtárban található.

29.11. Távoli gépek naplózása `syslogd` használatával

A rendszernaplókkal kapcsolatos mûveletek egyaránt fontosak a biztonság és a karbantartás szempontjából. Ha közepes vagy nagyobb méretû, esetleg különbözõ típusú hálózatokban adminisztrálunk több gépet, akkor könnyen átláthatatlanná válhat a naplók rendszeres felügyelete. Ilyen helyzetekben a távoli naplózás beállításával az egész folyamatot sokkal kényelmesebbé tehetjük.

Némileg képesek vagyunk enyhíteni a naplóállományok kezelésének terhét, ha egyetlen központi szerverre küldjük át az adatokat. Ekkor a FreeBSD alaprendszerében megtalálható alapeszközökkel, mint például a syslogd(8) vagy a newsyslog(8) felhasználásával egyetlen helyen be tudjuk állítani a naplók összegyûjtését, összefésülését és cseréjét. A most következõ példa konfigurációban az `A` gép, a `naploszerver.minta.com` fogja gyûjteni a helyi hálózatról érkezõ naplóinformációkat. A `B` gép, a `naplokliens.minta.com` pedig a szervernek küldi a naplózandó adatokat. Éles környezetben mind a két gépnek rendelkeznie kell megfelelõ DNS bejegyzésekkel, vagy legalább szerepelniük kell egymás /etc/hosts állományaiban. Ha ezt elmulasztjuk, a szerver nem lesz hajlandó adatokat fogadni.

29.11.1. A naplószerver beállítása

A naplószerverek olyan gépek, amelyeket úgy állítottunk be, hogy naplózási információkat tudjanak fogadni távoli számítógépekrõl. A legtöbb esetben így egyszerûsíteni tudunk a konfiguráción, vagy olykor egyszerûen csak hasznos, ha ezt a megoldást alkalmazzuk. Függetlenül attól, hogy miért használjuk, a továbblépés elõtt néhány elõkészületet meg kell tennünk.

Egy rendesen beállított naplószervernek legalább a következõ követelményeknek kell eleget tennie:

- az 514-es UDP portot engedélyezni kell mind a kliensen, mind pedig a szerveren futó tûzfal szabályrendszerében;
- a syslogd(8) képes legyen a távoli kliens gépekrõl érkezõ üzeneteket fogadni;
- a syslogd(8) szervernek és az összes kliensnek rendelkeznie kell érvényes DNS (közvetlen és inverz) bejegyzésekkel vagy szerepelnie kell az /etc/hosts állományban.

A naplószerver beállításához mindegyik klienst fel kell vennünk az /etc/syslog.conf állományba, valamint meg kell adnunk a megfelelõ funkciót (facility):

```
+naplokliens.minta.com
*.*     /var/log/naplokliens.log
```

A syslog.conf(5) man oldalán megtalálhatjuk a különbözõ támogatott és elérhetõ *funkciókat*.

Miután beállítottuk, az összes adott funkcióhoz tartozó üzenet az elõbb megadott állományba (/var/log/naplokliens.log) fog kerülni.

A szerveren továbbá meg kell adnunk a következõ sort az /etc/rc.conf állományban:

```
syslogd_enable="YES"
syslogd_flags="-a naplokliens.minta.com -vv"
```

Az elsõ sorral engedélyezzük a `syslogd` elindítását a rendszerindítás során, majd a második sorral engedélyezzük, hogy a kliens naplózni tudjon a szerverre. Itt még látható a `-vv` opció, amellyel a naplózott üzenetek részletességét tudjuk növelni. Ennek nagyon fontos a szerepe a naplózási funkciók behangolásakor, mivel így a rendszergazdák pontosan láthatják milyen típusú üzenetek milyen funkcióval kerültek rögzítésre a naplóban.

Befejezésképpen hozzuk létre a naplóállományt. Teljesen mindegy, hogy erre milyen megoldást alkalmazunk, például a touch(1) remekül megfelel:

```
# touch /var/log/naplokliens.log
```

Ezután indítsuk újra és ellenõrizzük a `syslogd` démont:

```
# /etc/rc.d/syslogd restart
# pgrep syslog
```

Ha válaszul megkapjuk a futó démon azonosítóját, akkor sikerült újraindítanunk, elkezdhetjük a kliens beállítását. Ha valamiért nem indult volna újra a szerver, az /var/log/messages állományból próbáljuk meg kideríteni az okát.

29.11.2. A naplókliens beállítása

A naplókliens az a gép, amely egy helyi naplópéldány karbantartása mellett továbbküldni a naplózandó információkat egy naplószervernek.

Hasonlóan a naplószerverekhez, a klienseknek is teljesítenie bizonyos alapvetõ elvárásokat:

- a syslogd(8) démon küldjön bizonyos típusú üzeneteket a naplószervernek, amely ezeket pedig képes legyen fogadni;
- a hozzá tartozó tûzfal engedje át a forgalmat az 514-es UDP porton;
- rendelkezzen mind közvetlen, mind pedig inverz DNS bejegyzéssel, vagy szerepeljenek az /etc/hosts állományban.

A kliens beállítása sokkal egyszerûbb a szerverhez képest. A kliensen adjuk hozzá a következõ sorokat az /etc/rc.conf állományhoz:

```
syslogd_enabled="YES"
syslogd_flags="-s -vv"
```

A szerver beállításaihoz hasonlóan itt is engedélyezzük a `syslogd` démont és megnöveljük a naplózott üzenetek részletességét. A `-s` kapcsolóval pedig megakadályozzuk, hogy a kliens más

gépekrõl is hajlandó legyen naplóüzeneteket elfogadni.

A funkciók a rendszernek azon részét írják le, amelyhez létrejön az adott üzenet. Tehát például az `ftp` és `ipfw` egyaránt ilyen funkciók. Amikor keletkezik egy naplóüzenet valamelyikükhöz, általában megjelenik a nevük. A funkciókhoz tartozik még egy prioritás vagy szint is, amellyel az adott üzenet fontosságát jelzik. Ezek közül a leggyakoribb a `warning` (mint "figyelmeztetés") és `info` (mint "információ"). A használható funkciók és a hozzájuk tartozó prioritások teljes listáját a syslog(3) man oldalán olvashatjuk.

A naplószervert meg kell adnunk a kliens /etc/syslog.conf állományában. Itt a `@` szimbólummal jelezzük, hogy az adatokat egy távoli szerverre szeretnénk továbbküldeni, valahogy így:

```
*.*             @naploszerver.minta.com
```

Ezután a beállítás érvényesítéséhez újra kell indítanunk a `syslogd` démont:

```
# /etc/rc.d/syslogd restart
```

A logger(1) használatával próbáljuk ki a kliensrõl a aplóüzenetek hálózaton keresztüli küldését, és küldjünk valamit a `syslogd` démonnak:

```
# logger "Udvozlet a naplokliensrol"
```

A parancs kiadása után az üzenetnek mind a kliens, mind pedig a szerver /var/log/messages állományában meg kell jelennie.

29.11.3. Hibakeresés

Elõfordulhat, hogy a naplószerver valamiért nem kapja meg rendesen az üzeneteket, ezért valamilyen módon meg kell keresnünk a hiba okát. Ez több minden lehet, de általában két leggyakoribb ok valamilyen hálózati kapcsolódási vagy DNS beállítási hiba. Ezek teszteléséhez gondoskodjunk róla, hogy a gépek kölcsönösen elérhetõek egymásról az /etc/rc.conf állományban megadott hálózati nevük szerint. Ha ezzel látszólag minden rendben van, akkor próbáljuk meg módosítani a `syslogd_flags` értékét az /etc/rc.conf állományban.

A most következõ példában a /var/log/naplokliens.log teljesen üres, illetve a /var/log/messages állomány semmilyen hibára utaló okot nem tartalmaz. A hibakereséshez még több információt a `syslogd_flags` átírásával tudunk kérni:

```
syslogd_flags="-d -a naploklien.minta.com -vv"
```

Természetesen ne felejtsük el újraindítani a szervert:

```
# /etc/rc.d/syslogd restart
```

A démon újraindítása után közvetlenül az alábbiakhoz hasonló üzenetek árasztják el a képernyõt:

```
logmsg: pri 56, flags 4, from naploszerver.minta.com, msg syslogd: restart
syslogd: restarted
logmsg: pri 6, flags 4, from naploszerver.minta.com, msg syslogd: kernel boot file is
/boot/kernel/kernel
Logging to FILE /var/log/messages
syslogd: kernel boot file is /boot/kernel/kernel
cvthname(192.168.1.10)
validate: dgram from IP 192.168.1.10, port 514, name naplokliens.minta.com;
rejected in rule 0 due to name mismatch.
```

A diagnosztikai üzeneteket végigolvasva nyilvánvaló válik, hogy azért dobja el az üzeneteket a szerver, mert nem megfelelõ a gép neve. Miután átnézzük a beállításainkat, felfedezhetünk az /etc/rc.conf állományban egy apró hibát:

```
syslogd_flags="-d -a naploklien.minta.com -vv"
```

Láthatjuk, hogy ebben a sorban a `naplokliens` névnek kellene szerepelni, nem pedig a `naploklien` névnek. Miután elvégeztük a szükséges javításokat, indítsuk újra a szervert és vizsgáljuk meg az eredményt:

```
# /etc/rc.d/syslogd restart
logmsg: pri 56, flags 4, from naploszerver.minta.com, msg syslogd: restart
syslogd: restarted
logmsg: pri 6, flags 4, from naploszerver.minta.com, msg syslogd: kernel boot file is
/boot/kernel/kernel
syslogd: kernel boot file is /boot/kernel/kernel
logmsg: pri 166, flags 17, from naploszerver.minta.com, msg Dec 10 20:55:02
<syslog.err> naploszerver.minta.com syslogd: exiting on signal 2
cvthname(192.168.1.10)
validate: dgram from IP 192.168.1.10, port 514, name naplokliens.minta.com;
accepted in rule 0.
logmsg: pri 15, flags 0, from naplokliens.minta.com, msg Dec 11 02:01:28 pgj: Masodik
teszt uzenet
Logging to FILE /var/log/naplokliens.log
Logging to FILE /var/log/messages
```

Itt már minden üzenet rendben megérkezett és a megfelelõ állományokba került (a /var/log/messages a kliensen, és a /var/log/naplokliens.log a szerveren)).

29.11.4. Biztonsági megfontolások

Mint minden hálózati szolgáltatás esetén, ilyenkor is figyelembe kell vennünk bizonyos biztonsági megfontolásokat a tényleges konfiguráció kiépítése elõtt. Olykor elõfordulhat, hogy a naplók különbözõ kényes információkat tartalmaznak, mint például a helyi rendszeren futó szolgáltatások nevei, felhasználói nevek vagy egyéb konfigurációs adatok. A kliens és a szerver között hálózaton

utazó adatok viszont se nem titkosítottak, se nem jelszóval védettek. Ha titkosítást szeretnénk használni, akkor javasoljuk például a security/stunnel portot, amellyel egy titkosított tunnelen keresztül tudunk adatokat küldeni a hálózaton.

A helyi rendszer biztonságának szavatolása is fontos lehet. A naplók sem a használat során, sem pedig a lecserélésük után nem kerülnek titkosításra. Emiatt a helyi rendszerhez hozzáférõ felhasználók kedvükre nyerhetnek ki belõlük a rendszerünket érintõ konfigurációs információkat. Ezért ilyenkor nagyon fontos, hogy mindig a megfelelõ engedélyeket állítsuk be a naplókra. A newsyslog(8) segédprogrammal be tudjuk állítani a frissen létrehozott és a lecserélt naplók engedélyeit. Tehát könnyen megakadályozhatjuk a helyi felhasználók kíváncsiskodását, ha itt a naplók engedélyeit például a `600` kóddal adjuk meg.

Chapter 30. Tűzfalak

30.1. Bevezetés

A tûzfalakkal a rendszerünkön keresztülfolyó bejövõ és kimenõ forgalmat tudjuk szûrni. A tûzfalak egy vagy több "szabályrendszer" alapján vizsgálják az éppen érkezõ vagy távozó hálózati csomagokat, és vagy továbbengedik ezeket vagy megállítják. A tûzfalak szabályai a csomagok egy vagy több jellemzõjét veszik szemügyre, amelyek lehetnek például a protokoll típusa, a forrás vagy cél hálózati címe, esetleg a forrás- vagy a célport.

A tûzfalak jelentõs mértékben képesek gyarapítani egy gép vagy egy hálózat védelmét. Leginkább a következõkre tudjuk felhasználni:

- A belsõ hálózatunkban futó alkalmazások, szolgáltatások, gépek megvédésére és elszigetelésére az internetrõl érkezõ nem kívánt forgalom ellen
- A belsõ hálózatban levõ gépek elérését tudjuk korlátozni vagy letiltani az interneten elérhetõ szolgáltatások felé
- A hálózati címfordítás (Network Address Translation, NAT) beállításához, ahol a belsõ hálózatunk privát IP-címeket használnak és egy közös kapcsolaton keresztül érik el az internetet (egyetlen IP-címmel, vagy pedig automatikusan kiosztott publikus címekkel).

A fejezet elolvasása során megismerjük:

- hogyan adjuk meg helyesen a csomagok szûrését leíró szabályokat;
- a FreeBSD-be épített tûzfalak közti különbségeket;
- hogyan állítsuk be és használjuk az OpenBSD PF tûzfalát;
- hogyan állítsuk be és használjuk az IPFILTER tûzfalat;
- hogyan állítsuk be és használjuk az IPFW tûzfalat.

A fejezet elolvasása elõtt ajánlott:

- a FreeBSD-hez és az internethez kötõdõ alapvetõ fogalmak ismerete.

30.2. Röviden a tûzfalakról

A tûzfalak szabályrendszereit alapvetõen kétféleképpen tudjuk összeállítani: "inkluzív", vagyis megengedõ, illetve "exkluzív" vagyis kizáró módon. Az exkluzív tûzfalak minden forgalmat átengednek, amirõl nem rendelkeznek a tûzfal szabályai. Az inkluzív tûzfalak ennek pontosan az ellenkezõjét teszik. Csak azt a forgalmat engedik át, amirõl van szabály és minden mást blokkolnak.

Az inkluzív tûzfalak alkalmazásával sokkal jobban kezünkbentudjuk tartani a hálózatunk kimenõ forgalmát, ezért leginkább az internetes szolgáltatásokat futtató rendszerek esetében bizonyulhat jobb választásnak. Emellett az internetrõl a hálózatunk felé irányuló forgalmat is képes szabályozni. Ekkor az egyetlen szabályra sem illeszkedõ csomagokat egyszerûen eldobjuk és naplózzuk. Az inkluzív tûzfalak általában biztonságosabbak az exkluzív típusú társaiknál, mivel esetükben jelentõs mértékben visszaszorul a nem kívánatos átfolyó forgalom.

Hacsak nem emeljük ki külön, a fejezet további részében minden példaként megadott szabályrendszer inkluzív tûzfalat hoz létre.

Ez a típusú védelem még tovább fokozható az "állapottartó tûzfalak" (stateful firewall) használatával. Az ilyen típusú tûzfalak szemmel tartják a rajtuk keresztül megnyitott kapcsolatokat, és vagy csak a már meglevõ kapcsolathoz tartozó forgalmat engedik át vagy nyitnak egy újat. Az állapottartó tûzfalak hátránya, hogy a "Denial of Service" (DoS) típusú támadásokkal szemben sokkal sérülékenyebbek olyan helyzetekben, amikor az új kapcsolatok nagyon gyorsan jönnek létre. A legtöbb tûzfal esetében azonban tudjuk vegyíteni az állapottartó és nem állapottartó viselkedést, és ezzel egy ideális beállítást kialakítani.

30.3. Tûzfalak

A FreeBSD alaprendszerébe három különbözõ tûzfalat építettek be, melyek a következõk: az *IPFILTER* (másik nevén IPF), az *IPFIREWALL* (más néven IPFW) és az *OpenBSD csomagszûrõje* (Packet Filter, azaz PF). A forgalom szabályozására (vagyis alapvetõen a sávszélesség kihasználtságának vezérlésére) a FreeBSD két beépített csomagot tartalmaz: ez az altq(4) és a dummynet(4). Általában a Dummynet az IPFW, míg az ALTQ a PF partnere. Az IPFILTER esetében maga az IPFILTER végzi a címfordítást és a szûrést, a sávszélességet pedig az IPFW a dummynet(4) *vagy* a PF az ALTQ segítségével. Az IPFW és a PF szabályokkal rendelkezik a rendszerünkbe érkezõ vagy onnan távozó csomagokról, habár megoldásaik teljesen máshogy mûködnek és a szabályok megadási módja is eltér.

A FreeBSD azért tartalmaz egyszerre ennyiféle tûzfalat, mert az emberek elvárásai és igényei eltérnek. Egyikük sem tekinthetõ a legjobbnak.

A szerzõ egyébként az IPFILTER megoldását részesíti elõnyben, mivel egy hálózati címfordítást alkalmazó környezetben sokkal könnyebb vele megfogalmazni az állapottartó szabályokat, valamint tartalmaz egy beépített FTP proxyt is, amivel így a kimenõ FTP kapcsolatok beállítása még tovább egyszerûsödik.

Mivel az összes tûzfal a csomagok fejlécének bizonyos mezõinek alapján dolgozik, ezért a tûzfal szabályrendszerét megalkotó egyénnek teljesen tisztában kell lennie a TCP/IP mûködésével, továbbá azzal, hogy ezekben a mezõkben milyen értékek szerepelhetnek és ezeket hogyan használják egy átlagos kapcsolat alatt. Ebben a témában a http://www.ipprimer.com/overview.cfm címen találhatunk egy remek ismertetõt (angolul).

30.4. Az OpenBSD csomagszûrõje (PF) és az ALTQ

2003 júliusában az OpenBSD PF néven ismert csomagszûrõjét átírták FreeBSD-re és elérhetõvé tették a FreeBSD Portgyûjteményének részeként. A PF programot beépítetten tartalmazó elsõ kiadás pedig 2004 novemberében a FreeBSD 5.3 volt. A PF egy teljes, mindentudó tûzfal, amely támogatja az ún. ALTQ (Alternate Queuing, vagyis a "váltóbesorolás") megoldást. Az ALTQ lehetõvé teszi a sávszélesség korlátozását a szolgáltatás minõsége (Quality of Service, QoS) alapján.

Az OpenBSD Projekt kiváló munkát végez a PF felhasználói útmutatójának karbantartásával. A kézikönyv ezen szakasza ezért elsõsorban azzal foglalkozik, hogyan kell a PF-et FreeBSD alatt használni, miközben igyekszik egy általános összefoglalást adni a témáról. A részletesebb

információkkal kapcsolatban azonban feltétlenül nézzük meg a felhasználói útmutatót.

A http://pf4freebsd.love2party.net/ címen olvashatunk többet arról (angolul), hogy a PF-et hogyan használjunk FreeBSD-n.

30.4.1. A PF rendszermagmodulok használata

A PF modul betöltéséhez a következõ sort kell felvennünk az /etc/rc.conf állományba:

```
pf_enable="YES"
```

Ezt követõen futtassuk le a hozzá tartozó rendszerindító szkriptet:

```
# /etc/rc.d/pf start
```

A PF modul abban az esetben nem fog betöltõdni, ha nem találja a szabályokat tartalmazó konfigurációs állományt. Ez alapértelmezés szerint az /etc/pf.conf állomány. Ha a szabályok leírása rendszerünkön máshol található, akkor az /etc/rc.conf állományban a következõ módon adhatjuk meg annak pontos helyét:

```
pf_rules="/elérési/út/pf.conf"
```

A FreeBSD 7.0 kiadással a minta pf.conf állomány az /etc könyvtárból átkerült a /usr/shared/examples/pf könyvtárba. A FreeBSD 7.0 elõtti kiadásokban alapértelmezés szerint található egy pf.conf állomány az /etc könyvtárban.

A PF modul parancssorból akár kézzel is betölthetõ:

```
# kldload pf.ko
```

A PF mûködésének naplózását a `pflog.ko` teszi lehetõvé, amelyet az alábbi sor hozzáadásával engedélyezhetünk az /etc/rc.conf állományban:

```
pflog_enable="YES"
```

A modul betöltését a hozzá tartozó rendszerindító szkript segítségével kérhetjük:

```
# /etc/rc.d/pflog start
```

Ha a PF többi funkcióját is használni szeretnénk, akkor ehhez egy új rendszermagot kell fordítanunk PF támogatással.

30.4.2. A PF rendszermagbeli beállításai

Noha egyáltalán nem szükséges beépítenünk a PF támogatását a rendszermagba, abban az esetben mégis szükségünk lehet rá, amikor a PF olyan komolyabb lehetõségeit szeretnénk kiaknázni, amelyek már nem részei a modulnak. Ilyen például a pfsync(4), amely a PF által használt állapottáblázatok bizonyos változásainak megjelenítésére alkalmas pszeudoeszköz. A carp(4) megoldásával párosítva így akár hibatûrõ tûzfalak is kialakíthatóak a PF-fel. A CARP megoldásáról a kézikönyvben bõvebb ismertetést a A Közös cím redundancia protokoll (CARP) ad.

A PF rendszermag konfigurációs beállításai a /usr/src/sys/conf/NOTES állományban találhatóak:

```
device pf
device pflog
device pfsync
```

A `device pf` beállítás engedélyezi a csomagszûrõ tûzfalat (pf(4)).

A `device pflog` megadásával keletkezik egy pflog(4) pszeudo hálózati eszköz, amellyel egy bpf(4) eszközre érkezõ forgalmat tudunk naplózni. Ezután a pflogd(8) démon használható tõle származó naplózott adatok rögzítésére.

A `device pfsync` engedélyezi a pfsync(4) pszeudo hálózati eszköz létrejöttét, amely az ún. "állapotváltások" megfigyelésére alkalmas.

30.4.3. Az rc.conf állományban elérhetõ beállítások

A következõ rc.conf(5) beállítások aktiválják a rendszerindítás során a PF és a pflog(4) használatát:

```
pf_enable="YES"                 # a PF engedélyezése (a modul betöltése, ha kell)
pf_rules="/etc/pf.conf"         # a pf szabályait tartalmazó állomány
pf_flags=""                     # a pfctl indításához szükséges további paraméterek
pflog_enable="YES"              # a pflogd(8) elindítása
pflog_logfile="/var/log/pflog"  # hol tartsa a pflogd az naplóit
pflog_flags=""                  # a pflogd indításához szükséges paraméterek
```

Ha a tûzfalunk mögött egy helyi hálózat is meghúzódik, akkor az ott levõ gépek számára valamilyen módon tudnunk kell továbbítani a csomagokat vagy címfordítást kell végezni, így ez is mindenképpen kelleni fog:

```
gateway_enable="YES"            # az átjáró funkciók engedélyezése
```

30.4.4. A szûrési szabályok megfogalmazása

A PF a beállításait a pf.conf(5) állomány tárolja (amely alapértelmezés szerint az /etc/pf.conf helyen található), és az ebben található szabályok alapján módosítja, dobja el vagy éppen engedi át a csomagokat. A FreeBSD rendszerünkben ehhez találhatunk néhány példát a

/usr/shared/examples/pf/ könyvtárban. A PF által használt szabályokról minden részletre kiterjedõen a PF felhasználói útmutatójában olvashatunk.

A PF felhasználói útmutatójának olvasásakor ne feledkezzünk meg róla, hogy a különbözõ FreeBSD verziók különbözõ PF verziókat tartalmaznak. A FreeBSD 7.*X* és késõbbi változatok az OpenBSD 4.1 kiadásában szereplõ PF változatot tartalmazzák.

A FreeBSD packet filter levelezési lista remek hely a PF tûzfal beállításával és futtatásával kapcsolatos kérdésekre. A kérdezés elõtt azonban ne felejtsük el alaposan átnézni az archívumot!

30.4.5. A PF használata

A PF a pfctl(8) segítségével vezérelhetõ. Az alábbiakban ezzel kapcsolatban most összefoglalunk néhány hasznos parancsot (de ne felejtsük el megnézni a pfctl(8) man oldalon található többi lehetõséget sem):

Parancs	Leírás
`pfctl -e`	A PF engedélyezése
`pfctl -d`	A PF tiltása
`pfctl -F all -f /etc/pf.conf`	Az összes (címfordítási, szûrési, állapottartási stb.) szabály törlése, és az /etc/pf.conf állomány újratöltése
`pfctl -s [ rules | nat | state ]`	A szûrési (`rules`), címfordítási (`nat`) és állapottartási (`state`) információk lekérdezése
`pfctl -vnf /etc/pf.conf`	Az /etc/pf.conf állomány ellenõrzése a benne levõ szabályok betöltése nélkül

30.4.6. Az ALTQ engedélyezése

Az ALTQ kizárólag csak úgy használható, ha a konfigurációs beállításokon keresztül beépítjük a FreeBSD rendszermagjába. Az ALTQ alkalmazását nem minden hálózati kártya meghajtója támogatja, ezért ezt a altq(4) man oldalon ellenõrizzük.

A következõ rendszermag konfigurációs beállításokkal engedélyezhetjük az ALTQ használatát és bõvíthetjük azt további lehetõségekkel:

```
options         ALTQ
options         ALTQ_CBQ        # osztályozás alapú besorolás (Class Bases Queuing,
CBQ)
options         ALTQ_RED        # véletlen korai észlelés (Random Early Detection,
RED)
options         ALTQ_RIO        # RED befele/kifele
options         ALTQ_HFSC       # hiearchikus csomagütemezõ (Hierarchical Packet
Scheduler, HFSC)
options         ALTQ_PRIQ       # prioritásos besorolás (Priority Queuing, PRIQ)
```

```
options         ALTQ_NOPCC      # az SMP esetén kell
```

Az `options ALTQ` az ALTQ rendszert engedélyezi.

Az `options ALTQ_CBQ` engedélyezi a osztályozás alapú besorolást (*Class Based Queuing*, CBQ). A CBQ használatával a kapcsolatunkhoz tartozó sávszélességet különbözõ osztályokra vagy sorokra tudjuk bontani és a szûrési szabályoknak megfelelõen osztályozni segítségükkel a forgalmat.

Az `options ALTQ_RED` a véletlen korai észlelés (*Random Early Detection*, RED) használatát engedélyezi. A RED a hálózati forgalomban keletkezõ torlódások elkerülésére alkalmas. A RED ezt a problémát úgy oldja meg, hogy méri a sorok hosszát és összeveti a hozzá tartozó minimális és maximális küszöbértékekkel. Ha a sor hossza meghaladja a számára elõírt maximális értéket, akkor az új csomagokat eldobja. Nevéhez hûen a RED az eldobásra ítélt csomagokat véletlenszerûen választja ki.

Az `options ALTQ_RIO` engedélyezi a RED használatát mind a két irányba, tehát be- és kifelé.

Az `options ALTQ_HFSC` a pártatlan hierachikus szolgáltatási görbe alapú csomagütemezõt (*Hierarchical Fair Service Curve Packet Scheduler*, HFSC) engedélyezi. Vele kapcsolatban a http://www-2.cs.cmu.edu/~hzhang/HFSC/main.html címen találhatunk bõvebben olvasnivalót (angolul).

Az `options ALTQ_PRIQ` a prioritásos besorolást (*Priority Queuing*, PRIQ) teszi elérhetõvé. A PRIQ mindig elsõként a nagyobb értékû sorban levõ forgalmat továbbítja.

Az `options ALTQ_NOPCC` az ALTQSMP, vagyis többprocesszoros támogatását adja meg. Ilyen típusú rendszerekben ez kötelezõ.

30.5. Az IPFILTER (IPF) tûzfal

Az IPFILTER szerzõje Darren Reed. Az IPFILTER nem kötõdik egyik rendszerhez sem: ez egy olyan nyílt forráskódú alkalmazás, amelyet átírtak FreeBSD, NetBSD, OpenBSD, SunOS™, HP/UX és Solaris™ operációs rendszerekre. Az IPFILTER karbantartása és támogatása pillanatnyilag is aktív, folyamatosan jelennek meg újabb változatai.

Az IPFILTER egy rendszermag oldalán mûködõ tûzfalazási és egy címfordítási mechanizmusra alapszik, amelyet felhasználói programokkal tudunk felügyelni és vezérelni. A tûzfal szabályai az ipf(8) segédprogrammal állíthatóak be vagy törölhetõek. A hálózati címfordításra vonatkozó szabályokat az ipnat(1) segédprogrammal állíthatjuk be vagy törölhetjük. Az ipfstat(8) segédprogram képes futás közben statisztikákat készíteni az IPFILTER rendszermagban elhelyezkedõ részeinek viselkedésérõl. Az ipmon(8) program pedig az IPFILTER cselekvéseit képes a rendszernaplókba feljegyezni.

Az IPF eredetileg olyan szabályfeldolgozási módszer szerint készült, amelyben "az utolsó egyezõ szabály nyer" és csak állapotnélküli szabályokat ismert. Az idõ múlásával az IPF részévé vált a "quick" opció és a "keep state" opción keresztül az állapottartás is, melyek drámai mértékben korszerûsítették a szabályok feldolgozásának elvét. Az IPF hivatalos dokumentációja csak a régi szabályok létrehozását és azok feldolgozásának leírását tartalmazza. A korszerûsített funkciók csak kiegészítésképpen jelennek meg, és az általuk felkínált elõnyök megértése egy sokkal magasabb

szintû és biztonságosabb tûzfal megépítését teszik lehetõvé.

A szakaszban szereplõ utasításokban olyan szabályok szerepelnek, amelyek kihasználják a "quick" és "keep state" opciókat. Ezek az inkluzív tûzfalszabályok létrehozásának alapjai.

A régi típusú szabályokról a http://www.obfuscation.org/ipf/ipf-howto.html#TOC_1 és http://coombs.anu.edu.au/~avalon/ip-filter.html címeken olvashatunk (angolul).

Az IPF gyakran ismételt kérdései a http://www.phildev.net/ipf/index.html címen érhetõek el (angolul).

A nyílt forrású IPFILTER levelezési lista kereshetõ archívumait a http://marc.theaimsgroup.com/?l=ipfilter címen találjuk (angolul).

30.5.1. Az IPF engedélyezése

Az IPF megtalálható a FreeBSD alaptelepítésében mint menet közben külön betölthetõ modul. Ha az rc.conf állományba beírjuk a `ipfilter_enable="YES"` sort, akkor ez a modul dinamikusan betöltõdik. A betölthetõ modul alapból naplóz és a `default pass all` beállítást tartalmazza. Ha helyette a `block all` szabályt akarjuk használni, akkor emiatt még nem kell feltétlenül újrafordítanunk a FreeBSD rendszermagját, elég ha egyszerûen csak a szabályrendszerünk végére beszúrjuk.

30.5.2. A rendszermag beállításai

Az IPF használatához nem kötelezõ a következõ beállításokkal újrafordítani a FreeBSD rendszermagját, itt csupán háttérinformációként szerepel. Amikor az IPF a rendszermagba kerül, a betölhetõ modulra nem lesz szükség.

Az IPF a rendszermag forrásai között található /usr/src/sys/conf/NOTES állományban megadott beállításai a következõ módon foglalhatóak össze:

```
options IPFILTER
options IPFILTER_LOG
options IPFILTER_DEFAULT_BLOCK
```

Az `options IPFILTER` engedélyezi az "IPFILTER" tûzfal támogatását.

Az `options IPFILTER_LOG` hatására az IPF az ipl csomagnaplózó pszeudo eszközre jegyzi fel a forgalmat - minden olyan szabály esetén, ahol megjelenik a `log` kulcsszó.

Az `options IPFILTER_DEFAULT_BLOCK` megváltoztatja az alapértelmezett viselkedést, tehát minden olyan csomag, amely nem illeszkedik a tûzfal valamelyik `pass` típusú (átengedõ) szabályára, blokkolásra kerül.

Ezek a beállítások csak azt követõen érvényesülnek, ha fordítottunk és telepítettünk velük egy új rendszermagot.

30.5.3. Az rc.conf állomány beállításai

Az /etc/rc.conf állományban a következõ utasításokra lesz szükségünk az IPF mûködésbe hozására a rendszer indítása során:

```
ipfilter_enable="YES"             # az ipf tûzfal indítása
ipfilter_rules="/etc/ipf.rules"   # betölti a szabályokat tartalmazó szöveges
állományt
ipmon_enable="YES"                # elindítja az IP monitor naplózását
ipmon_flags="-Ds"                 # D = indítás démonként
                                  # s = naplózás a syslog használatával
                                  # v = a tcp ablak, ack, seq csomagok naplózása
                                  # n = az IP-címek és portok feloldása
```

Ha olyan helyi hálózat áll meg a tûzfal mögött, amely egy fenntartott privát IP-címtartományt használ, akkor még a következõ utasításokra is szükségünk lesz a címfordítás bekapcsolásához:

```
gateway_enable="YES"              # a helyi hálózat átjárója
ipnat_enable="YES"                # az ipnat funkció elindítása
ipnat_rules="/etc/ipnat.rules"    # az ipnat mûködéséhez szükséges definíciók
```

30.5.4. IPF

Az ipf(8) parancs használható a szabályokat tartalmazó állomány betöltésére. Általában egy állományba írjuk össze a tûzfal szabályait és ezzel a paranccsal cseréljük le egyszerre a tûzfalban levõ jelenlegi szabályokat:

```
# ipf -Fa -f /etc/ipf.rules
```

Az `-Fa` az összes belsõ szabály törlését jelenti.

Az `-f` jelzi, hogy egy állományból kell beolvasni a betöltendõ szabályokat.

Ezzel mintegy lehetõségünk van változtatni a korábban összeállított szabályainkon, futtatni a fenti IPF parancsot és ezen keresztül úgy frissíteni a szabályok friss másolatával a már mûködõ tûzfalat, hogy nem is kell újraindítanunk a rendszert. Ez a módszer igen kényelmes az új szabályok kipróbálásához, mivel bármikor tetszõlegesen végrehajtható.

Az ipf(8) man oldala tartalmazza a parancsnak megadható további beállításokat.

Az ipf(8) parancs a szabályokat tároló állományt egy szabványos szöveges állománynak tekinti, semmilyen szimbolikus helyettesítést alkalmazó szkriptet nem fogad el.

Lehetõségünk van azonban olyan IPF szabályokat készíteni, amelyek kiaknázzák a szkriptek szimbolikus helyettesítésének lehetõségeit. Errõl bõvebben lásd A szabályok felírása szimbolikus helyettesítéssel.

30.5.5. Az IPFSTAT

Az ipfstat(8) alapértelmezés szerint a arra használatos, hogy le tudjuk kérdezni és megjeleníteni a tûzfalhoz tartozó számlálók értékeit, amelyek a legutóbbi indítás vagy az `ipf -Z` parancs által kiadott lenullázásuk óta a bejövõ vagy kimenõ forgalomból a megadott szabályoknak megfelelõ csomagok alapján gyûjtenek össze statisztikákat.

A parancs mûködésének részleteit az ipfstat(8) man oldalon olvashatjuk.

Az ipfstat(8) meghívása alapból így néz ki:

```
input packets: blocked 99286 passed 1255609 nomatch 14686 counted 0
 output packets: blocked 4200 passed 1284345 nomatch 14687 counted 0
 input packets logged: blocked 99286 passed 0
 output packets logged: blocked 0 passed 0
 packets logged: input 0 output 0
 log failures: input 3898 output 0
 fragment state(in): kept 0 lost 0
 fragment state(out): kept 0 lost 0
 packet state(in): kept 169364 lost 0
 packet state(out): kept 431395 lost 0
 ICMP replies: 0 TCP RSTs sent: 0
 Result cache hits(in): 1215208 (out): 1098963
 IN Pullups succeeded: 2 failed: 0
 OUT Pullups succeeded: 0 failed: 0
 Fastroute successes: 0 failures: 0
 TCP cksum fails(in): 0 (out): 0
 Packet log flags set: (0)
```

Az `-i` mint bejövõ (inbound), vagy az `-o` mint kimenõ (outbound) forgalomra vonatkozó paraméterek megadásával a rendszermagban az adott oldalon jelenleg telepített és alkalmazott szabályokat kérhetjük le és jeleníthetjük meg.

Az `ipfstat -in` parancs így a bejövõ forgalomra vonatkozó belsõ szabályokat mutatja a szabályok számával.

Az `ipfstat -on` parancs a kimenõ forgalmat érintõ belsõ szabályokat mutatja a szabályok számával.

Az eredmény körülbelül ilyen lesz:

```
@1 pass out on xl0 from any to any
@2 block out on dc0 from any to any
@3 pass out quick on dc0 proto tcp/udp from any to any keep state
```

Az `ipfstat -ih` a bejövõ forgalomhoz tartozó belsõ szabályokat mutatja és mindegyik elé odaírja, hogy eddig mennyi csomag illeszkedett rájuk.

Az `ipfstat -oh` ugyanígy a kimentõ forgalom esetén mutatja a belsõ szabályokat és mindegyik elõtt feltünteti, hogy az adott pillanatig mennyi csomag illeszkedett rájuk.

A kimenete nagyjából ilyen lesz:

```
2451423 pass out on xl0 from any to any
354727 block out on dc0 from any to any
430918 pass out quick on dc0 proto tcp/udp from any to any keep state
```

Az `ipfstat` parancs talán egyik legfontosabb funkciója a `-t` kapcsolóval csalható elõ, melynek hatására a rendszerben aktív állapotok táblázatát mutatja meg ugyanúgy, ahogy a top(1) a FreeBSD rendszerben futó programokat. Amikor a tûzfalunk támadás alatt áll, ezzel a funkcióval tudjuk a problémát beazonosítani, leásni a mélyébe és látni a támadótól érkezõ csomagokat. A kiegészítésképpen megadható alkapcsolók megadásával kiválaszthatjuk azt a cél vagy forrás IP-címet, portot vagy protokollt, amelyet valós idõben meg akarunk figyelni. Ennek részleteit az ipfstat(8) man oldalán láthatjuk.

30.5.6. Az IPMON

Az `ipmon` megfelelõ mûködéséhez be kell kapcsolnunk a rendszermag `IPFILTER_LOG` beállítását. Ez a parancs két különbözõ módban használható. Ha parancsot a `-D` opció nélkül gépeljük be, akkor ezek közül alapból a natív módot kapjuk meg.

A démon mód abban az esetben hasznos, ha folyamatosan naplózni akarjuk a rendszerben zajló eseményeket, majd késõbb ezeket átnézni. Így képes egymással együttmûködni a FreeBSD és az IPFILTER. A FreeBSD beépítve tartalmaz olyan lehetõséget, aminek révén magától cseréli a rendszernaplókat. Ezért ha átküldjük a syslogd(8) démonnak a naplózandó üzeneteket, akkor sokkal jobban járunk, mintha egyszerûen csak mezei állományba naplóznánk. Az rc.conf alapértelmezései között az `ipmon_flags` beállítás a `-Ds` kapcsolókat rögzíti:

```
ipmon_flags="-Ds" # D = indítás démonként
                  # s = naplózás a syslog használatával
                  # v = a tcp ablak, ack, seq csomagok naplózása
                  # n = az IP-címek és portok nevének feloldása
```

Ennek a viselkedésnek az elõnyei minden bizonnyal egyértelmûek. Segítségével képesek vagyunk az esetek megtörténte után átnézni, hogyan milyen csomagokat dobott el a rendszer, azok milyen címekrõl érkeztek és hova szánták. Ez egy komoly fegyver a támadók lenyomozásában.

Hiába engedélyezzük a naplózást, az IPF önszántából semmilyen naplózási szabályt nem fog gyártani. A tûzfal gazdájának kell eldöntenie, hogy a szabályokat közül melyiket akarja naplózni, és így neki kell megadnia a `log` kulcsszót ezekben az esetekben. Normális esetben csak a `deny` szabályokat naplózzák.

Egyáltalán nem ritka, hogy a szabályrendszer végén egy alapértelmezés szerint mindent eldobó szabály áll, amely naplóz. Ezzel lehetõségünk nyílik rögzíteni azokat a csomagokat, amelyek egyetlen szabályra sem illeszkedtek.

30.5.7. Naplózás az IPMON használatával

A syslogd egy saját módszert alkalmaz a naplózott adatok elkülönítésére. Egy "funkciók" (facility) és "szintek" (level) segítségével kialakított speciális csoportosítást alkalmaz. Az IPMON `-Ds` módja alapértelmezés szerint a `local0` "funkciót" használja. Ezen túl a következõ szinteken különíthetjük el igényeinknek megfelelõen a naplózott adatokat:

```
LOG_INFO - az átengedés vagy blokkolás helyett a "log" kulcsszóval ellátott csomagok
LOG_NOTICE - az át is engedett csomagok
LOG_WARNING - a blokkolt csomagok
LOG_ERR - a naplózott csomagok közül azok, amelyek túlságosan kicsik (hibás a
fejlécük)
```

Az IPFILTER csak akkor tud naplózni a /var/log/ipfilter.log állományba, ha elõtte létrehozzuk. Az alábbi parancs erre tökéletesen megfelelõ:

```
# touch /var/log/ipfilter.log
```

A syslogd(8) mûködését az /etc/syslog.conf állományban szereplõ definíciók vezérlik. A syslog.conf állomány számottevõ mértékben képes meghatározni azt, ahogy a syslog az IPF és a hozzá hasonló alkalmazásoktól kapott rendszerszintû üzeneteket kezeli.

Az /etc/syslog.conf állományba az alábbi sor kell felvennünk:

```
local0.* /var/log/ipfilter.log
```

A `local0.*` megadásával az összes ilyen típusú üzenet egy elõre rögzített helyre kerül.

Az /etc/syslog.conf állományban elvégzett módosításokat úgy léptethetjük érvénybe, ha újraindítjuk a számítógépet vagy az `/etc/rc.d/syslogd reload` paranccsal megkérjük a syslogd(8) démont, hogy olvassa újra az /etc/syslog.conf állományt.

Az imént létrehozott naplót ne felejtsük el megadni az /etc/newsyslog.conf állományban sem, és akkor ezzel a cseréjét is megoldjuk.

30.5.8. A naplózott üzenetek formátuma

Az `ipmon` által létrehozott üzenetek whitespace karakterekkel elválasztott adatmezõkbõl állnak. A következõ mezõk az összes üzenet esetében megjelennek:

1. A csomag megérkezésének dátuma
2. A csomag megérkezésének idõpontja. ÓÓ:PP:MM.E alakban jelennek meg az órák, percek, másodpercek és ezredmásodpercek (ez több számjegy hosszú is lehet) szerint
3. Azon interfész a neve, ahol a csomag feldolgozásra került, például dc0
4. A szabályhoz tartozó csoport és sorszám, például `@0:17`

Ezek az `ipfstat -in` paranccsal nézhetõek meg.

1. Cselekvés: a p mint átment (passed), b mint blokkolt (blocked), S mint rövid csomag (short packet), n mint egyik szabályra sem illeszkedett (not match), L mint naplózás (log). A módosítók megjelenítésének sorrendje: S, p, b, n, L. A nagybetûs P és B azt jelzi, hogy a csomagot egy felsõbb szintû beállítás miatt naplózták, nem egy szabály hatására.
2. Címek: ez tulajdonképpen három mezõt takar: a forrás címet és portot (melyet egy vesszõ választ el), a → jelet és cél címet és portot. Például: `209.53.17.22,80 → 198.73.220.17,1722`.
3. A `PR` után a protokoll neve vagy száma olvasható, például `PR tcp`.
4. A `len` csomaghoz tartozó fejléc és törzsének teljes hosszát jelöli, például `len 20 40`.

Amennyiben a csomag TCP, egy kötõjellel kezdõdõen további mezõk is megjelenhetnek a beállított opcióknak megfelelõ betûk képében. A betûket és beállításaikat az ipf(5) man oldalán olvashatjuk.

Amennyiben a csomag ICMP, a sort két mezõ zárja, melyek közül az elsõ tartalma mindig "ICMP", és ezt egy perjellel elválasztva az ICMP üzenet típusa és altípusa követi. Tehát például az ICMP 3/3 a "nem elérhetõ port" üzenetet hordozza.

30.5.9. A szabályok felírása szimbolikus helyettesítéssel

Az IPF használatában gyakorlott felhasználók közül néhányan képesek olyan stílusú szabályrendszert készíteni, ahol szimbolikus helyettesítést használnak. Ennek az egyik legnagyobb elõnye az, hogy ilyenkor elég csak a szimbolikus névhez tartozó értéket megváltoztatni és amikor a szkript lefut, akkor az összes rá hivatkozó szabályba ez kerül be. Szkript lévén a szimbolikus helyettesítéssel ki tudjuk emelni a gyakran használt értékeket és behelyettesíteni ezeket több helyre. Ezt a most következõ példában láthatjuk.

Az itt alkalmazott felírás kompatibilis az sh(1), csh(1) és tcsh(1) parancsértelmezõkkel.

A szimbolikus helyettesítést egy dollárjellel fejezzük ki: `$`.

A szimbolikus mezõkben nem szerepel a $ jelölés.

A szimbolikus mezõ tartalmát kettõs idézõjelbe (") tesszük.

Kezdjük így el a szabályok írását:

```
######### Az IPF szabályait tartalmazó szkript eleje ############

oif="dc0"            # a kimenõ interfész neve
odns="192.0.2.11"    # az internet szolgáltató névszerverének IP-címe
myip="192.0.2.7"     # a szolgáltatótól kapott statikus IP-címünk
ks="keep state"
fks="flags S keep state"

# Választhatunk, hogy az /etc/ipf.rules állományt ebbõl a szkriptbõl
# hozzuk létre vagy futtathatjuk "magát" a szkriptet.
#
# Egyszerre csak az egyik sort használjuk.
```

```
#
# 1) Ezzel gyárhatjuk le az /etc/ipf.rules állományt:
#cat > /etc/ipf.rules << EOF
#
# 2) Ezzel futtathajuk "magát" a szkriptet:
/sbin/ipf -Fa -f - << EOF

# Engedélyezzük a szolgáltató névszerverének elérését.
pass out quick on $oif proto tcp from any to $odns port = 53 $fks
pass out quick on $oif proto udp from any to $odns port = 53 $ks

# Engedélyezzük kifelé a titkosítatlan www funkciót.
pass out quick on $oif proto tcp from $myip to any port = 80 $fks

# Engedélyezzük kifelé a TLS SSL felett üzemelõ titkosított www funkciót.
pass out quick on $oif proto tcp from $myip to any port = 443 $fks
EOF
################### Itt az IPF szkript vége ##########################
```

Ennyi lenne. A példában szereplõ szabályok most nem annyira lényegesek, a hangsúly most igazából a szimbolikus helyettesítésen és annak használatán van. Ha a fenti példát az /etc/ipf.rules.script állományba mentjük, akkor ezeket a szabályokat a következõ paranccsal újra tudjuk tölteni:

```
# sh /etc/ipf.rules.script
```

Egyetlen aprócska gond van a beágyazott szimbólumokat tartalmazó állományokkal: az IPF maga nem képes megérteni a helyettesítéseket, azért közvetlenül nem olvassa a szkriptet.

Ez a szkript két módon hasznosítható:

- Vegyük ki megjegyzésbõl a `cat` paranccsal kezdõdõ sort, és tegyük megjegyzésbe az `/sbin/ipf` kezdetût. A megszokottak szerint tegyük az `ipfilter_enable="YES"` sort az /etc/rc.conf állományba, majd minden egyes módosítása után futtassuk le a szkriptet az /etc/ipf.rules állomány létrehozásához vagy frissítéséhez.
- Tiltsuk le az IPFILTER aktiválását a rendszerindításkor, tehát írjuk bele az `ipfilter_enable="NO"` sort (ami mellesleg az alapértelmezett értéke) az /etc/rc.conf állományba.

  Tegyünk egy, az alábbi szkripthez hasonlót az /usr/local/etc/rc.d/ könyvtárba. A szkriptnek adjuk valamilyen értelmes nevet, például ipf.loadrules.sh. Az .sh kiterjesztés használata kötelezõ.

  ```
  #!/bin/sh
  sh /etc/ipf.rules.script
  ```

  A szkript engedélyeit állítsuk be úgy, hogy a `root` tulajdonában legyen és képes legyen olvasni, írni valamint végrehajtani.

```
# chmod 700 /usr/local/etc/rc.d/ipf.loadrules.sh
```

Most miután a rendszer elindult, az IPF szabályai be fognak töltõdni.

30.5.10. Szabályrendszerek az IPF-ben

Az IPF esetében a szabályrendszer olyan szabályokból áll, amelyek a csomagokról tartalmuk alapján eldöntik, hogy át kell engedni vagy vissza kell tartani. A gépek közt két irányban áramló csomagok egy munkamenet alapú társalgást képeznek. A tûzfalhoz tartozó szabályrendszer egyaránt feldolgozza a internetrõl a hálózatunk felé igyekvõ csomagokat, illetve a hálózatunk ezekre adott válaszait. Az egyes TCP/IP szolgáltatásokat (mint például telnet, www, levelezés stb.) a hozzájuk tartozó protokol és szabványos (fogadó) portszám írja le. Ezekre a forrásról általában valamilyen nem szabványos (magasabb értékû) portról érkeznek csomagok. Ekkor a kommunikáció összes paramétere (vagyis a portok és címek) bármelyike alapján definiálhatunk blokkolást vagy továbbengedést leíró szabályokat.

Az IPF eredetileg úgy íródott, hogy a szabályokat "az utolsó illeszkedõ szabály nyer" stílusban dolgozza fel és csak állapot nélküli szabályokat ismert. Az idõk folyamán az IPF szabályai kiegészültek a "quick" és az állapottartásra vonatkozó "keep state" opciókkal, amelynek köszönhetõen óriási mértékben korszerûsödött a szabályok feldolgozása.

A szakaszban szereplõ utasítások olyan szabályokat alkalmaznak, amelyekben egyaránt szerepel a "quick" és az állapottartásért felelõs "keep state" beállítás. Ez az inkluzív tûzfalak létrehozásának egyik alapeszköze.

A tûzfal szabályainak összeállítása során *nagyon óvatosnak* kell lennünk! Bizonyos beállítások hatására akár *ki is zárhatjuk magunkat* a szerverünkrõl. Az ebbõl fakadó esetleges kellemetlenségek elkerülése érdekében javasoljuk, hogy a tûzfal alapjait elõször helyi konzolról építsük fel, ne pedig távolról, például ssh segítségével.

30.5.11. A szabályok felépítése

A szabályok felépítésének bemutatását itt most leszûkítjük a modern állapottartó szabályokra és az "elsõ illeszkedõ szabály nyer" típusú feldolgozásra. A szabályok felírásának régebbi módjai az ipf(8) man oldalon találhatóak.

A # karakterrel egy megjegyzés kezdetét jelezzük, és általában a sor végén vagy egy külön sorban bukkan fel. Az üres sorokat a rendszer nem veszi figyelembe.

A szabályok kulcsszavakat tartalmaznak. Ezeknek a kulcsszavaknak balról jobbra haladva adott sorrendben kell szerepelniük. A kulcsszavakat kiemeltük. Egyes kulcsszavakhoz további beállítások is tartozhatnak, amelyek maguk is kulcsszavak lehetnek, és még további opciókkal rendelkezhetnek. Az alábbi nyelvtan mindegyik elemét kiemeltük és az alábbiakban egyenként kifejtjük a részleteiket.

CSELEKVÉS BE-KI OPCIÓK SZûRÉS ÁLLAPOTTARTÓ PROTOKOLL FORRÁS_CÍM,CÉL_CÍM OBJEKTUM PORTSZÁM TCP_BEÁLLÍTÁS ÁLLAPOTTARTÓ

CSELEKVÉS = block | pass

BE-KI = in | out

OPCIÓK = log | quick | on *interfész*

SZûRÉS = proto *érték* | *forrás/cél IP* | port = *szám* | flags *beállítás*

PROTOKOLL = tcp/udp | udp | tcp | icmp

FORRÁS_CÍM,CÉL_CÍM = all | from *objektum* to *objektum*

OBJEKTUM = *IP-cím* | any

PORTSZÁM = *portszám*

TCP_BEÁLLÍTÁS = S

ÁLLAPOTTARTÓ = keep state

CSELEKVÉS

A cselekvés határozza meg, hogy mit kell tenni azokkal a csomagokkal, amelyek illeszkednek a szabály többi részére. Minden szabályhoz tartoznia *kell* egy cselekvésnek. A következõ cselekvések közül választhatunk:

A `block` megadásával a szabályban szereplõ szûrési feltételre illeszkedõ csomagot eldobjuk.

A `pass` megadásával a szabályban szereplõ szûrési feltételre illeszkedõ csomagot átengedjük a tûzfalon.

BE-KI

Az összes szûrési szabály esetében kötelezõ egyértelmûen nyilatkozunk arról, hogy a bemenõ vagy a kimenõ forgalomra vonatkozik. Ezért a következõ kulcsszó vagy az `in` vagy pedig az `out`, de közülük egyszerre csak az egyiket szabad használni, máskülönben a szabály hibásnak minõsül.

Az `in` jelenti, hogy a szabályt az internet felõl az adott interfészen beérkezõ csomagokra kell alkalmazni.

Az `out` jelenti, hogy a szabályt az internet felé az adott interfészen kiküldött csomagokra kell alkalmazni.

OPCIÓK

Ezek az opciók csak a lentebb bemutatott sorrendben használhatók.

A `log` jelzi, hogy illeszkedés esetén a csomag fejlécét az ipl eszközön keresztül naplózni kell (lásd a naplózásról szóló szakaszt).

A `quick`jelzi, hogy illeszkedés esetén ez lesz a legutolsónak ellenõrzött szabály és így egy olyan "rövidzárat" tudunk képezni a feldolgozásban, amellyel elkerüljük a csomagra egyébként vonatkozó

többi szabály illesztését. Ez az opció a korszerûsített szabályfeldolgozás kihasználásához elengedhetetlen.

Az `on` használatával a szûrés feltételei közé bevonhatjuk a csomaghoz tartozó hálózati interfészt. Itt az interfészek az ifconfig(8) által megjelenített formában adhatóak meg. Az opció megadásával csak az adott interfészen az adott irányba (befelé/kifelé) közlekedõ csomagokra fog illeszkedni a szabály. Ez az opció a korszerûsített szabályfeldolgozás kihasználásához nélkülözhetetlen.

Amikor naplózunk egy csomagot, akkor a hozzá tartozó fejléc az IPL csomagnaplózó pszeudo eszközhöz kerül. A `log` kulcsszó után közvetlenül a következõ minõsítõk szerepelhetnek (a következõ sorrendben):

A `body` jelzi, hogy a csomag tartalmának elsõ 128 byte-ját még jegyezzük fel a fejléc mellé.

A `first` minõsítõt akkor érdemes használnunk, amikor a `log` kulcsszót a `keep state` opcióval együtt alkalmazzuk, mivel ilyenkor csak a szabályt kialakító csomag kerül naplózásra és nem minden olyan, ami illeszkedik az állapottartási feltételekre.

SZûRÉS

Ebben a szakaszban olyan kulcsszavak jelenhetnek meg, amelyekkel a csomagok különféle tulajdonságai alapján ítélkezhetünk azok illeszkedéséről. Itt adott egy kiinduló kulcsszó, amelyhez további kulcsszavak is tartoznak, és amelyek közül csak egyet választhatunk. Az alábbi általános tulajdonságok alapján tudjuk szûrni a csomagokat, ebben a sorrendben:

PROTOKOLL

A `proto` egy olyan kulcsszó, amelyhez hozzá kell rendelnünk még valamelyik opcióját is. Ez az opció segít az adott protokolloknak megfelelõen válogatni a csomagok között. A korszerûsített szabályfeldolgozás lehetõségeinek kihasználásához nélkülözhetetlen.

Opcióként a `tcp/udp | udp | tcp | icmp`, vagy bármelyik, az /etc/protocols állományban megtalálható kulcsszó felhasználható. A `tcp/udp` ebbõl a szempontból speciálisnak tekinthetõ, mivel hatására egyszerre illeszthetõek a szabályra a TCP és UDP csomagok, és így a protokolltól eltekintve azonos szabályok felesleges többszörözését kerülhetjük el.

FORRÁS_CÍM/CÉL_CÍM

Az `all` kulcsszó gyakorlatilag a "from any to any" ("bárhonnan bárhova") szinonímája és nem tartozik hozzá paraméter.

A `from forrás to cél` felépítése: a `from` és `to` kulcsszavak az IP-címek illesztésére használhatóak. Ilyenkor a szabályokban a forrás *és* a cél paramétereknek is szerepelniük kell. Az `any` egy olyan speciális kulcsszó, amely tetszõleges IP-címre illeszkedik. Néhány példa az alkalmazására: `from any to any` vagy `from 0.0.0.0/0 to any`, `from any to 0.0.0.0/0`, `from 0.0.0.0/0 to any` vagy `from any to 0.0.0.0`.

Az IP-címek megadhatóak pontozott numerikus formában a hálózati maszk bitekben mért hosszával együtt, vagy akár egyetlen pontozott numerikus IP-címként.

Nincs lehetõség olyan IP-címtartományok illesztésére, amelyek nem adhatóak meg kényelmesen

ponttal elválasztott számok és maszk hosszával. A net-mgmt/ipcalc port az ilyen számításokat könnyíti meg. A hálózati maszkok hosszának megállapításban segíthet az említett segédprogram (angol nyelvû) honlapja: http://jodies.de/ipcalc.

PORT

Amikor portra vonatkozó illeszkedést írunk elõ, megadhatjuk a forrásra és célra, amit aztán vagy csak TCP vagy pedig csak UDP csomagokra alkalmazunk. A portok feltételeinek megfogalmazásánál használhatjuk a portok számát vagy az /etc/services állományban szereplõ nevüket. Amikor a port egy `from` típusú objektum leírásában jelenik meg, akkor automatikusan a forrásportot jelenti, míg a `to` objektum leírásában pedig a célportot. A `to` objektumoknál a port megadása elengedhetetlen a korszerûsített szabályfeldolgozás elõnyeinek kihasználásához. Példa: `from any to any port = 80`.

Az egyes portokat különbözõ mûveletek segítségével, numerikusan hasonlíthatjuk össze, ahol akár porttartományt is megadhatunk.

port "=" | "!=" | "<" | ">" | "⇐" | ">=" | "eq" | "ne" | "lt" | "gt" | "le" | "ge".

A porttartományok megadásához használjuk a `port` "<>" | "><" felírási módot.

A forrásra és célra vonatkozó paraméterek után szereplõ másik két paraméter nélkülözhetetlen a korszerûsített szabályfeldolgozás mûködéséhez.

TCP_BEÁLLÍTÁS

A beállítások csak a TCP forgalom szûrésénél érvényesülnek. A betûk jelölik azokat a lehetséges beállításokat, amelyek a TCP csomagok fejlécében megvizsgálhatóak.

A korszerûsített szabályfeldolgozás a `flags S` paraméter segítségével ismeri fel a TCP munkameneteket kezdeményezõ kéréseket.

ÁLLAPOTTARTÓ

A `keep state` jelzi, hogy a szabály paramétereinek megfelelõ bármely csomag aktiválja az állapottartó szûrés használatát.

Ez a beállítás feltétlenül szükséges a korszerûsített szabályfeldolgozás megfelelõ kihasználásához.

30.5.12. Állapottartó csomagszûrés

Az állapottartó szûrés a csomagok kétirányú áramlását egy létrejött kapcsolatba sorolja be. Amikor aktiválódik, az állapottartó szabály elõre dinamikusan létrehozza a kétirányú kommunikációban megforduló csomagokhoz a megfelelõ belsõ szabályokat. Olyan vizsgálatokat végez, amelyek segítségével ki tudja deríteni, hogy a csomag küldõje és címzettje között fennálló kétirányú kapcsolat érvényes szabályok szerint zajlik-e. Minden olyan csomagot, amely nem illeszkedik megfelelõen a kapcsolatra vonatkozó sémára, csalásnak tekintjük és automatikusan eldobjuk.

Az állapottartás révén lehetõségünk van a TCP vagy UDP kapcsolatokhoz tartozó ICMP csomagokat

is átengedni a tûzfalon. Tehát ha kapunk egy 3-as típusú, 4-es kódú ICMP választ valamilyen böngészésre használt állapottartó szabályon keresztül kiküldött kérésre, akkor az automatikusan bejöhet. Amelyik csomagot az IPF egyértelmûen képes besorolni az aktív kapcsolatba, még ha az eltérõ protokollt is használ, beengedi.

Ami ilyenkor történik:

Az internethez csatlakozó interfészen keresztül kifelé haladó csomagokat elõször egy dinamikus állapottábla alapján illesztjük, és ha a csomag illeszkedik az aktív kapcsolatban következõként várt csomagra, akkor átmegy a tûzfalon és a dinamikus állapottáblában frissül a kapcsolat állapota. Az aktív munkameneten kívül csomagok pedig egyszerûen a kimenõ szabályrendszer szerint kerülnek ellenõrzésre.

Hasonlóan az elõzõhöz, az internethez csatlakozó interfészen keresztül befelé haladó csomagokat elõször egy dinamikus állapottábla alapján illesztjük, és ha a csomag illeszkedik az aktív kapcsolatban következõként várt csomagra, akkor átmegy a tûzfalon és a dinamikus állapottáblában frissül a kapcsolat állapota. Az aktív munkamenethez nem tartozó csomagok pedig egyszerûen a bejövõ szabályrendszer szerint kerülnek ellenõrzésre.

Amikor egy kapcsolat befejezõdik, automatikusan törlõdik a dinamikus állapottáblából.

Az állapottartó csomagszûrés használatával az újonnan keletkezõ kapcsolatok elutasítására vagy engedélyezésére tudunk koncentrálni. Ha engedélyeztük egy új kapcsolat létrejöttét, akkor a rákövetkezõ összes többi csomag automatikusan átmegy a tûzfalon és minden más hamis csomag eldobódik. Ha tiltjuk az új kapcsolatot, akkor egyetlen rákövetkezõ csomag sem juthat át. Az állapottartó szûrés által felkínált fejlett elemzési lehetõségek képesek védelmet nyújtani a behatolók részérõl alkalmazott megannyi különbözõ támadási módszer ellen.

30.5.13. Példa inkluzív szabályrendszerre

A most következõ szabályrendszer arra mutat példát, hogyan programozzunk le egy nagyon biztonságos inkluzív tûzfalat. Az inkluzív tûzfalak csak a szabályainak megfelelõ szolgáltatásokat engedik keresztül, és alapértelmezés szerint minden mást blokkolnak. Egy hálózat gépeit védõ tûzfalnak, amelyet gyakran "hálózati tûzfalnak" (network firewall) is neveznek, legalább két hálózati interfésszel kell rendelkeznie. Ezeket az interfészeket általában úgy állítják be, hogy tökéletesen megbíznak az egyik oldalban (a helyi hálózatban), a másikban (az internetben) pedig egyáltalán nem. A tûzfalat egyébként úgy is beállíthatjuk, hogy csak a tûzfalat mûködtetõ gépet védje - ezt "egyrendszeres tûzfalnak" (host based firewall) nevezik. Az ilyen típusú megoldásokat nem biztonságos hálózaton keresztül kommunikáló szervereknél alkalmaznak.

Mindegyik UNIX®-típusú rendszert, köztük a FreeBSD-t is úgy alakították ki, hogy az operációs rendszeren belüli kommunikáció az lo0 interfészen és a `127.0.0.1` IP-címen keresztül történik. A tûzfal szabályai között feltétlenül szerepelniük kell olyanoknak, amelyek lehetõvé teszik ezen a speciális intefészen a csomagok zavartalan mozgását.

Az internetre csatlakozó interfészhez kell rendelni a kifelé és befelé haladó forgalom hitelesítését é a hozzáférésének vezérlését. Ez lehet a felhasználói PPP által létrehozott tun0 interfész vagy a DSL-, illetve kábelmodemhez csatlakozó hálózati kártya.

Ahol egy vagy több hálózati kártya is csatlakozik több különbözõ helyi hálózathoz, úgy kell

beállítani a hozzájuk tartozó interfészeket, hogy egymás felé és az internet felé képesek legyenek küldeni és fogadni.

A szabályokat elõször három nagy csoportba kell szerveznünk: elõször jönnek a megbízható interfészek, ezeket követik az internet felé mutató interfészek, végül internet felõl jövõ, nem megbízható interfészeke.

Az egyes csoportokban szereplõ szabályokat úgy kell megadni, hogy közülük elõre kerüljenek a leggyakrabban alkalmazottak, és a csoport utolsó szabálya blokkoljon és naplózzon minden csomagot az adott interfészen és irányban.

A kimenõ forgalomat vezérlõ szabályrendszer csak `pass` (tehát átengedõ) szabályokat tartalmazhat, amelyek bentrõl az interneten elérhetõ szolgáltatásokat azonosítják egyértelmûen. Az összes ilyen szabályban meg kell jelenni a `quick`, `on`, `proto`, `port` és `keep state` beállításoknak. A `proto tcp` szabályok esetében meg kell adni a `flag` opciót is, amivel fel tudjuk ismertetni a kapcsolatok keletkezését és ezen keresztül aktiválni az állapottartást.

A bejövõ forgalmat vezérlõ szabályrendszerben elõször az eldobni kívánt csomagokat kell megadni, aminek két eltérõ oka van. Elõször is elõfordulhat, hogy a veszélyes csomagok részleges illeszkedés miatt szabályosnak tûnnek. Az ilyen csomagokat értelemszerûen nem lenne szabad beengedni a szabályok részleges megfelelése alapján. A másodszor az eleve ismerten problémás és értelmetlen csomagokat csendben el kellene vetni, mielõtt a szakaszhoz tartozó utolsó szabály fogná meg és naplózná. Ez az utolsó szabály egyébként szükség esetén felhasználható a támadók elleni bizonyítékok begyûjtésére.

A másik, amire még oda kell figyelnünk, hogy a blokkolt csomagok esetében semmilyen válasz nem keletkezzen, egyszerûen csak tûnjenek el. Így a támadó nem fogja tudni, hogy a csomagjai vajon elérték-e a rendszerünket. Minél kevesebb információt tudnak összegyûjteni a rendszerünkrõl a támadók, annál több idõt kell szánniuk csínytevéseik kieszelésére. A `log first` opciót tartalmazó szabályok csak az illeszkedésnél fogják naplózni a hozzájuk tartozó eseményt. Erre láthatunk példát az `nmap OS fingerprint` szabálynál. Az security/nmap segédprogramot a támadók gyakran alkalmazzák a megtámadni kívánt szerver operációs rendszerének felderítésére.

Minden `log first` opcióval megadott szabály illeszkedésénél a `ipfstat -hio` parancs meghatározódik az eddigi illeszkedések aktuális száma. Nagyobb értékek esetében következtethetünk arra, hogy a rendszerünket megtámadták (vagyis csomagokkal árasztják éppen el).

Az ismeretlen portszámok felderítésére az /etc/services állomány, esetleg a http://www.securitystats.com/tools/portsearch.php (angol nyelvû) honlap használható.

Érdemes továbbá megnézni a trójai programok által használt portokat a http://www.simovits.com/trojans/trojans.html címen (angolul).

A következõ szabályrendszer egy olyan biztonságos "inkluzív" típusú tûzfal, amelyet éles rendszeren is használnak. Ezt a rendszerünkön nem használt szolgáltatásokra vonatkozó `pass` szabályok törlésével könnyedén a saját igényeink szerint alakíthatjuk.

Ha nem akarunk látni bizonyos üzeneteket, akkor vegyünk fel hozzájuk egy `block` típusú szabályt a befelé irányuló forgalomhoz tartozó szabályok közé.

A szabályokban írjuk át a dc0 interfész nevét annak a hálózati kártyának az interfészére, amelyen keresztül csatlakozunk az internethez. A felhasználói PPP esetében ez a tun0 lesz.

Tehát a következõket kell beírni az /etc/ipf.rules állományba:

```
######################################################################
# A helyi hálózatunkon zajló forgalmat ne korlátozzuk.
# Csak akkor kell, ha helyi hálózathoz is csatlakozunk.
######################################################################

#pass out quick on xl0 all
#pass in quick on xl0 all

######################################################################
# A belsõ interfészen szintén ne korlátozzunk semmit.
######################################################################
pass in quick on lo0 all
pass out quick on lo0 all

######################################################################
# Az internet felé forgalmazó interfész (kimenõ kapcsolatok)
# A saját hálózatunkról belülrõl vagy errõl az átjáróról
# kezdeményezett kapcsolatokat vizsgáljuk az internet felé.
######################################################################

# Engedélyezzük az internet szolgáltatók névszerverének elérését,
# az "xxx" helyett a névszervet IP-címét kell megadni.
# Másoljuk le ezeket a sorokat, ha a szolgáltatónknak több
# névszerverét is beakarjuk állítani. A címeiket az /etc/resolv.conf
# állományban találjuk.
pass out quick on dc0 proto tcp from any to xxx port = 53 flags S keep state
pass out quick on dc0 proto udp from any to xxx port = 53 keep state

# DSL vagy kábeles hálózatoknál engedélyezzük a
# szolgáltatónk DHCP szerverének elérését.
# Ez a szabály nem kell, ha "felhasználói PPP"-vel
# kapcsolódunk az internethez, ilyenkor tehát az egész
# csoport törölhetõ.
# Használjuk az alábbi szabályt és keressük meg a naplóban az
# IP-címet. Ha megtaláltuk, akkor tegyük bele a megjegyzésben
# szereplõ szabályba és töröljük az elsõ szabályt.
pass out log quick on dc0 proto udp from any to any port = 67 keep state
#pass out quick on dc0 proto udp from any to z.z.z.z port = 67 keep state

# Kifelé engedélyezzük a szabványos nem biztonságos WWW funkciókat.
pass out quick on dc0 proto tcp from any to any port = 80 flags S keep state

# Kifelé engedélyezzük a biztonságos WWW funkciókat TLS SSL
# protokollal.
pass out quick on dc0 proto tcp from any to any port = 443 flags S keep state
```

```
# Kifelé engedélyezzük az e-mailek küldését és fogadását.
pass out quick on dc0 proto tcp from any to any port = 110 flags S keep state
pass out quick on dc0 proto tcp from any to any port = 25 flags S keep state

# Kifelé engedélyezzük az idõ szolgáltatást.
pass out quick on dc0 proto tcp from any to any port = 37 flags S keep state

# Kifelé engedélyezzük az nntp híreket.
pass out quick on dc0 proto tcp from any to any port = 119 flags S keep state

# Kifelé engedélyezzük az átjáróról és a helyi hálózatról a nem
# biztonságos FTP használatát (passzív és akív módokban is). Ez a
# funkció a mûködéséhez a nat szabályokat tartalmazó állományban
# hivatkozott FTP proxyt használja. Amennyiben a pkg_add paranccsal
# csomagokat akarunk telepíteni az átjáróra, erre a szabályra
# mindenképpen szükségünk lesz.
pass out quick on dc0 proto tcp from any to any port = 21 flags S keep state

# Kifelé engedélyezzük az ssh/sftp/scp # (biztonságos telnet/rlogin/FTP)
# szolgáltatások # elérését az SSH (secure shell) használatával.
pass out quick on dc0 proto tcp from any to any port = 22 flags S keep state

# Kifelé engedélyezzük a nem biztonságos telnet elérését.
pass out quick on dc0 proto tcp from any to any port = 23 flags S keep state

# Kifelé engedélyezzük FreeBSD CVSUp funkcióját.
pass out quick on dc0 proto tcp from any to any port = 5999 flags S keep state

# Kifelé engedélyezzük a pinget.
pass out quick on dc0 proto icmp from any to any icmp-type 8 keep state

# Kifelé engedélyezzük a helyi hálózatról érkezõ whois kéréseket.
pass out quick on dc0 proto tcp from any to any port = 43 flags S keep state

# Minden mást eldobunk és naplózzuk az elsõ elõfordulásukat.
# Ez a szabály blokkol alapértelmezés szerint mindent.
block out log first quick on dc0 all

######################################################################
# Az internet felõli interfész (bejövõ kapcsolatok)
# A saját hálózatunk felé vagy erre az átjáróra
# nyitott kapcsolatokat vizsgáljuk az internet felõl.
######################################################################

# Eldobjuk az összes olyan bejövõ forgalmat, amit hivatalosan nem
# lehetne továbbítani vagy fenntartott címterülethez tartozik.
block in quick on dc0 from 192.168.0.0/16 to any    #RFC 1918: privát IP
block in quick on dc0 from 172.16.0.0/12 to any     #RFC 1918: privát IP
block in quick on dc0 from 10.0.0.0/8 to any        #RFC 1918: privát IP
block in quick on dc0 from 127.0.0.0/8 to any       #helyi
```

```
block in quick on dc0 from 0.0.0.0/8 to any         #helyi
block in quick on dc0 from 169.254.0.0/16 to any    #DHCP
block in quick on dc0 from 192.0.2.0/24 to any      #dokumentációs célokra fenntartva
block in quick on dc0 from 204.152.64.0/23 to any   #Sun klaszterek összekötésére
használt
block in quick on dc0 from 224.0.0.0/3 to any       #D és E osztályú multicast

##### Itt eldobunk egy rakás csúf dolgot ############
# Ezeket nem akarjuk a naplóban látni:

# Eldobjuk a töredékcsomagokat.
block in quick on dc0 all with frags

# Eldobjuk a túlságosan rövid TCP csomagokat.
block in quick on dc0 proto tcp all with short

# Eldobjuk a forrás által közvetített (source routed) csomagokat.
block in quick on dc0 all with opt lsrr
block in quick on dc0 all with opt ssrr

# Elutasítjuk az "OS fingerprint" kéréseket.
# Naplózzuk az elsõ elõfordulást, így nálunk lesz a kíváncsiskodó
# egyén IP-címe.
block in log first quick on dc0 proto tcp from any to any flags FUP

# Eldobunk mindent, aminek speciális beállításai vannak.
block in quick on dc0 all with ipopts

# Elutasítjuk a publikus pinget.
block in quick on dc0 proto icmp all icmp-type 8

# Elutasítjuk az ident kéréseket.
block in quick on dc0 proto tcp from any to any port = 113

# Blokkoljuk az összes Netbios szolgáltatást: 137=név, 138=datagram,
# 139=session. A Netbios az MS Windows megosztását implementálja.
# Blokkoljuk az MS Windows hosts2 névszerver kéréseit is a 81-es
# porton.
block in log first quick on dc0 proto tcp/udp from any to any port = 137
block in log first quick on dc0 proto tcp/udp from any to any port = 138
block in log first quick on dc0 proto tcp/udp from any to any port = 139
block in log first quick on dc0 proto tcp/udp from any to any port = 81

# Engedélyezzük a szolgáltatónk DHCP szerverétõl érkezõ forgalmat.
# Ebben a szabályban meg kell adnunk a szolgáltató DHCP szerverének
# IP-címét, mivel itt csak a hiteles forrásból fogadunk el csomagokat.
# Erre csak DSL- és kábelmodemes kapcsolat esetében van szükség, a
# "felhasználói PPP" alkalmazása során szükségtelen. Ez az IP-cím
# megegyezik a kimenõ kapcsolatoknál megadott címmel.
pass in quick on dc0 proto udp from z.z.z.z to any port = 68 keep state
```

```
# Befelé engedélyezzük a szabványos WWW funkciót, mivel webszerverünk
# van.
pass in quick on dc0 proto tcp from any to any port = 80 flags S keep state

# Befelé engedélyezzük az internetrõl érkezõ nem biztonságos telnet
# kapcsolatokat. Azért nem biztonságos, mert az azonosítókat és
# jelszavakat titkosítatlan formában közli az interneten keresztül.
# Töröljük ezt a szabályt, ha nem használunk telnet szervert.
#pass in quick on dc0 proto tcp from any to any port = 23 flags S keep state

# Befelé engedélyezzük az internetrõl # érkezõ ssh/sftp/scp (biztonságos
# telnet/rlogin/FTP) # kapcsolatokat az SSH (secure shell) használatával.
pass in quick on dc0 proto tcp from any to any port = 22 flags S keep state

# Minden mást dobjuk el és naplózzuk az elsõ elõfordulásukat.
# Az elsõ alkalom naplózásával elejét tudjuk venni a "Denial of
# Service" típusú támadásoknak, amivel egyébként lehetséges lenne a
# napló elárasztása.
# Ez a szabály blokkol alapértelmezés szerint mindent.
block in log first quick on dc0 all
################### Itt van a szabályok vége ##############################
```

30.5.14. NAT

A NAT jelentése *Network Address Translation*, vagyis hálózati címfordítás. A Linux® esetében ezt "IP masqueradingnak", vagyis IP maszkolásnak hívják. A hálózati címfordítás és az IP maszkolás lényegben ugyanazt takarja. Az IPF címfordításért felelõs funkciójának köszönhetõen képesek vagyunk a tûzfal mögött elhelyezkedõ helyi hálózat számára megosztani az internet-szolgáltatól kapott publikus IP-címet.

Sokakban felmerülhet a kérdés, hogy erre vajon mi szükségünk lehet. Az internet-szolgáltatók a magánszemélyeknek általában dinamikus IP-címeket osztanak ki. A dinamikus itt arra utal, hogy a címünk minden alkalommal változik, amikor betárcsázunk a szolgáltatóhoz vagy amikor ki- és bekapcsoljuk a modemünket. Ez a dinamikus IP-cím fog azonosítani minket az interneten.

Most tegyük fel, hogy öt gépünk van otthon, viszont csak egyetlen elõfizetéssel rendelkezünk. Ebben az esetben öt telefonvonalat kellene használnunk és mindegyik géphez elõfizetni az internetre.

A hálózati címfordítás alkalmazásával azonban mindössze egyetlen elõfizetés kell. A gépek közül négyet hozzákötünk egy switch-hez és a switch-et pedig a fennmaradó géphez, amelyen FreeBSD fut. Ez utóbbi lesz az így kialakított helyi hálózatunk átjárója. A tûzfalban mûködõ címfordítás segítségével a helyi hálózaton található gépek IP-címeit észrevétlenül át tudjuk fordítani a hálózatunk publikus IP-címére, ahogy a csomagok elhagyják az átjárót. A beérkezõ csomagok esetében mindez visszafelé történik meg.

Az IP-címek közül adott egy tartomány, amit a címfordítást használó helyi hálózatok részére tartanak fenn. Az RFC 1918 szerint az alábbi IP-címtartományok használhatók a helyi hálózatban, mivel ezeken keresztül közvetlenül sosem lehet kijutni az internetre:

Kezdõ IP: 10.0.0.0	-	Záró IP: 10.255.255.255
Kezdõ IP: 172.16.0.0	-	Záró IP: 172.31.255.255
Kezdõ IP: 192.168.0.0	-	Záró IP: 192.168.255.255

30.5.15. IPNAT

A címfordításra vonatkozó szabályokat az ipnat paranccsal tudjuk betölteni. Az ilyen típusú szabályokat általában az /etc/ipnat.rules állományban találjuk. A részleteket lásd az ipnat(1) man oldalán.

Amikor a címfordítás üzembe helyezése után meg akarjuk változtatni a címfordítás szabályait, elõször a címfordítás szabályait tartalmazó állományt módosítsuk, majd a belsõ címfordítási szabályok és a címfordítási táblázatban szereplõ aktív bejegyzések törléséhez futassuk le az ipnat parancsot a -CF beállítással.

A címfordítási szabályok újratöltését egy ehhez hasonló paranccsal tudjuk elvégezni:

```
# ipnat -CF -f /etc/ipnat.szabályok
```

A címfordításhoz tartozó statisztikákat ezzel a paranccsal tudjuk lekérdezni:

```
# ipnat -s
```

A címfordítási táblázatban pillanatnyilag szereplõ összerendeléseket a következõ paranccsal tudjuk listázni:

```
# ipnat -l
```

A szabályok feldolgozásával és az aktív szabályokkal/bejegyzésekkel kapcsolatos információk részletezését így engedélyezhetjük:

```
# ipnat -v
```

30.5.16. A címfordítási szabályok

A címfordítási szabályok nagyon rugalmasak és rengeteg olyan funkciót meg tudunk velük valósítani, ami az üzleti és otthoni felhasználók számára egyaránt hasznos.

Itt most a szabályok felépítését csak egyszerûsítve mutatjuk be, leginkább a nem üzleti környezetek tekintetében. A szabályok komplett formai leírását az ipnat(5) man oldalán találjuk.

Egy címfordítási szabály tehát valahogy így néz ki:

```
map INTERFÉSZ HELYI_IP_TARTOMÁNY -> PUBLIKUS_CÍM
```

A szabályt a `map` kulcsszó kezdi.

A *INTERFÉSZ* helyére az internet felé mutató külsõ interfész nevét írjuk be.

A *HELYI_IP_TARTOMÁNY* lesz az, amelyben a kliensek címeznek. Ez például a `192.168.1.0/24`.

A *PUBLIKUS_CÍM* lehet egy külsõ IP-cím vagy a `0/32` speciális kulcsszó, amellyel a *FELÜLET*-hez rendelt IP-címre hivatkozunk.

30.5.17. Hogyan mûködik a hálózati címfordítás

A publikus cél felé haladó csomag megérkezik a helyi hálózatról. Miután a kimenõ kapcsolatokra vonatkozó szabályok átengedik, a címfordítás kapja meg a szerepet és fentrõl lefelé haladva nekilát alkalmazni a saját szabályait, ahol az elsõ egyezõ szerint cselekszik. A címfordítás a szabályokat a csomaghoz tartozó interfészre és a forrás IP-címére illeszti. Amikor a csomag interfészének neve illeszkedik egy címfordítási szabályra, akkor ezután a csomag forrás (vagyis a helyi hálózaton belüli) IP-címérõl igyekszik eldönteni, hogy a szabály nyilának bal oldalán szereplõ tartományba esik-e. Ha erre is illeszkedik, akkor a forrás IP-címét átírjuk a `0/32` kulcsszó alapján felderített publikus IP-címre. A címfordító rutin ezt feljegyzi a saját belsõ táblázatába, így amikor a csomag visszatér az internetrõl, akkor képes lesz visszafordítani az eredeti belsõ IP-címére és feldolgozásra átadni a tûzfal szabályainak.

30.5.18. A címfordítás engedélyezése

A címfordítás életre keltéséhez a következõket kell beállítanunk az /etc/rc.conf állományban.

Elõször engedélyezzük a gépünknek, hogy közvetítsen forgalmat az interfészek között:

```
gateway_enable="YES"
```

Minden alkalommal indítsuk el a címfordításért felelõs IPNAT programot:

```
ipnat_enable="YES"
```

Adjuk meg az IPNAT számára a betöltendõ szabályokat:

```
ipnat_rules="/etc/ipnat.rules"
```

30.5.19. Hálózati címfordítás nagyon nagy helyi hálózatok esetében

Az olyan helyi hálózatokban, ahol rengeteg PC található vagy több alhálózatot is tartalmaz, az összes privát IP-cím egyetlen publikus IP-címbe tömörítése igen komoly problémává tud dagadni és az azonos portok gyakori használata a helyi hálózatra kötött számítógépek között ütközéseket okoz.

Két módon tudunk megoldást nyújtani erre a problémára.

A használható portok kiosztása

Egy normális címfordítási szabály valahogy így nézne ki:

```
map dc0 192.168.1.0/24 -> 0/32
```

A fenti szabályban a csomag forrásportját az IPNAT változatlanul a feldolgozás után hagyja. Ha ehhez még hozzátesszük a `portmap` kulcsszót, akkor ezzel utasítani tudjuk az IPNAT-ot, hogy csak az adott tartományban képezze le a forrásportokat. Például a következõ szabály hatására az IPNAT a forrásportokat egy adott tartományon belül fogja módosítani:

```
map dc0 192.168.1.0/24 -> 0/32 portmap tcp/udp 20000:60000
```

Ha viszont még inkább meg akarjuk könnyíteni a dolgunkat, akkor itt egyszerûen csak adjuk meg az `auto` kulcsszót, amellyel az IPNAT önmagától megállapítja, hogy milyen portokat tud használni:

```
map dc0 192.168.1.0/24 -> 0/32 portmap tcp/udp auto
```

Több publikus cím használata

Minden nagyobb helyi hálózat esetében elérkezünk ahhoz a ponthoz, ahol már egyetlen publikus cím nem elég. Ha több publikus IP-címmel is rendelkezünk, akkor ezekbõl a címekbõl egy "közös készletet" hozhatunk létre, amibõl majd az IPNAT válogathat miközben a csomagok címeit átírja kifelé menetben.

Például ahelyett, hogy a csomagokat egyetlen publikus IP-címre képeznénk le, ahogy itt tesszük:

```
map dc0 192.168.1.0/24 -> 204.134.75.1
```

A hálózati maszk segítségével meg tudjuk adni IP-címek egy tartományát is:

```
map dc0 192.168.1.0/24 -> 204.134.75.0/255.255.255.0
```

CIDR-jelöléssel:

```
map dc0 192.168.1.0/24 -> 204.134.75.0/24
```

30.5.20. A portok átirányítása

Gyakran elõfordul, hogy van webszerverünk, levelezõ szerverünk, adatbázis szerverünk és névszerverünk, melyek a helyi hálózat különbözõ gépein futnak. Ebben az esetben a szerverekhez

tartozó forgalmat is fordítanunk kell, illetve valamilyen módon a bejövõ forgalmat is át kell irányítanunk a helyi hálózat megfelelõ gépeihez. Az IPNAT ezt a gondot a hálózati címfordítás átirányítást támogató funkcióival szünteti meg. Tegyük fel, hogy a 10.0.10.25 belsõ címen van egy webszerverünk, amelyhez a 20.20.20.5 publikus IP tartozik. Ilyenkor a következõ szabályt adjuk meg:

```
rdr dc0 20.20.20.5/32 port 80 -> 10.0.10.25 port 80
```

vagy:

```
rdr dc0 0.0.0.0/0 port 80 -> 10.0.10.25 port 80
```

Így tudjuk beállítani a 10.0.10.33 címmel rendelkezõ névszervert a kintrõl érkezõ névfeloldási kérések fogadására:

```
rdr dc0 20.20.20.5/32 port 53 -> 10.0.10.33 port 53 udp
```

30.5.21. Az FTP és a címfordítás

Az FTP egy olyan õskövület, amely még az internet egy régi korszakából maradt fenn, amikor az egyetemek között még bérelt vonal létezett és az FTP szolgált a kutatók közt az állományok megosztására. Ez még abban az idõben történt, amikor a biztonság egyáltalán nem volt lényeges szempont. Az évek elõrehaladtával az FTP protokoll beleivódott a feltörekvõ internet gerincébe és a titkosítatlanul küldött azonosítóival és jelszavaival továbbra is ugyanolyan védtelen maradt. Az FTP két változatban, aktív és passzív módban képes mûködni. Az eltérés kettejük között az adatcsatorna megállapításában van. A passzív mód sokkal biztonságosabb, mivel ilyenkor az adatcsatornát az FTP kapcsolatot kezdeményezõ állítja be. Az FTP különbözõ módjainak magyarázatát és a köztük levõ különbséget a http://www.slacksite.com/other/ftp.html címen ismerhetjük meg részleteiben (angolul).

Az IPNAT szabályai

Az IPNAT egy speciális beépített FTP proxyval rendelkezik, amelyre a hálózati címfordítás leképezései között hivatkozhatunk. Képes figyelni az összes aktív vagy passzív FTP kapcsolathoz tartozó kimenõ kérést és ezekhez dinamikusan létrehozni olyan ideiglenes szûrési szabályokat, amelyek valóban csak az adatcsatornához felhasznált portokat tartalmazzák. Ezzel ki tudjuk küszöbölni az FTP azon káros hatását a tûzfalra nézve, hogy egyszerre túlságosan sok magasabb tartománybeli port legyen nyitva.

Ez a szabály a belsõ hálózat összes FTP forgalmát lekezeli:

```
map dc0 10.0.10.0/29 -> 0/32 proxy port 21 ftp/tcp
```

Ez a szabály pedig az átjáróról érkezõ FTP forgalommal bírkózik meg:

```
map dc0 0.0.0.0/0 -> 0/32 proxy port 21 ftp/tcp
```

Ez a szabály kezeli a belsõ hálózatról érkezõ összes nem FTP típusú forgalmat:

```
map dc0 10.0.10.0/29 -> 0/32
```

Az FTP leképzésére vonatkozó szabály a szokásos leképzési szabály elé kerül. Az összes csomag fentrõl haladva az elsõ illeszkedõ szabály alapján kerül feldolgozásra. Elõször az interfész nevét vizsgáljuk, majd a belsõ hálózatbeli forrás IP-t, végül azt, hogy a csomag egy FTP kapcsolat része. Ha minden paraméterében megfelel, akkor az FTP proxy készít egy ideiglenes szûrési szabályt hozzá, amellyel az FTP kapcsolathoz tartozó csomagok mind a két irányba képesek lesznek vándorolni, természetesen a címfordítással együtt. Az összes többi bentrõl érkezõ csomag átlép ezen a szabályon és megáll a harmadiknál, ahol az interfésznek és forrás IP-nek megfelelõen átfordítjuk a címét.

Az IPNAT szûrési szabályai FTP-re

Az FTP esetében csak egyetlen szûrési szabályra van szükségünk a hálózati címfordításba épített FTP proxy használatához.

FTP proxy nélkül az alábbi három szabály kellene:

```
# Kifelé engedélyezzük a belsõ gépek FTP elérést az internet irányába,
# aktív és passzív módokban.
pass out quick on rl0 proto tcp from any to any port = 21 flags S keep state

# Kifelé engedélyezzük a passzív módhoz tartozó magasabb tartománybeli
# adatcsatornákat.
pass out quick on rl0 proto tcp from any to any port > 1024 flags S keep state

# Aktív módban beengedjük az FTP szervertõl érkezõ adatcsatornát.
pass in quick on rl0 proto tcp from any to any port = 20 flags S keep state
```

30.6. IPFW

Az IPFIREWALL (IPFW) a FreeBSD által támogatott tûzfalazó alkalmazás, melyet a FreeBSD Projektben résztvevõ önkéntesek fejlesztettek ki és tartanak karban. Régi típusú, állapottartás nélküli szabályokat használ, és az itt használatos szabályírási technikát "egyszerû állapottartó megoldásnak" nevezzük.

Az IPFW szabvány FreeBSD-ben levõ, mintaként szolgáló szabályrendszere (ez az /etc/rc.firewall és /etc/rc.firewall6 állományokban található meg) annyira egyszerû, hogy komolyabb módosítások nélkül nem ajánlatos használni. Ez a példa nem tartalmaz állapottartó szûrést, ami viszont a legtöbb esetben kívánatos lenne, ezért ezt a szakaszt nem erre alapozzuk.

Az IPFW állapottartás nélküli szabályainak felépítésében olyan technikailag kifinomult leválogatási

képességek bújnak meg, amelyek jócskán meghaladják az átlagos tûzfalépítõk tudását. Az IPFW elsõsorban olyan szakemberek vagy szakmailag elõrehaladott felhasználók számára készült, akiknek speciális csomagszûrési igényeik vannak. A különbözõ protokollok használatának és a hozzájuk tartozó fejlécinformációk mindenre kiterjedõ ismerete szinte nélkülözhetetlen az IPFW valódi erejének kihasználásához. Ez a szint azonban túlmutat a kézikönyv ezen szakaszának keretein.

Az IPFW hét komponensbõl épül fel, melyek közül az elsõdleges a rendszermag tûzfalazásért felelõs szabályfeldolgozó és a hozzá tartozó csomagnyilvántartás, majd ezt követi a naplózás, a hálózati címfordítást aktiváló `divert` szabály, valamint a komolyabb célok megvalósítására alkalmas lehetõségek: a forgalom korlátozásáért felelõs dummynet, a továbbküldésre alkalmas `fwd rule` szabály, a hálózati hidak támogatása, illetve az ipstealth. Az IPFW egyaránt használható IPv4 és IPv6 esetén.

30.6.1. Az IPFW engedélyezése

Az IPFW az alap FreeBSD telepítésben külön, futás idõben betölthetõ modulként érhetõ el. Ha az rc.conf állományban megadjuk a `firewall_enable="YES"` beállítást, akkor a rendszer indulásakor ezt a modult dinamikusan betölti. Az IPFW-t csak akkor kell a FreeBSD rendszermagjába beépítenünk, ha szükségünk van a címfordítási funkciójára is.

Ha tehát az rc.conf állományban megadtuk a `firewall_enable="YES"` sort és újraindítottuk a számítógépünket, akkor a következõ fehérrel kiemelt üzenet fog megjelenni a rendszerindítás során:

```
ipfw2 initialized, divert disabled, rule-based forwarding disabled, default to deny,
logging disabled
```

A "logging disabled" üzenetbõl kiderül, hogy a modul nem végez naplózást. A naplózást és a hozzá tartozó részletesség szintjét úgy tudjuk beállítani, ha az /etc/sysctl.conf állományba felvesszük a következõ sorokat, amivel a következõ indításkor már mûködni fog:

```
net.inet.ip.fw.verbose=1
net.inet.ip.fw.verbose_limit=5
```

30.6.2. A rendszermag beállításai

Ha nem akarjuk kihasználni az IPFW által felkínált címfordítási lehetõségeket, akkor egyáltalán nem szükséges a FreeBSD rendszermagjába belefordítani a támogatását. Ezért az alábbiakat csak kiegészítõ információként tüntettük fel.

```
options    IPFIREWALL
```

Ez a beállítás engedélyezi az IPFW használatát a rendszermag részeként.

```
options    IPFIREWALL_VERBOSE
```

Ezzel és a `log` kulcsszóval tudjuk az IPFW szabályain keresztülhaladó csomagokat naplózni.

```
options    IPFIREWALL_VERBOSE_LIMIT=5
```

Ez az érték korlátozza a syslogd(8) segítségével naplózott azonos bejegyzések maximális számát. Ezt a beállítást olyan veszélyes környezetekben érdemes használnunk, ahol naplózni akarunk. Segítségével meg tudjuk akadályozni, hogy a rendszernapló elárasztásával megakasszák a rendszerünket.

```
options    IPFIREWALL_DEFAULT_TO_ACCEPT
```

Ezen beállítás hatására a tûzfal alapértelmezés szerint mindent átenged, ami általában akkor jöhet jól, amikor elõször beállítjuk a tûzfalat.

```
options    IPDIVERT
```

Ezzel a beállítással engedélyezzük a címfordítás használatát.

Ha nem adjuk meg az IPFIREWALL_DEFAULT_TO_ACCEPT beállítást, vagy ha nem engedélyezzük a bejövõ csomagokat, akkor a gépünkre semmilyen csomag nem lesz képes bejutni, illetve onnan kijutni.

30.6.3. Az /etc/rc.conf beállításai

Így tudjuk engedélyezni a tûzfalat:

```
firewall_enable="YES"
```

A FreeBSD-hez mellékelt alapértelmezett tûzfaltípusok közül az /etc/rc.firewall állomány átolvasásával tudunk választani, és megadni az alábbi helyett:

```
firewall_type="open"
```

A következõ értékek állnak rendelkezésünkre:

- `open` - átengedi az összes forgalmat
- `client` - csak ezt a gépet védi
- `simple` - az egész hálózatot védi
- `closed` - a helyi interfész kivételével minden IP alapú forgalmat tilt

- `UNKNOWN` - tiltja a tûzfal szabályainak betöltését
- állománynév - a tûzfal szabályait tartalmazó állomány abszolút elérési útvonala

Két különbözõ módon lehet betölteni a saját ipfw szabályainkat. Az egyik közülük, ha a `firewall_type` változóban megadjuk a *tûzfal szabályait* tartalmazó állomány abszolút elérési útvonalát, az ipfw(8) parancssori beállításai nélkül. Az alábbi példában egy olyan egyszerû szabályrendszert láthatunk, amely blokkolja az összes bejövõ és kimenõ forgalmat:

```
add deny in
add deny out
```

Másrészrõl az `firewall_script` változóban is megadhatjuk azt a szkriptet, amelyben a rendszerindítás során meghívjuk `ipfw` parancsot. Az iménti szabályrendszert az alábbi szkripttel tudjuk kiváltani:

```
#!/bin/sh

ipfw -q flush

ipfw add deny in
ipfw add deny out
```

Ha a `firewall_type` változó `client` vagy `simple` értékét használjuk, akkor az /etc/rc.firewall állományban található alapértelmezett szabályokat érdemes átvizsgálnunk, hogy kellõen illeszkednek-e az adott géphez. Hozzátennénk, hogy a fejezetben szereplõ példák azt feltételezik, hogy a `firewall_script` értéke az /etc/ipfw.rules állomány.

A naplózás így engedélyezhetõ:

```
firewall_logging="YES"
```

A `firewall_logging` változó egyedül csak annyit tesz, hogy beállítja a `net.inet.ip.fw.verbose` sysctl változónak az `1` értéket (lásd Az IPFW engedélyezése). A napló korlátozására nincs külön változó az rc.conf állományon belül, de az /etc/sysctl.conf állomány segítségével és manuálisan be tudjuk állítani a hozzá tartozó változót:

```
net.inet.ip.fw.verbose_limit=5
```

Amennyiben a gépünk átjáróként viselkedik, tehát a natd(8) segítségével címfordítást végez, a Hálózati címfordításban olvashatunk utána, hogy ehhez az /etc/rc.conf állományban milyen beállításokat kell megadnunk.

30.6.4. Az IPFW parancs

Normál esetben az `ipfw` parancs használatos arra, hogy a tûzfal mûködése közben az aktív belsõ szabályai közé vegyünk fel vagy töröljünk közülük manuálisan bejegyzéseket. Ennek a módszernek az egyedüli hátránya, hogy az így végrehajtott módosítások el fognak veszni a rendszer leállításával. Itt inkább azt a megoldást javasoljuk, hogy az összes szabályt tegyük bele egy állományba és a rendszerindítás során ezt töltsük be, majd ha változtatni akarunk a tûzfalon, akkor ezt az állományt módosítsuk és a régiek törlésével töltsük be újra az egész szabályrendszert.

Az `ipfw` parancs mellesleg remekül használható a jelenleg futó tûzfalszabályok megjelenítésére a konzolon. Az IPFW nyilvántartásában az egyes szabályokhoz dinamikusan jönnek létre számlálók, amelyek a rá illeszkedõ csomagokat számolják. A tûzfal tesztelése folyamán a szabályok és hozzá tartozó számlálók lekérdezése a megfelelõ mûködés ellenõrzésének egyik lehetséges módja.

A szabályokat így tudjuk egymás után felsoroltatni:

```
# ipfw list
```

A szabályokat így tudjuk az utolsó illeszkedésük idejével együtt megjeleníteni:

```
# ipfw -t list
```

A következõ példában a nyilvántartási információkat kérdezzük le, ekkor a szabályok mellett az illeszkedõ csomagok száma is láthatóvá válik. Az elsõ sorban a szabály száma szerepel, majd ezt követi rendre az illeszkedõ kimenõ és bejövõ csomagok mennyisége, valamint végül maga a szabály.

```
# ipfw -a list
```

A statikus szabályok mellett a dinamikusakat így lehet kilistázni:

```
# ipfw -d list
```

A lejárt dinamikus szabályokat is meg tudjuk nézni:

```
# ipfw -d -e list
```

A számlálók nullázása:

```
# ipfw zero
```

Csak a *SZÁM* sorszámú szabályhoz tartozó számlálók nullázása:

```
# ipfw zero SZÁM
```

30.6.5. Szabályrendszerek az IPFW-ben

Az IPFW esetében a szabályrendszer olyan szabályokból áll, amelyek a csomagokról tartalmuk alapján eldöntik, hogy át kell engedni vagy vissza kell tartani. A gépek közt két irányban áramló csomagok egy munkamenet alapú társalgást képeznek. A tûzfalhoz tartozó szabályrendszer egyaránt feldolgozza a internetrõl a hálózatunk felé igyekvõ csomagokat, illetve a hálózatunk ezekre adott válaszait. Az egyes TCP/IP szolgáltatásokat (mint például telnet, www, levelezés stb.) a hozzájuk tartozó protokol és szabványos (fogadó) portszám írja le. Ezekre a forrásról általában valamilyen nem szabványos (magasabb értékû) portról érkeznek csomagok. Ekkor a kommunikáció összes paramétere (vagyis a portok és címek) bármelyike alapján definiálhatunk blokkolást vagy továbbengedést leíró szabályokat.

Amikor egy csomag eléri a tûzfalat, a szabályrendszer elsõ szabályával kerül összehasonlításra és amíg nem illeszkedik valamelyikre, addig lefut rá a többi szabály is fentrõl lefelé egyesével, a sorszámuknak megfelelõ növekvõ sorrendben. Ha a csomag megfelel valamelyik szabály leválogatási paramétereinek, akkor a benne megnevezett cselekvés zajlik le, és számára a feldolgozás befejezõdik. Ezt a viselkedést neveztük "az elsõ illeszkedés nyer" típusú keresésnek. Amennyiben a csomag egyetlen szabályra sem illeszkedik, akkor az IPFW 65535-ös sorszámú állandó szabálya fogja elcsípni, amely feladata szerint eldobja az összes hozzá beérkezõ csomagot anélkül, hogy bármit is válaszolna a csomag feladójának.

A keresés a `count`, `skipto` és `tee` szabályok után még folytatódik.

Az itt szereplõ utasítások különbözõ állapottartásra vonatkozó opciókat, például a `keep state`, `limit`, `in`, `out` és `via` kulcsszavakat tartalmazó szabályokon alapulnak. Lényegében ezt tekinthetjük az inkluzív típusú tûzfalak kiindulási alapjaként.

A tûzfal szabályainak beállítása során nem árt óvatosnak lennünk, mert figyelmetlenségünk révén könnyen kizárathatjuk magunkat a gépünkrõl.

A szabályok felépítése

Az itt bemutatásra kerülõ szabályok felépítését csak olyan mértékig részletezzük, ami elengedõ a szabványos inkluzív típusú tûzfalak kialakításához. A szabályok felépítésének pontos leírását az ipfw(8) man oldalán találhatjuk meg.

A szabályok kulcsszavakat tartalmaznak. Ezeket a kulcsszavakat soronként egy elõre rögzített sorrendben kell szerepeltetni. A kulcsszavakat a szövegben kiemeltük. Bizonyos kulcsszavakhoz további opciókhoz is tartozhatnak, amelyek gyakran maguk is kulcsszavak és szintén további opciókat tartalmazhatnak.

A `#` egy megjegyzés kezdetét jelzi, mely egyaránt megjelenhet egy külön sorban, vagy egy szabályt tartalmazó sor végén. Az üres sorok nem vesznek részt a feldolgozásban.

PARANCS SZABÁLY_SZÁM CSELEKVÉS NAPLÓZÁS SZûRÉS ÁLLAPOTTARTÁS

PARANCS

Minden új szabály elõttt az add (mint hozzáadás) parancsnak kell szerepelni, amellyel a belsõ táblázatba tudjuk felvenni.

SZABÁLY_SZÁM

A szabályokhoz mindig tartozik egy sorszám is.

CSELEKVÉS

A szabályhoz az alábbi cselekvések valamelyike kapcsolható, amely akkor hajtódik végre, amikor a csomag megfelel a hozzá tartozó szûrési feltételeknek.

allow | accept | pass | permit

A fentiek közül mindegyik ugyanazt jelenti, vagyis hatásukra az illeszkedõ csomag kilép a tûzfalból. Ez a szabály megállítja a keresést.

check-state

A csomagot a dinamikus szabályokat tároló táblázattal veti össze. Ha itt egyezést talál, akkor végrehajtja az egyezõ dinamikus szabályhoz tartozó cselekvést, minden más esetben továbblép a következõ szabályra. Ennek a szabálynak nincs illeszthetõ paramétere. Ha a szabályrendszerben nem szerepel ilyen, akkor a dinamikus szabályok vizsgálatát az elsõ `keep-state` vagy `limit` használatánál vonja be a rendszer.

deny | drop

Mind a két szó ugyanarra utal, vagyis a szabályra illeszkedõ csomagokat el kell dobni. Ebben az esetben a keresés befejezõdik.

NAPLÓZÁS

log vagy logamount

Amikor egy csomag egy `log` kulcsszót tartalmazó szabályra illeszkedik, akkor a rendszernaplóban egy üzenet keletkezik a `security` (biztonság) funkción keresztül. A naplóba ténylegesen csak akkor kerül bele az üzenet, ha az adott szabály még nem haladta meg a hozzá tartozó `logamount` paraméter értékét. Ha ezt nem adtuk meg, akkor az itt érvényes korlát a `net.inet.ip.fw.verbose_limit` sysctl változóból fog származni. A nulla érték mind a két esetben megszünteti ezt a korlátozást. Ha elértük a korlátot, akkor a naplózást úgy tudjuk újra engedélyezni, ha töröljük a naplózáshoz tartozó számláló értékét, lásd az `ipfw reset log` parancsot.

A naplózás mindig az összes paraméter illeszkedésének ellenõrzése után történik, de még a cselekvés (accept, deny) elvégzése elõtt. Teljesen rajtunk múlik, hogyan milyen szabályokat naplózunk.

SZûRÉS

Ebben a szakaszban azok a kulcsszavak találhatóak, amelyek segítségével a csomagok különbözõ

tulajdonságait tudjuk megvizsgálni és eldönteni, hogy illeszkedik-e a szabályra vagy sem. A következõ általános tulajdonságokat tudjuk megvizsgálni, ebben a kötött sorrendben:

udp | tcp | icmp

Bármilyen más olyan protokoll is megadható, amely megtalálható az /etc/protocols állományban. Ezzel adjuk a csomaghoz tartozó protokollt. Használata kötelezõ.

from *forrás* to *cél*

Mind a `from` és `to` kulcsszavak IP-címek illesztésére alkalmasak. A szabályoknak tartalmazniuk kell a *forrás* ÉS a *cél* paramétereket is. Az `any` egy olyan kulcsszó, amely tetszõleges IP-címre illeszkedik. A `me` pedig egy olyan speciális kulcsszó, amely a tûzfalat mûködtetõ FreeBSD-s gép (tehát ez a gép) adott interfészhez tartozó IP-címét jelöli, mint ahogy a `from me to any`, `from any to me`, `from 0.0.0.0/0 to any`, `from any to 0.0.0.0/0`, `from 0.0.0.0 to any`, `from any to 0.0.0.0` vagy `from me to 0.0.0.0` paraméterekben. Az IP-címek numerikus pontozott formában a hálózati maszk hosszával együtt (CIDR-jelöléssel), vagy egyszerûen csak pontozott formában adhatóak meg. A hálózati maszkok megállapításában a net-mgmt/ipcalc port lehet segítségünkre. Errõl bõvebb információkat a segédprogram honlapján, a http://jodies.de/ipcalc címen találhatunk (angolul).

port *szám*

A portszámokat is ismerõ protokollok esetében (mint például a TCP vagy UDP) adhatjuk meg. Fontos, hogy itt annak a szolgáltatásnak a portszámát adjuk meg, amelyre a szabály vonatkozik. A szolgáltatás (az /etc/services állományból származó) nevét is megadhatjuk a port száma helyett.

in | out

A beérkezõ valamint a kimenõ csomagokat adhatjuk meg ezen a módon. Itt az `in` és `out` kulcsszavak, melyeket kötelezõ megadni a szabály részeként.

via *interfész*

Név szerint az adott interfészen keresztül haladó csomagokat tudjuk szûrni. A `via` kulcsszó hatására a használt interfész is számítani fog a csomag feldolgozása során.

setup

Ez a kulcsszó a TCP csomagok esetében a kapcsolatok felépítésére vonatkozó kéréseket segít beazonosítani.

keep-state

Ez egy kötelezõ kulcsszó. Feldolgozásakor a tûzfal létrehoz dinamikus szabályt, amely alapértelmezés szerint az egyazon protokollt használó forrás és cél IP/port párosok közti kétirányú forgalomra fog automatikusan illeszkedni.

limit {*forráscím* | *forrásport* | *célcím* | *célport*}

A tûzfal csak *N* darab, a szabálynak megfelelõ azonos paraméterû kapcsolatot fog átengedi. Itt egy vagy több forrás- és célcím valamint forrás- és célport adható meg. A `limit` és a `keep-state` egy

szabályon belül nem használható. A `limit` ugyanazokat az állapottartó funkciókat képviseli, mint a `keep-state`, csak a saját kiegészítéseivel megtoldva.

ÁLLAPOTTARTÁS

Az állapottartó szûrés a kétirányú csomagváltásokat egy létrejött kapcsolatba sorolja. Olyan vizsgálatokat végez, amivel képes megállapítani, hogy a csomag küldõje és címzettje között kialakult kommunikáció követ-e valamilyen kétirányú csomagküldésre érvényes folyamatot. Az így felállított sablontól eltérõ összes csomag hamisnak minõsül és automatikusan eldobásra kerül.

A `check-state` segítségével ellenõrizhetjük, hogy az adott csomag a IPFW szerint megfelel-e valamelyik dinamikusan leképzett szabálynak. Ha egyezik valamelyikõjükkel, akkor a csomag a tûzfalból kilépve folytatja útját és a kommunikációban soron következõ csomag számára létrejön egy másik dinamikus szabály. Ha nincs egyezés, akkor csomag feldolgozása a szabályrendszer következõ szabályánál folytatódik.

A dinamikus szabályokat kezelõ rutin sebezhetõ, mivel ha egyszerre nagy mennyiségû SYN csomagot küldünk, akkor olyan sok dinamikus bejegyzés keletkezik, hogy egyszerûen kifogyunk a rendelkezésre álló erõforrásokból. A FreeBSD fejlesztõi azonban az ilyen természetû támadások kivédésére is felkészítették, és kialakították belõle a `limit` opciót. Alkalmazásával le tudjuk korlátozni az egyszerre folyó párhuzamos kapcsolatok számát a forrás vagy a cél a `limit` paraméternél megadott mezõinek és a csomag IP-címe alapján. Így az adott szabályhoz és IP-címhez csak elõre rögzített mennyiségû nyitott állapotú dinamikus szabály létezhet egy idõben. Ha ezt a korlátot átlépjük, a csomag eldobódik.

A tûzfal üzeneteinek naplózása

A naplózás elõnyei nyilvánvalóak. Ha engedélyezzük, aktiválása után képesek leszünk olyan információknak utánanézni, mint például milyen csomagokat dobtunk el, honnan érkeztek, hova tartottak. Ez egy komoly fegyverünk lehet a potenciális támadókkal szemben.

Azonban hiába engedélyezzünk önmagában a naplózást, attól az IPFW még saját magától nem fog naplózást elõíró szabályokat gyártani. A tûzfal karbantartóinak maguknak kell eldöntenie, hogy a szabályrendszerben mely szabályokhoz tartozzon naplózás, nekik kell felvenni ezekhez a `log` kulcsszót. Általában csak az eldobással járó `deny` típusú szabályokat vagy a bejövõ ICMP pingeket szokták naplózni. Gyakran úgy oldják meg ezt, hogy a szabályrendszer utolsó szabályaként lemásolják az `ipfw` alapértelmezett "mindent eldobunk" szabályát és a naplózást adják meg benne. Ezen a módon fény derül azokra a csomagokra, amelyek a szabályrendszerben semmire sem illeszkedtek.

A naplózás azonban egy kétélû fegyver, mivel ha nem vagyunk elég körültekintõek, akkor a sok naplóinformáció között könnyen el tudunk veszni és a lemezünk is gyorsan betelhet a mindent elfoglaló naplóktól. Mellesleg a naplók megdagasztását célzó DoS típusú támadás a rendszerek lebénítására alkalmazott egyik legõsibb technika. Ezek az üzenetek nem csak a rendszernaplóba kerülnek bele, hanem az elsõdleges konzol képernyõjére is kiíródnak, ami egy idõ után idegesítõ tud lenni.

A rendszermag `IPFIREWALL_VERBOSE_LIMIT=5` beállításával azonban képesek vagyunk korlátozni azokat a rendszernapló felé küldött egymás után következõ üzeneteket, amelyek ugyanarra a

szabályra vonatkoznak. Amikor ezt a beállítást megadjuk a rendszermag fordításánál, akkor az egyes szabályokhoz az általa meghatározott értéken felül nem jön létre több hasonló üzenet. Hiszen semmi sem derül ki 200 teljesen azonos naplóüzenetbõl. Például, ha az egyes szabályokhoz legfeljebb öt egymást követõ üzenetet engedélyezünk, akkor a többi fennmaradó azonos üzenetet összeszámolja a rendszer és a következõ módon közvetíti a rendszernaplózó szolgáltatás felé:

```
last message repeated 45 times
```

Ami magyarul így hangzik:

```
az utolsó üzenet 45 alkalommal ismétlõdött meg
```

Az összes csomagokkal kapcsolatos naplózás alapértelmezés szerint a /var/log/security állományba kerül, amelyet az /etc/syslog.conf állomány definiál.

Szabályokat tartalmazó szkript készítése

A rutinosabb IPFW felhasználók a szabályokat egy állományban programozzák le olyan stílusban, hogy szkriptként is futtatható legyen. Ennek az egyik legnagyobb elõnye, hogy a tûzfal szabályai így egyszerre cserélhetõek a rendszer újraindítása nélkül. Ez a módszer nagyon kényelmes az új szabályok kipróbálásánál, mivel tetszõleges alkalommal végrehajthatjuk. Mivel ez egy szkript, ki tudjuk használni az itt megszokott szimbolikus helyettesítés által felkínált lehetõségeket, és ezzel a gyakran használt értékeket is egyszerre több szabályban tudjuk helyettesíteni. Erre a következõkben fogunk egy konkrét példát látni.

A szkript felépítése kompatibilis a sh(1), csh(1) és tcsh(1) parancsértelmezõkkel. A szimbolikus mezõk helyettesítését a $ vagyis dollárjel vezeti be. Maguk a szimbolikus mezõk nem tartalmazzák a $ elõtagot. A szimbolikus mezõk értékeit "kettõs idézõjelek" között kell megadni.

A szabályok összeírását kezdjük el így:

```
####### itt kezdõdik az ipfw szabályait tartalmazó szkript ######
#
ipfw -q -f flush       # töröljük az összes aktuális szabályt
# Set defaults
oif="tun0"             # a kimenõ interfész
odns="192.0.2.11"      # az internet szolgáltató névszerverének IP-címe
cmd="ipfw -q add "     # a szabályok hozzáadásához szükséges elemek
ks="keep-state"        # csupán a lustaság miatt
$cmd 00500 check-state
$cmd 00502 deny all from any to any frag
$cmd 00501 deny tcp from any to any established
$cmd 00600 allow tcp from any to any 80 out via $oif setup $ks
$cmd 00610 allow tcp from any to $odns 53 out via $oif setup $ks
$cmd 00611 allow udp from any to $odns 53 out via $oif $ks
#### itt fejezõdik be az ipfw szabályait tartalmazó szkript ######
```

Ezzel készen is vagyunk. Most ne törõdjünk a példában szereplõ szabályokkal, itt most a szimbolikus helyettesítés használatát igyekeztük bemutatni.

Ha az iménti példát az /etc/ipfw.rules állományba mentettük el, akkor az alábbi parancs kiadásával tudjuk újratölteni a benne szereplõ szabályokat:

```
# sh /etc/ipfw.rules
```

Az /etc/ipfw.rules állományt egyébként tetszõleges néven hívhatjuk és bárhová rakhatjuk.

Ugyanez természetesen elérhetõ a következõ parancsok egymás utáni begépelésével is:

```
# ipfw -q -f flush
# ipfw -q add check-state
# ipfw -q add deny all from any to any frag
# ipfw -q add deny tcp from any to any established
# ipfw -q add allow tcp from any to any 80 out via tun0 setup keep-state
# ipfw -q add allow tcp from any to 192.0.2.11 53 out via tun0 setup keep-state
# ipfw -q add 00611 allow udp from any to 192.0.2.11 53 out via tun0 keep-state
```

Állapottartó szabályrendszerek

A most következõ címfordítás nélküli szabályrendszer arra mutat példát, hogyan valósítsunk meg egy biztonságos "inkluzív" tûzfalat. Az inkluzív tûzfalak csak a szabályainak megfelelõ szolgáltatásokat engedik át, minden mást alapértelmezés szerint tiltanak. A komplett hálózati szegmensek védelmére összeállított tûzfalaknak legalább két interfészük van, amelyek mindegyikéhez tartoznia kell szabályoknak a megfelelõ mûködéshez.

Az UNIX® mintájú operációs rendszer, köztül a FreeBSD is olyan, hogy a rendszerben belüli kommunikációt a lo0 nevû interfészen és a `127.0.0.1` IP-címen bonyolítja le. A tûzfalban mindenképpen szerepelniük kell olyan szabályoknak, amelyek gondoskodnak ezen speciális belsõ csomagok zavartalan közlekedésérõl.

Az internet felé csatlakozó interfész lesz az, amelyen keresztül a kifelé menõ kéréseket hitelesítjük és vezéreljük az internet elérését, valamint ahol szûrjük az internet felõl érkezõ kéréseket. Ez lehet a PPP esetében a tun0 eszköz, vagy a DSL-, illetve kábelmodemhez csatlakozó hálózati kártya.

Abban az esetben, amikor egy vagy több hálózati kártyával csatlakozunk a tûzfal mögött található belsõ helyi hálózatra, szintén gondoskodnunk kell a helyi hálózaton belül mozgó csomagok akadálymentes továbbításáról.

A szabályokat elõször három nagyobb osztályba kell sorolnunk: az összes szabadon forgalmazó interfész, a publikus kimenõ és a publikus bejövõ interfész csoportjába.

A publikus interfészekhez tartozó csoportokban úgy kell rendeznünk a szabályokat, hogy elõre kerüljenek a gyakrabban használtak és hátra a kevésbé használtak, valamint a csoportok utolsó szabálya blokkoljon és naplózzon minden csomagot az adott interfészen és irányban.

A következõ szabályrendszerben szereplõ, a kimenõ kapcsolatokat tartalmazó csoport csak olyan `allow` típusú szabályokat tartalmaz, amelyek szûrési feltételei egyértelmûen azonosítják az interneten elérhetõ szolgáltatásokat. Az összes szabályban megjelennek a `proto`, `port`, `in`/`out`, `via` és `keep state` opciók. A `proto tcp` szabályokban emellett szerepel még egy `setup` opció is, amellyel a kapcsolatokat kezdeményezõ csomagokat tudjuk azonosítani és felvenni az állapottartásért felelõs dinamikus szabályok közé.

A bejövõ forgalmat vezérlõ szabályrendszerben elõször az eldobni kívánt csomagokat kell megadni, aminek két eltérõ oka van. Elõször is elõfordulhat, hogy a veszélyes csomagok részleges illeszkedés miatt szabályosnak tûnnek. Az ilyen csomagokat értelemszerûen nem lenne szabad beengedni a szabályok részleges megfelelése alapján. A másodszor az eleve ismerten problémás és értelmetlen csomagokat csendben el kellene vetni, mielõtt a szakaszhoz tartozó utolsó szabály fogná meg és naplózná. Ez az utolsó szabály egyébként szükség esetén felhasználható a támadók elleni bizonyítékok begyûjtésére.

A másik, amire még oda kell figyelnünk, hogy a blokkolt csomagok esetében semmilyen válasz nem keletkezzen, egyszerûen csak tûnjenek el. Így a támadó nem fogja tudni, hogy a csomagjai vajon elérték-e a rendszerünket. Minél kevesebb információt tudnak összegyûjteni a rendszerünkrõl a támadók, annál biztonságosabbnak tekinthetõ. Amikor ismeretlen portokra érkezõ csomagokat naplózunk, érdemes az /etc/services/ állományban vagy http://www.securitystats.com/tools/portsearch.php címen (angolul) utánanézni a porthoz tartozó szolgáltatásnak. A különbözõ trójai programok által portok számai ezen a linken érhetõek el (angolul): http://www.simovits.com/trojans/trojans.html.

Példa egy inkluzív szabályrendszerre

A most következõ, címfordítást nem tartalmazó szabályrendszer teljesen inkluzív típusú. Éles rendszereken is nyugodtan alkalmazhatjuk. Egyszerûen csak annyit kell tennünk, hogy megjegyzésbe tesszük az olyan szolgáltatásokra vonatkozó szabályokat, amelyeket nem akarunk engedélyezni. Amikor pedig olyan üzenetek jelennek meg a naplóban, amelyeket nem akarunk tovább látni, a bejövõ kapcsolatokhoz vegyünk fel egy `deny` típusú szabályt hozzájuk. Minden szabályban cseréljük ki a `dc0` interfészt arra a hálózati kártyára, amely közvetlenül csatlakoztatja rendszerünket az internethez. A felhasználói PPP esetében ez a `tun0`.

A szabályok használatában felfedezhetünk egyfajta rendszerszerûséget:

- Mindegyik sorban, ahol az internet felé nyitunk meg egy kapcsolatot, a `keep-state` opciót használjuk.
- Az internetrõl az összes hitelesített szolgáltatás elérése tartalmazza a `limit` opciót az elárasztások kivédése miatt.
- Az összes szabályban az `in` vagy az `out` paraméterrel megadjuk szûrni kívánt forgalom irányát.
- Az összes szabályban szerepel a `via` paraméterrel a csomagokat továbbító interfész neve.

Az alábbi szabályokat tegyük az /etc/ipfw.rules állományba.

```
############### Itt kezdõdnek az IPFW szabályai ############################
# Kezdés elõtt töröljük az összes aktív szabályt.
ipfw -q -f flush
```

```
# Állítsuk be a parancsok további szükséges opciót.
cmd="ipfw -q add"
pif="dc0"     # az internethez csatlakozó
              # interfész neve

###############################################################
# A belsõ hálózat számára ne korlátozzunk semmit se.
# Ha nincs helyi hálózatunk, akkor erre nincs szükségünk.
# Az 'xl0' nevét írjuk át a helyi hálózatra csatlakozó
# interfész nevére.
###############################################################
#$cmd 00005 allow all from any to any via xl0

###############################################################
# A rendszer belsõ interfészét se szûrjük.
###############################################################
$cmd 00010 allow all from any to any via lo0

###############################################################
# A csomagot engedjük át a tûzfalon, ha korábban már felvettünk
# hozzá egy dinamikus szabályt a keep-state opcióval.
###############################################################
$cmd 00015 check-state

###############################################################
# Az internet felé forgalmazó interfész (kimenõ kapcsolatok)
# A saját hálózatunkról belülrõl vagy errõl az átjáróról
# kezdeményezett kapcsolatokat vizsgáljuk az internet felé.
###############################################################

# Kifelé engedélyezzük az internet-szolgáltatónk névszerverének
# elérését. Az x.x.x.x a szolgáltatónk névszerverének IP-címe
# legyen. Ha a szolgáltatónak több névszervere is van, akkor
# másoljuk le ezeket a sorokat és az /etc/resolv.conf
# állományban található IP-címeket helyettesítsük be.
$cmd 00110 allow tcp from any to x.x.x.x 53 out via $pif setup keep-state
$cmd 00111 allow udp from any to x.x.x.x 53 out via $pif keep-state

# Kábel/DSL konfigurációk esetében kifelé engedélyezzük a
# szolgáltatónk DHCP szerverének elérését. Ha a "felhasználói
# PPP"-t használjuk, akkor erre nem lesz szükségünk, az egész
# csoportot törölhetjük. Az alábbi szabállyal csíphetjük el a
# beírandó IP-címet. Ha a naplóban megtaláltuk, akkor vegyük
# ki az elsõ szabályt, a másodikba írjuk bele a címet és
# engedélyezzük.
$cmd 00120 allow log udp from any to any 67 out via $pif keep-state
#$cmd 00120 allow udp from any to x.x.x.x 67 out via $pif keep-state

# Kifelé engedélyezzük a szabvány nem biztonságos WWW
# funkció elérését.
```

```
$cmd 00200 allow tcp from any to any 80 out via $pif setup keep-state

# Kifelé engedélyezzük a biztonságos HTTPS funkció
# elérését TLS SSL használatával.
$cmd 00220 allow tcp from any to any 443 out via $pif setup keep-state

# Kifelé engedélyezzük a e-mailek küldését és fogadását.
$cmd 00230 allow tcp from any to any 25 out via $pif setup keep-state
$cmd 00231 allow tcp from any to any 110 out via $pif setup keep-state

# Kifelé engedélyezzük a FreeBSD (a make install és a CVSUP)
# funkcióit. Ezzel lényegében a rendszeradminisztrátornak
# ,,ISTENI'' jogokat adunk.
$cmd 00240 allow tcp from me to any out via $pif setup keep-state uid root

# Kifelé engedélyezzük a pinget.
$cmd 00250 allow icmp from any to any out via $pif keep-state

# Kifelé engedélyezzük az idõ szolgáltatást.
$cmd 00260 allow tcp from any to any 37 out via $pif setup keep-state

# Kifelé engedélyezzük az nntp news szolgáltatást
# (vagyis a hírcsoportokat)
$cmd 00270 allow tcp from any to any 119 out via $pif setup keep-state

# Kifelé engedélyezzük a biztonságos FTP, telnet és SCP
# elérését az SSH (secure shell) használatával.
$cmd 00280 allow tcp from any to any 22 out via $pif setup keep-state

# Kifelé engedélyezzük a whois szolgáltatást.
$cmd 00290 allow tcp from any to any 43 out via $pif setup keep-state

# Dobjuk el és naplózzunk mindent, ami megpróbál kijutni.
# Ez a szabály gondoskodik róla, hogy alapértelmezés szerint
# mindent blokkoljunk.
$cmd 00299 deny log all from any to any out via $pif

######################################################################
# Az internet felõli interfész (bejövõ kapcsolatok)
# A saját hálózatunk felé vagy erre az átjáróra
# nyitott kapcsolatokat vizsgáljuk az internet felõl.
######################################################################

# Blokkoljunk minden olyan bejövõ forgalmat, amely a fenntartott
# címtartományok felé tart.
$cmd 00300 deny all from 192.168.0.0/16 to any in via $pif  #RFC 1918: privát IP
$cmd 00301 deny all from 172.16.0.0/12 to any in via $pif   #RFC 1918: privát IP
$cmd 00302 deny all from 10.0.0.0/8 to any in via $pif      #RFC 1918: privát IP
$cmd 00303 deny all from 127.0.0.0/8 to any in via $pif     #helyi
$cmd 00304 deny all from 0.0.0.0/8 to any in via $pif       #helyi
$cmd 00305 deny all from 169.254.0.0/16 to any in via $pif  #DHCP
```

```
$cmd 00306 deny all from 192.0.2.0/24 to any in via $pif    #dokumentációs célokra
fenntartott
$cmd 00307 deny all from 204.152.64.0/23 to any in via $pif #Sun klaszterek
összekötésére használt
$cmd 00308 deny all from 224.0.0.0/3 to any in via $pif     #D és E osztályú multicast

# A nyilvános pingek tiltása.
$cmd 00310 deny icmp from any to any in via $pif

# Az ident szolgáltatás tiltása.
$cmd 00315 deny tcp from any to any 113 in via $pif

# Blokkoljuk az összes Netbios szolgáltatást: 137=név, 138=datagram,
# 139=session. A Netbios az MS Windows megosztását implementálja.
# Blokkoljuk az MS Windows hosts2 névszerver kéréseit is a 81-es
# porton.
$cmd 00320 deny tcp from any to any 137 in via $pif
$cmd 00321 deny tcp from any to any 138 in via $pif
$cmd 00322 deny tcp from any to any 139 in via $pif
$cmd 00323 deny tcp from any to any 81 in via $pif

# Eldobjuk az összes késõn érkezõ csomagot.
$cmd 00330 deny all from any to any frag in via $pif

# Eldobjuk azokat az ACK csomagokat, amelyek egyik dinamikus
# szabálynak sem felelnek meg.
$cmd 00332 deny tcp from any to any established in via $pif

# Befelé engedélyezzük a szolgáltató DHCP szerverének válaszát. Ebben
# a szabályban csak a DHCP szerver IP-címe szerepelhet, mivel ez az
# egyetlen olyan hitelesített forrás, ami ilyen csomagokat küldhet.
# Ez csak a kábeles és DSL típusú kapcsolatok esetében szükséges.
# Amikor a "felhasználói PPP"-vel csatlakozunk az internethez, nem
# kell ez a szabály. Ugyanazt az IP-címet kell megadnunk, amelyet a
# kimenõ kapcsolatoknál is.
#$cmd 00360 allow udp from any to x.x.x.x 67 in via $pif keep-state

# Befelé engedélyezzük a szabvány WWW funkciót, mivel webszerverünk
# is van.
$cmd 00400 allow tcp from any to me 80 in via $pif setup limit src-addr 2

# Befelé engedélyezzük a biztonságos FTP, telnet és SCP
# típusú kapcsolatokat az internetrõl.
$cmd 00410 allow tcp from any to me 22 in via $pif setup limit src-addr 2

# Befelé engedélyezzük az internetrõl érkezõ nem biztonságos telnet
# kapcsolatokat. Azért tekintjük nem biztonságosnak, mert az
# azonosítók és a jelszavak az interneten titkosítatlanul vándorolnak.
# Töröljük ezt a csoportot, ha nincs telnet szolgáltatásunk.
$cmd 00420 allow tcp from any to me 23 in via $pif setup limit src-addr 2
```

```
# Dobjuk el és naplózzuk az összes többi kintrõl érkezõ csomagot.
$cmd 00499 deny log all from any to any in via $pif

# Alapértelmezés szerint dobjuk el mindent. Az ide érkezõ
# csomagokat is naplózzuk, amibõl többet is ki tudunk majd
# deríteni.
$cmd 00999 deny log all from any to any
############## Itt fejezõdnek be az IPFW szabályai ######################
```

Példa hálózati címfordításra és állapottartásra

Az IPFW címfordító funkciójának kihasználásához további konfigurációs beállítások alkalmazására is szükségünk lesz. A rendszermagban opció között meg kell adnunk az `option IPDIVERT` sort a többi `IPFIREWALL` sor mellett, és fordítanunk egy saját verziót.

Emellett még az /etc/rc.conf állományban is engedélyezni kell az IPFW alapvetõ funkcióit.

```
natd_enable="YES"                   # engedélyezzük a címfordításért felelõs démont
natd_interface="rl0"                # az internet felé mutató hálózati kártya neve
natd_flags="-dynamic -m"            # -m = a portszámok megtartása, ha lehetséges
```

Az állapottartó szabályok használata a `divert natd` címfordítási opcióval együtt nagyban növeli a szabályrendszer leprogramozásának bonyolultságát. A `check-state` és `divert natd` szabályok helye kritikus a megfelelõ mûködés tekintetében. Az eddig megszokott egyszerû viselkedés itt már nem érvényesül. Bevezetünk egy új cselekvést is, amelynek a neve `skipto`. A `skipto` parancs használatához elengedhetetlen a szabályok sorszámozása, mivel pontosan tudnunk kell, hogy a `skipto` hatására hova kell ugrania a vezérlésnek.

A következõ példában nem fogunk sok megjegyzést látni, mivel benne az egyik lehetséges programozási stílust próbáljuk érzékeltetni és a csomagok szabályrendszerek közti áramlását magyarázzuk.

A feldolgozás a szabályokat tartalmazó állomány tetején található elsõ szabállyal kezdõdik, és innen egyesével pereg végig lefelé a feldolgozás egészen addig, amíg a csomag a szûrési feltételek valamelyikének eleget nem tesz és távozik a tûzfalból. Leginkább a 100-as, 101-es, 450-es, 500-as és 510-es sorszámú szabályokat emelnénk ki. Ezek vezérlik kimenõ és bejövõ csomagok fordítását, ezért a hozzájuk tartozó dinamikus állapottartó bejegyzések mindig a helyi hálózat IP-címeire hivatkoznak. Amit még érdemes megfigyelnünk, hogy az összes áteresztõ és eldobó szabályban szerepel a csomag haladási iránya (tehát kimenõ vagy éppen bejövõ) és az érintett interfészt megnevezése. Emellett azt is vegyük észre, hogy az összes kifelé irányuló kapcsolatlétrehozási kérés az 500-as sorszámú szabályhoz fog ugrani a címfordítás elvégzéséhez.

Tegyük fel, hogy a helyi hálózatunkon levõ felhasználók szeretnek honlapokat nézgetni az interneten. A honlapok a 80-as porton keresztül kommunikálnak. Tehát amikor egy ilyen csomag eléri a tûzfalat, nem fog illeszkedni a 100-as szabályra, mert a fejléce szerint kifelé halad és nem befelé. A 101-es szabályon is átlép, mivel ez az elsõ csomag, így a dinamikus állapottartó táblázatban sem szerepel még. A csomag végül a 125-ös szabályra fog illeszkedni: kifelé halad az internetre csatlakozó hálózati kártyán. A csomagban azonban még mindig az eredeti forrás IP-címe

található, amely a helyi hálózat egyik gépére hivatkozik. A szabály illeszkedésekor két cselekvés is végbemegy. A `keep-state` opció hatására ez a szabály felveszi ezt a kapcsolatot az állapottartó dinamikus szabályok közé és végrehajtja a másik megadott feladatot. Ez a feladat része a dinamikus táblázatba rögzített bejegyzésnek, ami ebben az esetben a `skipto 500` ("ugorjunk az 500-as szabályra") lesz. Az 500-as szabály a továbbküldés elõtt lefordítja a csomag forrás IP-címét. Ezt ne felejtsük el, nagyon fontos! A csomag ezután eljut a céljához, és visszatérve ismét belép a szabályrendszer tetején. Ezúttal illeszkedni fog a 100-as szabályra és a cél IP-címét visszafordítjuk a helyi hálózatunk megfelelõ gépének címére. Ezután a `check-state` szabályhoz kerül, amely megtalálja a dinamikus szabályok között és továbbengedi a belsõ hálózatra. Ezzel visszakerül a küldõ géphez, amely egy újabb csomagot küld egy újabb adatszeletet kérve a távoli szervertõl. Ekkor már a `check-state` szabály megtalálja a hozzá tartozó bejegyzést a dinamikus szabályok között és végrehajtódik a korábban letárolt `skipto 500` mûvelet. A csomag erre az 500-as szabályra ugrik, ahol lefordítjuk a címét és továbbküldjük.

Az bejövõ oldalon minden, ami egy korábban kialakult kapcsolat részeként érkezik, automatikusan a `check-state` és a megfelelõ helyre rakott `divert natd` szabályok által dolgozódik fel. Itt mindössze a rossz csomagok eldobásával és a hitelesített szolgáltatások elérésének biztosításával kell foglalkoznunk. Például a tûzfalon egy webszerver fut, és azt szeretnénk, hogy az internetrõl képesek legyenek elérni a rajta levõ oldalakat. Az újonnan beérkezõ kapcsolatépítési kérelem a 100-as szabályra fog illeszkedni, amelynek a cél IP-címét a tûzfal helyi hálózaton található címére fogjuk leképezni. A csomagot ezután még megvizsgáljuk, nem tartalmaz-e valamilyen huncutságot, majd végül a 425-ös szabálynál fog kikötni. Az egyezéskor két dolog történhet: a csomaghoz felveszünk egy dinamikus szabályt, de ezúttal az adott forrás IP-címrõl érkezõ kapcsolatkérések számát 2-re lekorlátozzuk. Ezzel az adott szolgáltatás portján meg tudjuk óvni a tûzfalat üzemeltetõ gépet a DoS típusú támadásoktól. A csomagot ezután hozzá tartozó cselekvés szerint továbbengedjük a belsõ hálózat felé. Visszatéréskor a tûzfal felismeri, hogy a csomag egy már meglevõ kapcsolathoz tartozik, ezért közvetlenül az 500-as szabályhoz kerül címfordításra, majd a kimenõ interfészen keresztül továbbküldjük.

Íme az elsõ példa egy ilyen szabályrendszerre:

```
#!/bin/sh
cmd="ipfw -q add"
skip="skipto 500"
pif=rl0
ks="keep-state"
good_tcpo="22,25,37,43,53,80,443,110,119"

ipfw -q -f flush

$cmd 002 allow all from any to any via xl0  # nem szûrjük a belsõ hálózatot
$cmd 003 allow all from any to any via lo0  # nem szûrjük a helyi interfészt

$cmd 100 divert natd ip from any to any in via $pif
$cmd 101 check-state

# A kimenõ csomagok hitelesítése:
$cmd 120 $skip udp from any to xx.168.240.2 53 out via $pif $ks
$cmd 121 $skip udp from any to xx.168.240.5 53 out via $pif $ks
```

```
$cmd 125 $skip tcp from any to any $good_tcpo out via $pif setup $ks
$cmd 130 $skip icmp from any to any out via $pif $ks
$cmd 135 $skip udp from any to any 123 out via $pif $ks

# Az összes olyan csomagot eldobjuk, amely a fenntartott
# címtartományokba tart:
$cmd 300 deny all from 192.168.0.0/16  to any in via $pif  #RFC 1918: privát IP
$cmd 301 deny all from 172.16.0.0/12   to any in via $pif  #RFC 1918: privát IP
$cmd 302 deny all from 10.0.0.0/8      to any in via $pif  #RFC 1918: privát IP
$cmd 303 deny all from 127.0.0.0/8     to any in via $pif  #helyi
$cmd 304 deny all from 0.0.0.0/8       to any in via $pif  #helyi
$cmd 305 deny all from 169.254.0.0/16  to any in via $pif  #DHCP
$cmd 306 deny all from 192.0.2.0/24    to any in via $pif  #dokumentációs célokra
fenntartott
$cmd 307 deny all from 204.152.64.0/23 to any in via $pif  #Sun klaszter
$cmd 308 deny all from 224.0.0.0/3     to any in via $pif  #D és E osztályú multicast

# Az érkezõ csomagok hitelesítése:
$cmd 400 allow udp from xx.70.207.54 to any 68 in $ks
$cmd 420 allow tcp from any to me 80 in via $pif setup limit src-addr 1

$cmd 450 deny log ip from any to any

# Ide ugrunk a kimenõ állapottartó szabályoknál:
$cmd 500 divert natd ip from any to any out via $pif
$cmd 510 allow ip from any to any

####################### a szabályok vége ##################
```

A következõ példa teljesen megegyezik az elõzõvel, azonban itt már dokumentációs szándékkal szerepelnek megjegyzések is, melyek a tapasztalatlan IPFW szabályíróknak segítik jobban megérteni a szabályok pontos mûködését.

A második példa:

```
#!/bin/sh
############## Az IPFW szabályai itt kezdõdnek ############################
# Kezdés elõtt töröljük az összes jelenleg aktív szabályt:
ipfw -q -f flush

# Beállítjuk a parancsok megfelelõ elõtagjait:
cmd="ipfw -q add"
skip="skipto 800"
pif="rl0"     # az internethez csatlakozó
              # hálózati interfész neve

########################################################################
# A belsõ hálózat számára ne korlátozzunk semmit se.
# Ha nincs helyi hálózatunk, akkor erre nincs szükségünk.
# Az 'xl0' nevét írjuk át a helyi hálózatra csatlakozó
```

```
# interfész nevére.
#####################################################################
$cmd 005 allow all from any to any via xl0

#####################################################################
# A rendszer belsõ interfészét se szûrjük.
#####################################################################
$cmd 010 allow all from any to any via lo0

#####################################################################
# Ellenõrizzük, hogy ez egy beérkezõ csomag és ha igen, akkor
# fordítsuk a címét.
#####################################################################
$cmd 014 divert natd ip from any to any in via $pif

#####################################################################
# Ha ehhez a csomaghoz korábban már vettük fel dinamikus
# szabályt a keep-state opció révén, akkor engedjük tovább.
#####################################################################
$cmd 015 check-state

#####################################################################
# Az internet felé forgalmazó interfész (kimenõ kapcsolatok)
# A saját hálózatunkról belülrõl vagy errõl az átjáróról
# kezdeményezett kapcsolatokat vizsgáljuk az internet felé.
#####################################################################

# Kifelé engedélyezzük az internet-szolgáltatónk névszerverének
# elérését. Az x.x.x.x a szolgáltató névszerverének IP-címe
# lesz. Ha a szolgáltatónknak több névszervere is van, akkor
# az /etc/resolv.conf állományból nézzük ki a címeiket és
# másoljuk le az alábbi sor mindegyikükhöz.
$cmd 020 $skip tcp from any to x.x.x.x 53 out via $pif setup keep-state

# A kábeles és DSL kapcsolatok esetén engedélyezzük a szolgáltató
# DHCP szerverének elérését.
$cmd 030 $skip udp from any to x.x.x.x 67 out via $pif keep-state

# Kifelé engedélyezzük a szabvány nem biztonságos WWW funkciót
$cmd 040 $skip tcp from any to any 80 out via $pif setup keep-state

# Kifelé engedélyezzük a biztonságos HTTPS funkciót a TLS SSL
# használatával.
$cmd 050 $skip tcp from any to any 443 out via $pif setup keep-state

# Kifelé engedélyezzük az e-mailek küldését és fogadását.
$cmd 060 $skip tcp from any to any 25 out via $pif setup keep-state
$cmd 061 $skip tcp from any to any 110 out via $pif setup keep-state

# Kifelé engedélyezzük a FreeBSD (make install és CVSUP) funkcióit.
# Ezzel a rendszeradminisztrátornak ,,ISTENI'' jogokat adunk.
```

```
$cmd 070 $skip tcp from me to any out via $pif setup keep-state uid root

# Kifelé engedélyezzük a pinget.
$cmd 080 $skip icmp from any to any out via $pif keep-state

# Kifelé engedélyezzük az idõ szolgáltatást.
$cmd 090 $skip tcp from any to any 37 out via $pif setup keep-state

# Kifelé engedélyezzük az nntp news szolgáltatást (tehát a
# hírcsoportokat).
$cmd 100 $skip tcp from any to any 119 out via $pif setup keep-state

# Kifelé engedélyezzük a biztonságos FTP, telnet és SCP
# funkciókat az SSH (secure shell) használatával.
$cmd 110 $skip tcp from any to any 22 out via $pif setup keep-state

# Kifelé engedélyezzük ki a whois kéréseket.
$cmd 120 $skip tcp from any to any 43 out via $pif setup keep-state

# Kifelé engedélyezzük az NTP idõszerver elérését.
$cmd 130 $skip udp from any to any 123 out via $pif keep-state

#####################################################################
# Az internet felõli interfész (bejövõ kapcsolatok)
# A saját hálózatunk felé vagy erre az átjáróra
# nyitott kapcsolatokat vizsgáljuk az internet felõl.
#####################################################################

# Tiltsuk a fenntartott címtartományok felé haladó összes beérkezõ
# forgalmat.
$cmd 300 deny all from 192.168.0.0/16  to any in via $pif  #RFC 1918: privát IP
$cmd 301 deny all from 172.16.0.0/12   to any in via $pif  #RFC 1918: privát IP
$cmd 302 deny all from 10.0.0.0/8      to any in via $pif  #RFC 1918: privát IP
$cmd 303 deny all from 127.0.0.0/8     to any in via $pif  #helyi
$cmd 304 deny all from 0.0.0.0/8       to any in via $pif  #helyi
$cmd 305 deny all from 169.254.0.0/16  to any in via $pif  #DHCP
$cmd 306 deny all from 192.0.2.0/24    to any in via $pif  #dokumentációs célokra
fenntartott
$cmd 307 deny all from 204.152.64.0/23 to any in via $pif  #Sun klaszter
$cmd 308 deny all from 224.0.0.0/3     to any in via $pif  #D és E osztályú multicast

# Az ident tiltása.
$cmd 315 deny tcp from any to any 113 in via $pif

# Blokkoljuk az összes Netbios szolgáltatást: 137=név, 138=datagram,
# 139=session. A Netbios az MS Windows megosztását implementálja.
# Blokkoljuk az MS Windows hosts2 névszerver kéréseit is a 81-es
# porton.
$cmd 320 deny tcp from any to any 137 in via $pif
$cmd 321 deny tcp from any to any 138 in via $pif
$cmd 322 deny tcp from any to any 139 in via $pif
```

```
$cmd 323 deny tcp from any to any 81  in via $pif

# Dobjuk el a késõn érkezõ csomagokat.
$cmd 330 deny all from any to any frag in via $pif

# Dobjuk el azokat az ACK csomagokat, amelyekre nincs
# dinamikus szabály.
$cmd 332 deny tcp from any to any established in via $pif

# Engedélyezzük a szolgáltató DHCP szerverétõl érkezõ forgalmat. Ennek
# a szabálynak tartalmaznia kell a DHCP szerver címét, mert csak tõle
# fogadunk el ilyen típusú csomagokat. Egyedül csak kábeles vagy DSL
# konfigurációk esetén használatos, a "felhasználói PPP" esetében
# törölhetjük. Ez ugyanaz az IP-cím, amelyet a kimenõ kapcsolatoknál
# megadtunk.
$cmd 360 allow udp from x.x.x.x to any 68 in via $pif keep-state

# Befelé engedélyezzük a szabvány WWW funkciót, mivel van
# webszerverünk.
$cmd 370 allow tcp from any to me 80 in via $pif setup limit src-addr 2

# Befelé engedélyezzük a biztonságos FTP, telnet és SCP
# használatát az internetrõl.
$cmd 380 allow tcp from any to me 22 in via $pif setup limit src-addr 2

# Befelé engedélyezzük a nem biztonságos telnet elérését az
# internetrõl. Azért nem tekintjük biztonságosnak, mert az
# azonosítókat és a jelszavakat az interneten titkosítatlanul
# közvetíti. Ha nincs telnet szolgáltatásunk, akkor törölhetjük is ezt
# a csoportot.
$cmd 390 allow tcp from any to me 23 in via $pif setup limit src-addr 2

# Dobjuk el és naplózzuk az összes internetrõl érkezõ hitelesítetlen kapcsolatot.
$cmd 400 deny log all from any to any in via $pif

# Dobjuk el és naplózzuk az összes internetre menõ hitelesítetlen kapcsolatot.
$cmd 450 deny log all from any to any out via $pif

# Ez lesz a kimenõ szabályokhoz tartozó "skipto" célja.
$cmd 800 divert natd ip from any to any out via $pif
$cmd 801 allow ip from any to any

# Minden mást alapértelmezés szerint tiltunk és naplózunk.
$cmd 999 deny log all from any to any
############## Az IPFW szabályai itt fejezõdnek be ######################
```

Chapter 31. Egyéb haladó hálózati témák

31.1. Áttekintés

Ebben a fejezetben számos komolyabb hálózati témát fogunk tárgyalni.

A fejezet elolvasása során megismerjük:

- az átjárók és az útválasztás alapjait;
- hogyan állítsunk be IEEE® 802.11 és Bluetooth® eszközöket;
- a FreeBSD segítségével hogyan tudunk két hálózatot összekötni hálózati hidakon keresztül;
- hogyan indítsuk hálózatról egy lemez nélküli gépet;
- hogyan állítsunk be hálózati címfordítást;
- hogyan kapcsoljunk össze két számítógépet PLIP használatával;
- hogyan állítsuk be az IPv6 használatát egy FreeBSD-s gépen
- hogyan állítsuk be az ATM használatát;
- hogyan engedélyezzük és használjuk a Közös címredundancia protokollt FreeBSD-ben.

A fejezet elolvasásához ajánlott:

- az /etc/rc könyvtárban található szkriptek mûködésének ismerete;
- az alapvetõ hálózati fogalmak ismerete;
- egy új FreeBSD rendszermag beállításának és telepítésének ismerete (A FreeBSD rendszermag testreszabása);
- a külsõ szoftverek telepítésének ismerete (Alkalmazások telepítése. csomagok és portok).

31.2. Átjárók és az útválasztás

Egy gép egy másikat úgy tud megtalálni a hálózaton, ha erre létezik egy olyan mechanizmus, amely leírja, hogyan tudunk eljutni az egyiktõl a másikig. Ezt hívjuk *útválasztásnak* (routing). Az "útvonal" (route) címek egy párjaként adható meg, egy "céllal" (destination) és egy "átjáróval" (gateway). Ez a páros mondja meg, hogy ha el akarjuk érni ezt a *célt*, akkor ezen az *átjárón* keresztül kell továbbhaladnunk. A céloknak három típusa lehet: egyéni gépek, alhálózatok és az "alapértelmezett". Az "alapértelmezett útvonalat" (default route) abban az esetben alkalmazzuk, ha semelyik más útvonal nem megfelelõ. Az alapértelmezett útvonalakról a késõbbiekben még beszélni fogunk. Három típusa van az átjáróknak: egyéni gépek, felületek (avagy "linkek") és a hardveres Ethernet címek (MAC-címek).

31.2.1. Példa

Az útválasztás különbözõ területeit a következõ `netstat` parancs alapján fogjuk bemutatni:

```
% netstat -r
```

```
Routing tables

Destination      Gateway            Flags     Refs     Use     Netif Expire

default          outside-gw         UGSc        37      418      ppp0
localhost        localhost          UH           0      181       lo0
test0            0:e0:b5:36:cf:4f   UHLW         5    63288       ed0     77
10.20.30.255     link#1             UHLW         1     2421
example.com      link#1             UC           0        0
host1            0:e0:a8:37:8:1e    UHLW         3     4601       lo0
host2            0:e0:a8:37:8:1e    UHLW         0        5       lo0 =>
host2.example.com link#1             UC          0        0
224              link#1             UC           0        0
```

Az elsõ két sorban az alapértelmezett útvonalat (melyrõl részleteiben majd a következõ szakaszban fogunk szólni) és a `localhost` útvonalát láthatjuk.

A `localhost` címhez az útválasztási táblázatban a lo0 eszköz tartozik (a `Netif` oszlopban), amelyet loopback eszköznek is neveznek. Ez arra utasítja a rendszert, hogy az ide küldött csomagokat ne a helyi hálózaton küldje keresztül, hanem csak ezen a belsõ felületen, mivel úgyis oda jutnának vissza, ahonnan indultak.

A táblázatban a következõ sor egy `0:e0` kezdetû címet tartalmaz. Ez egy hardveres Ethernet cím, más néven MAC-cím. A FreeBSD magától képes beazonosítani tetszõleges gépet (ebben a példában a `test0` gépet) a helyi Ethernetes hálózaton és felvenni hozzá egy útvonalat, közvetlenül az ed0 Ethernetes csatolófelületen keresztül. Ehhez a típusú útvonalhoz tartozik még egy lejárati idõ is (a `Expire` oszlop), amely akkor kap szerepet, ha ennyi idõ elteltével nem kapunk semmilyen hírt a géprõl. Amikor ilyen történik, az géphez eddig nyilvántartott útvonal automatikusan törlõdik. Ezek a gépek a RIP (útvonal-információs protokoll, Routing Information Protocol) nevû mechanizmuson keresztül azonosítódnak, mely a legrövidebb út kiszámítása alapján határozza meg a helyi gépekhez vezetõ útvonalat.

A FreeBSD a helyi alhálózat (`10.20.30.255` és `example.com`, az alhálózathoz tartozó név) esetében is felvesz útvonalakat. A `link#1` megnevezés a gépben található elsõ Ethernet-kártyát jelöli. Megfigyelhetjük, hogy rajta kívül nincs is több felülete.

Mindegyik csoport (a helyi hálózati gépek és a helyi alhálózatokatok) útvonalait a routed nevû démon tartja automatikusan karban. Ha ez nem fut, akkor csak a statikusan definiált (vagyis az elõre megadott) útvonalak fognak létezni.

A `host1` sor a saját gépünkre vonatkozik, amelyet az Ethernet címe szerint ismerünk. Mivel mi vagyunk küldõ gép, a FreeBSD tudni fogja, hogy ilyenkor az Ethernetes felület helyett a loopback eszközt (lo0) kell használnia.

A két `host2` sor arra mutat példát, amikor az ifconfig(8) paranccsal álneveket hozunk létre (ennek konkrét okait lásd az Ethernetrõl szóló részben). A lo0 felület neve után szereplõ ⇒ szimbólum azt jelzi, hogy ez nem csak egy loopback felület (mivel a címe szintén a helyi gépre mutat), hanem a felület egy másik neve. Ilyen útvonalak csak az álneveket ismerõ gépeknél jelennek meg. A helyi hálózaton minden más gépnél egyszerûen csak a `link#1` jelenik meg az ilyen útvonalak esetében.

Az utolsó sor (a `224` céllal rendelkezõ alhálózat) a multicastre (többesküldésre) szolgál, amellyel majd egy másik szakaszban foglalkozunk.

Végezetül az útvonalakhoz tartozó különféle tulajdonságok a `Flags` oszlopban láthatóak. Az alábbi rövid táblázatban összefoglaltunk közülük néhányat:

U	Up: az útvonal aktív
H	Host: az útvonal egyetlen gépre mutat
G	Gateway: az adott cél felé ezen a gépen keresztül küldjünk, amely majd kitalálja, hogy merre küldje tovább
S	Static: ez az útvonal statikus, nem a rendszer hozta létre automatikusan
C	Clone: ebbõl az útvonalból származtatunk új útvonalat azokhoz a gépekhez, amelyekhez csatlakozunk. Ilyen útvonalakat általában a helyi hálózatokban találhatunk
W	WasCloned: azt jelzi, hogy ezt az útvonalat egy helyi hálózatra mutató (klón, avagy Clone típusú) útvonal alapján hoztuk létre automatikusan
L	Link: az útvonal Ethernetes hardverhez kapcsolódik

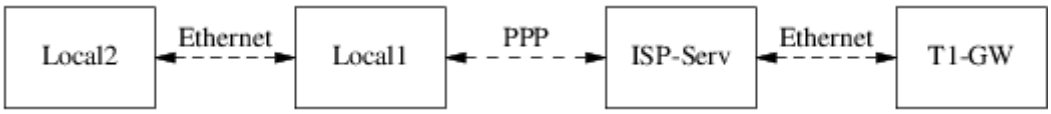
31.2.2. Alapértelmezett útvonalak

Amikor a helyi rendszernek fel kell vennie a kapcsolatot egy távoli géppel, ellenõrzi az útválasztási táblázatban, hogy létezik-e már hozzá valamilyen útvonal. Ha a távoli gép egy olyan alhálózatba esik, amelyet már el tudunk érni (klónozott útvonalak), akkor a rendszer megnézi, hogy a hozzá tartozó felületen képes-e kapcsolatot létesíteni.

Ha minden ismert útvonal csõdöt mond, akkor a rendszerünknek marad még egy utolsó esélye: az "alapértelmezett" útvonal használata. Ez az útvonal egy speciális átjáró útvonal (ebbõl általában csak egyetlen egy létezik a rendszerben) és tulajdonságai között mindig szerepel a `c`. A helyi hálózat gépei közül ez az átjáró az legyen, amelyik közvetlenül kapcsolódik a külsõ világhoz (PPP összeköttetéssel, DSL, kábelmodem, T1 vagy bármilyen más hálózati felületen keresztül).

Amikor pedig magát a külsõ világ felé átjáróként szolgáló gépet állítjuk be, az alapértelmezett útvonal az internet-szolgáltatónk által megadott gép címe lesz.

Vegyünk egy példát az alapértelmezett útvonalakra. Egy tipikus konfiguráció:



A `Helyi1` és `Helyi2` gépek a hálózatunk tagjai. A `Helyi1` az internet-szolgáltatót éri el egy betárcsázós

PPP kapcsolaton keresztül. A PPP szerver a külsõ felületén keresztül a helyi hálózaton pedig egy másik átjáróhoz csatlakozik.

Az egyes gépek alapértelmezett útvonalai így alakulnak:

Gép	Alapértelmezett átjáró	Felület
Helyi2	Helyi1	Ethernet
Helyi1	T1-ÁJ	PPP

Gyakran felmerül a kérdés, hogy "Miért (és hogy-hogy) a `T1-ÁJ` a `Helyi1` gép számára az alapértelmezett átjáró és nem a szolgáltató azon szervere, amelyhez csatlakozott?"

Ne felejtsük el, hogy a PPP felület a szolgáltató helyi hálózatában a mi részünkre kap címet, és a itt az összes többi géphez tartozó útvonal automatikusan létrejön. Emiatt már eleve el tudjuk érni a `T1-ÁJ` gépet, ezért amikor a szolgáltatón keresztül küldünk, nincs szükségünk egy további lépcsõre.

Általában a `X.X.X.1` címet szokták a helyi hálózat átjárójának kiosztani. Ezért (az elõbbi példát újrahasznosítva) ha a helyi hálózatunkon a C osztályú `10.20.30` címtartományt használjuk, és a szolgáltatónkhoz a `10.9.9` címtartomány tartozik, akkor az alapértelmezett útvonalak a következõk lesznek:

Gép	Alapértelmezett útvonal
Helyi2 (10.20.30.2)	Helyi1 (10.20.30.1)
Helyi1 (10.20.30.1, 10.9.9.30)	T1-ÁJ (10.9.9.1)

Az /etc/rc.conf állományon keresztül könnyen meg tudjuk adni az alapértelmezett útvonalat. A példánkban a `Helyi2` gép /etc/rc.conf állományába kell felvennünk a következõ sort:

```
defaultrouter="10.20.30.1"
```

A route(8) parancs használatával viszont akár közvetlenül is megtehetjük mindezt:

```
# route add default 10.20.30.1
```

A route(8) man oldalon olvashatunk arról bõvebben, hogy a hálózati útválasztási táblázatokat kézzel hogyan tudjuk módosítani.

31.2.3. Kettõs hálózatú gépek

Egy másik típusú konfigurációról is szót kell ejtenünk, ahol a gép egyszerre két hálózatnak is tagja. Gyakorlatilag az átjáróként üzemelõ számítógépek (mint például az, amelyik a fenti példában PPP kapcsolattal csatlakozott) ilyen kettõs hálózatú gépnek tekinthetõek. Ez a kifejezés azonban igazából csak azokra az esetekre illik, ahol a gép egyszerre két helyi hálózatban is megjelenik.

Az egyik esetben a gépben két Ethernet kártya található, melyek mindegyike birtokol egy-egy hálózati címet az egyes alhálózatokon. De elõfordulhat az is, hogy a gépünkben csupán egyetlen

Ethernet kártya van és az ifconfig(8) segítségével álneveket hoztunk létre hozzá. Az elõbbi általában két fizikailag elkülönölõ Ethernet alapú hálózat esetében történik, míg az utóbbinál csak egyetlen fizikai hálózati szegmensrõl van szó, amely viszont logikailag két külön alhálózatot tartalmaz.

Akármelyiket is vesszük, az útválasztási táblázatok úgy jönnek létre, hogy bennük a gép a másik alhálózat felé átjáróként (bejövõ útvonalként) lesz nyilvántartva. Ebben a konfigurációban a gép a két alhálózat között útválasztóként fog tevékenykedni, és gyakran valamelyik vagy éppen mind a két irányba be kell állítanunk valamilyen csomagszûrést vagy tûzfalazást.

Ha azt szeretnénk, hogy ez a gép a két felület között továbbítson csomagokat, akkor a FreeBSD-ben külön engedélyezni kell ezt a lehetõséget. A következõ szakaszban ennek részleteit tárjuk fel.

31.2.4. Az útválasztók beállítása

A hálózati útválasztó nem csinál mást, csak továbbküldi az egyik felületén beérkezõ csomagokat egy másik felületére. Az internetes szabványok és a sokéves mérnöki tapasztalat azonban nem engedik, hogy a FreeBSD Projekt alapértelmezés szerint is elérhetõvé tegye ezt a FreeBSD rendszerekben. Ezt a lehetõséget az alábbi változó `YES` értékûre állításával lehet engedélyezni az rc.conf(5) állományban:

```
gateway_enable="YES"          # Ez legyen YES, ha átjáróként akarunk üzemelni
```

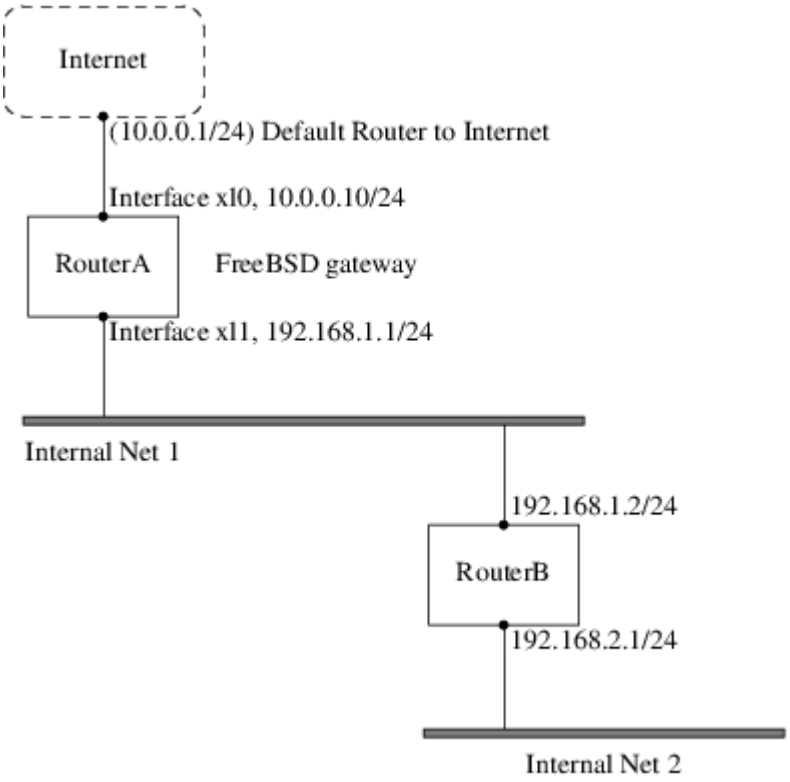
Ezzel lényegében a `net.inet.ip.forwarding` sysctl(8) változó értékét állítjuk `1`-re. Ha valamiért egy idõre szüneteltetni akarjuk a csomagok továbbküldését, akkor állítsuk a változó értékét `0`-ra.

Az új útválasztónak nem árt arról sem tudnia, hogy merre továbbítsa a forgalmat. Ha elég egyszerû a hálózatunk, akkor akár statikus útvonalakat is használhatunk. A FreeBSD alapból tartalmazza a BSD-k esetén szabványos routed(8) útválasztó démont, amely a RIP (v1 és v2) valamint az IRDP megoldásokat ismeri. A BGP v4, OSPF v2 és a többi fejlettebb útválasztási protokoll a net/zebra csomagban érhetõ el. Az ettõl bonyolultabb hálózati útválasztási feladatokhoz olyan kereskedelmi termékek is elérhetõek, mint például a GateD®.

31.2.5. Statikus útvonalak beállítása

Manuális konfiguráció

Tegyük fel, hogy hálózatunk a következõ:



Ebben a forgatókönyvben az `A-utvalaszto` a mi FreeBSD-s gépünk, amely az internet felé vezetõ útválasztó szerepét játssza. Számára az alapértelmezett útvonal a `10.0.0.1`, amelyen keresztül a külsõ világot tudja elérni. Feltételezzük, hogy a `B-utvalaszto` nevû gépet már eleve jól állítottuk be, ezért tudja merre kell mennie. (A kép alapján egyszerû: csak vegyünk fel egy alapértelmezett útvonalat a `B-utvalaszto` géphez, ahol így a `192.168.1.1` lesz az átjáró.)

Ha megnézzük most az `A-utvalaszto` útválasztási táblázatát, akkor nagyjából a következõket fogjuk látni:

```
% netstat -nr
Routing tables

Internet:
Destination        Gateway            Flags    Refs      Use  Netif  Expire
default            10.0.0.1           UGS         0    49378    xl0
127.0.0.1          127.0.0.1          UH          0        6    lo0
10.0.0/24          link#1             UC          0        0    xl0
192.168.1/24       link#2             UC          0        0    xl1
```

Az `A-utvalaszto` útválasztási táblázata alapján jelen helyzetben nem lehet elérni a 2. belsõ hálózatot. Nincs ugyanis olyan útvonal, amely a `192.168.2.0/24` alhálózat felé vezetne. Ezt például úgy tudjuk megoldani, ha manuálisan felvesszük ezt az útvonalat. Az alábbi paranccsal hozzáadjuk a 2. belsõ hálózat elérését az `A-utvalaszto` útválasztási táblázatához, ahol a `192.168.1.2` lesz a következõ ugrási pont (next hop):

```
# route add -net 192.168.2.0/24 192.168.1.2
```

Most már az A-utvalaszto bármelyik gépet képes elérni a 192.168.2.0/24 hálózaton.

Rögzített konfiguráció

A fenti példa tökéletesen szemlélti a statikus útvonalak felvételét egy mûködõ rendszeren. Azonban ezzel az a gond, hogy az így megadott útválasztási információ nem marad meg a gép újraindítása után. Ezért az elõbbihez hasonló statikus útvonalakat inkább az /etc/rc.conf állományban rögzítsük:

```
# A 2. belsõ hálózat elérését felvesszük statikus útvonalként
static_routes="belsohalo2"
route_belsohalo2="-net 192.168.2.0/24 192.168.1.2"
```

A static_routes konfigurációs változó karakterláncok szóközzel tagolt felsorolását tartalmazza. Mindegyik karakterlánc egy útvonal neve. Az iménti példában csak egyetlen ilyen név szerepelt a static_routes értékében, amely a *belsohalo2* volt. Utána beírtunk még egy konfigurációs változót is, amelynek a neve route_belsohalo2. Ide helyeztük a route(8) parancsnak átadandó beállítás összes paraméterét. Ez pontosan olyan, mintha a következõ parancsot adtuk volna ki:

```
# route add -net 192.168.2.0/24 192.168.1.2
```

Ezért kellett a "-net 192.168.2.0/24 192.168.1.2".

Ahogy már korábban is említettük, a static_routes értékében több karakterláncot is megadhatunk, aminek segítségével egyszerre több statikus útvonalat is létrehozhatunk. A következõ sorok arra mutatnak példát, hogy a 192.168.0.0/24 és 192.168.1.0/24 hálózatok számára miként állítsunk be statikus útvonalakat a képzeletbeli útválasztónkon:

```
static_routes="net1 net2"
route_net1="-net 192.168.0.0/24 192.168.0.1"
route_net2="-net 192.168.1.0/24 192.168.1.1"
```

31.2.6. Az útvonalak terjedése

Azt már tudjuk, hogyan adjuk meg a külvilág felé vezetõ útvonalakat, azonban arról még nem beszéltünk, hogy kívülrõl miként találnak meg bennünket.

Annyit már megismertünk, hogy az útválasztási táblázatokban megadhatjuk a hálózaton azt a gépet, amelyen keresztül az adott címtartomány (a példában egy C osztályú alhálózat) felé küldhetünk, amely pedig továbbküldi a hozzá érkezõ csomagokat.

Amikor a csatlakozunk az internet-szolgáltatónkhoz, a nála levõ útválasztási táblázatok úgy állítódnak be, hogy az alhálózatunk felé igyekvõ adatok a korábban létrejött PPP összeköttetésen keresztül jutnak el hozzánk. A világ többi részén levõ rendszerek viszont honnan fogják tudni, hogy a mi internet-szolgáltatónknak küldjenek?

Van egy rendszer (ez leginkább a névszerverek elosztott információs adatbázisához hasonlít), ami

nyilvántartja a pillanatnyilag kiosztott címtartományokat és megadja a csatlakozási pontjukat az internet gerinchálózatán. Ez a "gerinc" tulajdonképpen olyan fõvonalakból áll, amelyen keresztül a világban az országok között mozog az internet forgalma. A gerinchálózat mindegyik gépe tárolja a központi útválasztási táblázatok egy másolatát, ami a forgalmat egy adott hálózatról a megadott gerincbeli hordozóra irányítja át, végig az internet-szolgáltatók láncán egészen addig, amíg az el nem éri a hálózatunkat.

A szolgáltatónk feladata, hogy a gépünk felé leágazásként (és így a felénk vezetõ útként) beregisztálja magát a gerinchálózat gépein. Ezt nevezik az útvonal terjedésének.

31.2.7. Hibaelhárítás

Néha gondok lehetnek az útvonal terjedésével, és egyes gépek nem képesek elérni minket. A traceroute(8) parancs mind közül talán az egyik leghasznosabb ilyen helyzetekben, mivel ezzel fel tudjuk deríteni, hogy az útválasztás hol akad meg. Ugyanilyen jól hasznosítható azokban az esetekben, amikor látszólag nem tudunk elérni egy távoli gépet (tehát a ping(8) csõdöt mond).

A traceroute(8) parancsnak annak a távoli gépnek a nevét kell megadnunk, amelyhez csatlakozni akarunk. Futása közben megjeleníti azokat az átjárókat, amelyeken keresztül csatlakozni próbál, akár sikerült elérni a célgépet, akár a kapcsolat hiánya miatt kudarcot vall.

A parancs használatáról és mûködésérõl részletesebb információkat a traceroute(8) man oldalán találunk.

31.2.8. Útválasztás multicast esetén

A FreeBSD alapból támogatja mind a multicastet használó alkalmazásokat, mind pedig a multicasthez tartozó útválasztást. Multicast esetében semmilyen speciális beállítás nem szükségeltetik, az ilyen alkalmazások egyből el tudják érni ezt a lehetõséget. A multicast kérések útválasztásához azonban be kell építenünk némi támogatást a rendszermagba:

```
options MROUTING
```

Emellett még el kell indítanunk az mrouted(8) démont is, amelyhez az /etc/mrouted.conf állományban még be kell állítanunk tunneleket és a DVMRP használatát. A multicasthez tartozó további beállításokat az mrouted(8) man oldalán találhatjuk.

A FreeBSD 7.0 megjelenésével a mrouted(8) démont kivették az alaprendszerbõl. Azt a DVMRP többesküldési protokollt valósítja meg, amelyet a legtöbb alkalmazásban mostanság már a pim(4) segítségével oldanak meg. Ennek megfelelõen a hozzá tartozó multicast protokollt valósítja meg, amelyet a legtöbb alkalmazásban mostanság már a pim(4) segítségével oldanak meg. Ennek megfelelõen a hozzá tartozó map-mbone(8) és mrinfo(8) segédprogramok is eltávolításra kerültek. Ezek a programok attól a kiadástól kezdõdõen a Portgyûjtemény részeként érhetõek el a net/mrouted portban.

31.3. Vezeték nélküli hálózatok

31.3.1. A vezeték nélküli hálózatok alapjai

A legtöbb vezeték nélküli hálózat az IEEE® 802.11 szabványon nyugszik. Az alapvetõ vezeték nélküli hálózatokban több olyan állomást találhatunk, amelyek egymással rádiójelek szórásával kommunikálnak a 2,4 GHz vagy 5 GHz frekvenciatartományban (noha ez a helyi viszonyoknak megfelelõen változhat, és a 2,3 GHz, illetve a 4,9 GHz tartományokban is lehetséges a kommunikáció).

A 802.11 szabványú hálózatok kétféleképpen szervezõdnek. Elõször is *infrastrukturálisan*, (infrastructural mode) ahol az egyik állomást kinevezzük a központnak és a többi pedig ehhez fog tartozni. Az ilyen hálózatokat BSS-nek nevezzük és az imént említett központ neve hozzáférési pont (Access Point, AP) lesz. A BSS-ben az összes kommunikáció a hozzáférési pontokon keresztül halad még abban az esetben is, amikor az egyik állomás egy másik vezeték nélküli állomással akarja felvenni a kapcsolatot. Az ilyen jellegû hálózatok másik típusú szervezõdési módjában nincsenek kijelölt központok és a kommunikáció az állomások között közvetlenül zajlik. A hálózat ezen formáját IBBS-nek nevezzük, vagy ismeretebb nevén *ad-hoc hálózatnak* (ad-hoc network).

A 802.11 alapú hálózatok elsõként a 2,4 GHz-es sávot hódították meg, és az IEEE® 802.11 valamint 802.11b szabványokban rögzített protokollokat használták. Ezekben a specifikációkban megtalálhatjuk a mûködési frekvenciát, a közeghozzáférési réteg jellemzõinek leírását, beleértve a keretezést és az átviteli sebességeket (a kommunikáció ugyanis eltérõ sebességekkel is történhet). A késõbb kiadott 802.11a szabvány azt specifikálja, hogy az 5 GHz-es tartományban miként mûködjenek, ahol többek közt megtalálhatjuk a különféle jelkezelési mechanizmusokat és a nagyobb átviteli sebességek használatát. Ezt még a 802.11g szabvány követte, ami a 802.11b hálózatokkal kompatibilis módon lehetõvé tette a 802.11a jelkezelésének és átviteli módszereinek használatát a 2,4 GHz-es sávban.

A 802.11 alapú hálózatok mindenféle átviteli technikáitól eltekintve többféle biztonsági megoldással találkozhatunk. Az korai 802.11 dokumentumok egy nagyon egyszerû biztonsági protokollt, a WEP-et említenek. Ez a protokoll a hálózaton mozgó adatokat egy rögzített és ismert osztott kulccsal kódolja le az RC4 titkosítással. A kommunikációhoz az összes állomásnak elõre meg kell egyeznie ebben a kulcsban. Errõl a sémáról idõközben kiderült, hogy könnyen feltörhetõ és manapság már csak nagyon ritkán alkalmazzák, kivéve talán csak a kóbor felhasználók elijesztésére. A jelenleg érvényes biztonsági elõírásokat az IEEE® 802.11i specifikáció adja meg, amely új kriptográfiai titkosításokat definiál valamint egy további protokollt az állomások azonosítására és a kulcsok cseréjére. Emellett a titkosításhoz használt kulcsok idõszakosan frissülnek és külön eszközök állnak rendelkezésre a betörési kísérletek észlelésére (és azok elhárítására). A vezeték nélküli hálózatok esetében másik elterjedt titkosítási protokoll a WPA. Ez igazából 802.11i elõdjének tekinthetõ, amelyet egy ipari csoport definiált, amíg a 802.11i minõsítés alatt állt. A WPA ennek megfelelõen teljesíti a 802.11i szabvány elvárásainak egy részét és kifejezetten a régi hardverek számára készült. A WPA mûködéséhez egyedül a TKIP titkosításra van szükségünk, amely az eredeti WEP titkosításból származik. A 802.11i engedi a TKIP használatát, de az adatok kódolására egy erõsebb titkosítás, az AES-CCM ismeretét is igényli. (Az AES a WPA esetében nem kell, mivel a régi eszközök esetében túlságosan költségesnek ítélték meg a használatát.)

A fenti szabványokon kívül a 802.11e a másik fontos szabvány, amire tekintettel kell lennünk. Ez írja le a 802.11 hálózatokon a multimédiás alkalmazások közvetítéséhez, mint például a videók valós idejû lejátszásához vagy a VoIP (voice over IP) megvalósításához tartozó protokollokat. A 802.11i szabványhoz hasonlóan a 802.11e is magában foglal egy elõzetes specifikációt, amelyet WME (késõbb pedig már WMM)-nek neveznek. Ezt szintén egy ipari csoport definiálta a 802.11e részeként, amivel a 802.11e végsõ elfogadásáig tudják a multimédiás igényeket kiszolgálni. Amit a 802.11e és WME/WMM megoldásaival kapcsolatban érdemes tudnunk: a QoS (Quality of Service) protokoll és más egyéb fejlett közeghozzáférési protokollok segítségével a vezeték nélküli hálózatokban lehetõvé teszik a forgalom prioritás szerinti ütemezését. Ezen protokollok megfelelõ implementációjának segítségével tehát a fontosabb adatok nagy sebességû küldését és áramoltatását vagyunk képesek elérni.

A FreeBSD a 6.0 verzió óta ismeri a 802.11a, 802.11b és 802.11g szabványokon alapján mûködõ hálózatokat. A WPA és 802.11i biztonsági protokollok (a 11a, 11b és 11g szabványok bármelyike esetén) hasonlóképpen támogatottak, valamint a WME/WMM protokollok mûködéséhez szükséges QoS csak bizonyos vezeték nélküli eszközök esetében.

31.3.2. Kezdeti beállítások

A rendszermag beállítása

A vezeték nélküli hálózatok használatához egy vezeték nélküli hálózati kártyára lesz szükségünk, valamint a rendszermagban is be kell állítani ehhez a megfelelõ támogatást. Ez utóbbit több különbözõ modulra szedték szét, és ezek közül csak azokat kell beállítani, amelyeket tényleg használni is fogunk.

Elõször is tehát kell egy vezeték nélküli eszköz. Az elterjedtebb típusaik általában az Atheos által gyártott alkatrészeket tartalmazzák. Az ilyen fajtájú eszközöket az ath(4) meghajtó kezeli, melyet úgy tudunk a rendszer indításakor betölteni, ha a /boot/loader.conf állományba felvesszük a következõ sort:

```
if_ath_load="YES"
```

Az Atheos meghajtója három különálló részre oszlik: maga a meghajtó (ath(4)), a hardveres réteg, ami a chipfüggõ funkciókat kezeli (ath_hal(4)) és a keretek küldésével kapcsolatban az átviteli sebesség megválasztását lehetõvé tevõ algoritmus (ez itt most az ath_rate_sample). Amikor ezt a támogatást modulként töltjük be, ezek a függõségek automatikusan feloldódnak. Ha az Atheos eszközök helyett valamelyik másikhoz tartozó modult szeretnénk használni, akkor például az Intersil Prism esetében a wi(4) meghajtót kell megadnunk:

```
if_wi_load="YES"
```

A leírás további részeiben az ath(4) eszközt fogjuk használni, minden más esetben ennek a nevét kell csak lecserélünk a példákban. A rendszerben elérhetõ vezeték nélküli meghajtók és az általuk támogatott kártyák listája a FreeBSD Hardverjegyzetekben található. Ezek a jegyzetek a különbözõ architektúrákra és

kiadásokhoz a FreeBSD holnapjáról, a Kiadási jegyzetek oldalról érhetõek el. Ha a vezeték nélküli eszközünkhöz nem létezik natív FreeBSD-s meghajtó, akkor az NDIS meghajtó segítségével akár közvetlenül a Windows®-os meghajtóját is használhatjuk.

FreeBSD 7.*X* esetén az eszközmeghajtó beállításával együtt a 802.11 hálózatok támogatását is be kell töltenünk a rendszermagba. Ez az ath(4) meghajtó esetében a legalább a wlan(4), `wlan_scan_ap` és `wlan_scan_sta` modulok betöltését jelenti. A wlan(4) modul a vezetéknélküli eszköz meghajtóprogramjával együtt töltõdik be, míg a többi modult a /boot/loader.conf állomány használatával kell a rendszerindítás során betöltenünk:

```
wlan_scan_ap_load="YES"
wlan_scan_sta_load="YES"
```

A FreeBSD 8.0 kiadástól kezdõdõen ezek a modulok részei a wlan(4) meghajtónak, amely a hálózati kártya meghajtójával együtt mindig automatikusan betöltõdik.

Emellett még azokra a modulokra is szükségünk van, amelyek a használni kívánt biztonsági protokollokhoz nyújtanak kriptográfiai támogatást. Ezek hivatalosan a wlan(4) modul kérésére automatikusan betöltõdnek, azonban itt most manuálisan állítjuk be. Erre a célra a következõ modulokat találjuk: wlan_wep(4), wlan_ccmp(4) és wlan_tkip(4). A wlan_ccmp(4) és wlan_tkip(4) meghajtók csak akkor fognak kelleni, ha a WPA és/vagy a 802.11i biztonsági protokollokat használjuk. Amennyiben a hálózatunkon nincs titkosítás, akkor még a wlan_wep(4) támogatás sem kell. Ezeket a modulok úgy lehet betölteni a rendszerindításnál, ha felvesszük a következõ sorokat a /boot/loader.conf állományba:

```
wlan_wep_load="YES"
wlan_ccmp_load="YES"
wlan_tkip_load="YES"
```

Miután ezt megcsináltuk, egyszerûen csak indítsuk újra a gépünket. Ha még nem akarjuk újraindítani a gépet, akkor a kldload(8) parancs segítségével akár kézzel is betölthetjük az elõbb felsorolt modulokat.

Ha nem akarunk modulokat használni, a mûködéshez szükséges meghajtókat a rendszermagba is be tudjuk építeni a következõ sorok megadásával a rendszermag beállításait tartalmazó állományban:

```
device wlan                # a 802.11 támogatása
device wlan_wep            # 802.11 WEP  támogatás
device wlan_ccmp           # 802.11 CCMP támogatás
device wlan_tkip           # 802.11 TKIP támogatás
device wlan_amrr           # AMRR forgalomvezérlési algoritmus
device ath                 # Atheros IEEE 802.11 vezeték nélküli
hálózati meghajtó
device ath_hal             # az Atheros meghajtó hardveres rétege
```

```
options AH_SUPPORT_AR5416  # az AR5416 tx/rx leírók engedélyezése
device ath_rate_sample     # SampleRate forgalomvezérlési algoritmus
```

Hozzátesszük, hogy az alábbi sorok hozzáadása a FreeBSD 7.*X* változatában kötelezõ, más verzióknál viszont nem:

```
device wlan_scan_ap        # a 802.11 AP módú keresés
device wlan_scan_sta       # a 802.11 STA módú keresés
```

Az elõbbiek megadásával fordítsuk újra és telepítsük a rendszermagot, majd indítsuk újra a számítógépünket.

Miután a rendszerünk újra elindult, a rendszer indítás során generált üzenetei között találnunk kell valamennyi információt a felismert vezeték nélküli eszközökrõl. Például:

```
ath0: <Atheros 5212> mem 0x88000000-0x8800ffff irq 11 at device 0.0 on cardbus1
ath0: [ITHREAD]
ath0: AR2413 mac 7.9 RF2413 phy 4.5
```

31.3.3. Az infrastrukturális mûködési mód

Általában az infrastrukturális avagy a BBS mód használata a gyakori. Ebben a mûködési módban adott számú vezeték nélküli hozzáférési pont csatlakozik a hagyományos hálózatra. Mindegyik vezeték nélküli hálózatnak saját neve van, amit a hálózat SSID-jének hívunk. A vezeték nélküli kliensek ezekhez a vezeték nélküli hozzáférési pontokhoz kapcsolódnak.

A FreeBSD-s kliensek használata

Hogyan keressünk hozzáférési pontokat

A hálózatok kereséséhez az `ifconfig` paranccsal tudunk nekifogni. Egy ilyen kérés kiszolgálása eltarthat néhány pillanatig, mivel ekkor a rendszernek végig kell bóklásznia az összes elérhetõ frekvenciát és azokon hozzáférési pontok után kutatni. Egyedül a rendszeradminisztrátor kezdeményezheti ezeket a kereséseket:

```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 up scan
SSID/MESH ID    BSSID              CHAN RATE  S:N     INT CAPS
dlinkap         00:13:46:49:41:76   11   54M -90:96   100 EPS  WPA WME
freebsdap       00:11:95:c3:0d:ac    1   54M -83:96   100 EPS  WPA
```

Csak `up` jelzésû felületen tudunk hálózatokat keresni. További keresésekre már nincs szükség a felület `up` állapotban tartásához.

FreeBSD 7.*X* esetén a wlan0 eszköz helyett közvetlenül az adott eszköz nevét kell

megadnunk, például ath0. Az iménti sorokat ennek megfelelõen tehát ebben az esetben így kell értelmezni:

```
# ifconfig ath0 up scan
```

A leírás további részében a FreeBSD 7.*X* felhasználóknak ezen séma alapján kell használniuk a parancsokat és a konfigurációs beállításokat.

A keresés során keletkezõ listában láthatjuk megtalált BBS vagy IBBS fajtájú hálózatokat. A hálózatok neve és SSID-ja mellett még megjelenik egy BSSID oszlop is, ahol a hozzáférési pontok MAC-címe szerepel. A CAPS oszlop az egyes állomások tulajdonságait adja meg:

E
: Extended Service Set (ESS): az állomás egy infrastrukturális vagyis BBS hálózat része.

I
: IBSS/ad-hoc hálózat: az állomás egy ad-hoc hálózat része.

P
: Privacy: a BBS-en belül minden keretet titkosítani kell. Tehát a BSS arra kötelezi az állomást, hogy WEP, TKIP vagy AES-CCMP titkosítás használatával kódolja a hálózat tagjai között közlekedõ kereteket.

S
: Short Preamble: a hálózatban rövid bevezetõjeleket használnak (a 802.11b High Rate/DSSS PHY elõírásai szerint), ahol a szokványos 128 bites szinkronizációs mezõ hossza csak 56 bit.

s
: Short Slot Time: a 802.11g hálózat rövid slotidõt használ, mivel nem találhatóak benne régi (802.11b szabványú) állomások.

A jelenleg ismert hálózatok listáját így tudjuk lekérdezni:

```
# ifconfig wlan0 list scan
```

Ezt az információt maga az adapter automatikusan, vagy a felhasználó tudja frissíteni a scan kérés kiadásával. Az elavult adatok maguktól törlõdnek a gyorsítótárból, így idõvel a lista zsugorodni fog, hacsak nem keresünk folyamatosan hálózatokat.

Alapvetõ beállítások

Ebben a szakaszban arra mutatunk példákat, hogy miként tudunk FreeBSD alatt titkosítás nélkül használni egy vezeték nélküli hálózati kártyát. Miután elsajátítottuk az itt szereplõ ismereteket, határozottan javasoljuk, hogy a vezeték nélküli hálózatunkat WPA használatával állítsuk be.

A vezeték nélküli hálózatok beállítása három elemi lépésbõl épül fel: a hozzáférési pont kiválasztása, az állomásunk hitelesítése és az IP-cím beállítása. A következõkben ezeket a lépéseket

vitatjuk meg.

A hozzáférési pont kiválasztása

A legtöbb esetben hagyjuk, hogy a rendszer válassza ki magának a különbözõ heurisztikák alapján a leginkább megfelelõ hozzáférési pontot. Ez az alapértelmezett tevékenység, amikor aktiváljuk a felületet vagy valamilyen más módon, például az/etc/rc.conf állományból hivatkozunk rá:

```
wlans_ath0="wlan0"
ifconfig_wlan0="DHCP"
```

A korábban említettek szerint a FreeBSD 7.*X* felhasználóknak csak a kártyát kell beállítani:

```
ifconfig_ath0="DHCP"
```

Ha viszont több hozzáférési pont közül mi magunk akarunk kiválasztani egyet, akkor ezt az SSID megadásával tehetjük meg:

```
wlans_ath0="wlan0"
ifconfig_wlan0="ssid saját_ssid DHCP"
```

Amikor olyan környezetben vagyunk, ahol több hozzáférési pontnak is megegyezik az SSID-ja (gyakran így próbálják egyszerûsíteni azt, hogy automatikusan váltani lehessen köztük), akkor szükségünk lehet ezt egy adott eszközhöz hozzárendelni. Ebben az esetben a hozzáférési pont BSSID-ját is definiálni kell (és az SSID-t akár el is hagyhatjuk):

```
wlans_ath0="wlan0"
ifconfig_wlan0="ssid saját_ssid bssid xx:xx:xx:xx:xx:xx DHCP"
```

Más módokon is képesek vagyunk szabályozni a hozzáférési pontok megválasztását, például a rendszerünk által vizsgált frekvenciasávok megadásával. Ez olyankor tud hasznos lenni, ha többsávos vezeték nélküli kártyánk van, és az összes tartomány végigpásztázása túlságosan sok idõt venne el. Ezt a mûvelet a `mode` paraméter megadásával lehet egy konkrét sávra leszûkíteni, például a

```
wlans_ath0="wlan0"
ifconfig_wlan0="mode 11g ssid saját_ssid DHCP"
```

beállítás hatására a kártya 802.11g módban fog üzemelni, ami kizárólag csak 2,4 GHz-es frekvenciákon használható, így az 5 GHz-es csatornákat egyszerûen figyelmen kívül hagyjuk. Ugyanezt a `channel` paraméterrel is meg tudjuk oldani, mivel így a mûködést egy adott frekvenciára korlátozzuk, valamint a `chanlist` paraméterrel, ahol a pásztázandó csatornákat sorolhatjuk fel.

Ezekrõl a paraméterekrõl részletesebb leírást az ifconfig(8) man oldalon találhatunk.

Hitelesítés

Miután sikeresen kiválasztottuk a számunkra megfelelõ hozzáférési pontot, az adatok küldéséhez az állomásunknak valamilyen módon hitelesítenie kell magát. A hitelesítés több módon történhet. Erre a leggyakrabban alkalmazott sémát nyílt hitelesítésnek (open authentication) nevezik, ahol a hálózathoz tetszõleges állomás csatlakozhat és kommunikálhat vele. Ezt a típusú hitelesítést akkor érdemes használni, amikor a vezeték nélküli hálózatunkat teszteljük. Más sémákban az adatfolyam megindításához egy titkosítási kézfogás szükséges, vagy elõre megosztott kulcsok esetleg jelszavak segítségével, vagy bonyolultabb sémák esetében itt még olyan különbözõ háttérszolgáltatások is megjelennek, mint például a RADIUS. A legtöbb felhasználó a nyílt hitelesítést használja, ami egyben az alapértelmezés is. A másik legelterjedtebb beállítás a WPA-PSK, avagy WPA Personal, amelyrõl lentebb még szólni fogunk.

Ha Apple® AirPort® Extreme Base Station típusú hozzáférési pontunk van, akkor az osztott kulcsú hitelesítés mellett egy WEP kulcsot is be állítanunk. Ezt az /etc/rc.conf állományban vagy a wpa_supplicant(8) programban tehetjük meg. Ha egyetlen AirPort® bázisállomásunk van, akkor az elérést valahogy így tudjuk beállítani:

```
wlans_ath0="wlan0"
ifconfig_wlan0="authmode shared wepmode on weptxkey 1 wepkey 01234567
DHCP"
```

Általánosságban véve elmondhatjuk, hogy az osztott kulcsú hitelesítést inkább kerüljük el, mivel WEP kulcsok használatára alapszik és ráadásul olyan módon, hogy nagyon könnyû feltörni. Ha már mindenképpen a WEP mellett kell döntenünk (például a régebbi eszközökkel így tudunk csak kompatibilisek maradni), akkor jobban járunk, ha a `nyílt` hitelesítéshez alkalmazzuk. A WEP használatát érintõ további információkat a WEPban találjuk.

IP-cím szerzése DHCP használatával

Miután kiválasztottunk egy hozzáférési pontot és beállítottuk a hitelesítés paramétereit, egy IP-cím is kelleni fog a kommunikációhoz. Az esetek túlnyomó részében DHCP-n keresztül kapunk IP-címet a vezeték nélküli kapcsolatunkhoz. Ezt úgy érhetjük el, ha egyszerûen megnyitjuk az /etc/rc.conf állományt és az alábbihoz hasonló módon felvesszük a `DHCP` paramétert az eszközünk beállításaihoz:

```
wlans_ath0="DHCP"
ifconfig_wlan0="DHCP"
```

Így már készen is állunk a vezeték nélküli felület használatára:

```
# /etc/rc.d/netif start
```

Ahogy a felület mûködõképessé válik, az `ifconfig` parancs segítségével ellenõrizni is tudjuk az ath0 felület állapotát:

```
# ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether 00:11:95:d5:43:62
        inet 192.168.1.100 netmask 0xffffff00 broadcast 192.168.1.255
        media: IEEE 802.11 Wireless Ethernet OFDM/54Mbps mode 11g
        status: associated
        ssid dlinkap channel 11 (2462 Mhz 11g) bssid 00:13:46:49:41:76
        country US ecm authmode OPEN privacy OFF txpower 21.5 bmiss 7
        scanvalid 60 bgscan bgscanintvl 300 bgscanidle 250 roam:rssi 7
        roam:rate 5 protmode CTS wme burst
```

A `status: associated` azt jelenti, hogy sikeresen csatlakoztunk egy vezeték nélküli hálózathoz (jelen esetben ez a `dlinkap`). A `bssid 00:13:46:49:41:76` rész a hozzáférési pont MAC-címét tartalmazza. Az `authmode OPEN` pedig arról számol be, hogy a kommunikáció nem titkosított.

Statikus IP-cím

Ha valami okból nem tudjuk az IP-címünket DHCP szerveren keresztül lekérni, beállíthatunk rögzített IP-címet is. Ehhez nem kell mást tennünk, mint a korábban bemutatott `DHCP` kulcsszót kicserélni egy konkrét címmel. A hozzáférési ponthoz megadott többi paramétert azonban feltétlenül hagyjuk meg:

```
wlans_ath0="wlan0"
ifconfig_wlan0="inet 192.168.1.100 netmask 255.255.255.0ssid saját_ssidi"
```

WPA

A WPA (Wi-Fi Protected Access, vagyis védett wi-fi hozzáférés) a 802.11 szabványokban használatos biztonsági protokoll, amelyet a WEP gyengeségeinek és megfelelõ hitelesítésének ellensúlyozására dolgoztak ki. A WPA a 802.1X hitelesítési protokolljait erõsíti és az adat sértetlenségének megõrzésére a WEP helyett több titkosítási algoritmust is felhasznál. A WPA által igényelt egyetlen titkosítás a TKIP (Temporary Key Integrity Protocol, vagyis az ideiglenes kulcs integritási protokoll), amely a WEP által az integritás ellenõrzésére és a bejutások észlelésére és azok reagálására szánt alap RC4 titkosítást bõvíti ki. A TKIP a régebbi hardvereken csupán szoftveres módosítással mûködõképessé tehetõ. Ez a kompromisszum a védelmet ugyan növeli, de még mindig kevés a támadások megfelelõ elhárításához. A WPA a TKIP mellett tartalmazza még az AES-CCMP titkosítást is, és ennek a használata javasolt. Ezt a specifikációt gyakran WPA2 (vagy RSN) néven emlegetik.

A WPA definiál hitelesítési és titkosítási protokollokat. A hitelesítés általában a következõ két technika egyike alapján történik: vagy 802.1X és egy háttérszolgáltatás, például a RADIUS segítségével, vagy egy elõre megosztott kulcsot alkalmazó minimális kézfogással az állomás és a

hozzáférési pont között. Az elõbbit gyakran WPA Enterprise-nak, míg az utóbbit WPA Personalnak hívják. Mivel a legtöbben nem állítanak be egy komplett RADIUS alapú szervert a vezeték nélküli hálózatukhoz, ezért a WPA-PSK a WPA leginkább elterjedten használt változata.

A vezeték nélküli kapcsolat és a hitelesítés (kulcs alapján vagy szerverrel) vezérlését a wpa_supplicant(8) segédprogram végzi. Ennek a programnak mûködéséhez egy konfigurációs állományra van szüksége, amely az /etc/wpa_supplicant.conf néven érhetõ el. Errõl az állományról bõvebb információt a wpa_supplicant.conf(5) man oldalán lelhetünk.

WPA-PSK

A WPA-PSK, más néven WPA-Personal, egy adott jelszó alapján generált elõre megosztott kulcssal (pre-shared key, PSK) mûködik, amit a vezeték nélküli hálózatokban mesterkulcsként használnak. Ez azt jelenti, hogy minden egyes vezeték nélküli felhasználó ugyanazon a kulcson osztozik. A WPA-PSK olyan kis méretû hálózatok esetében megfelelõ, ahol a hitelesítést elvégzõ szerver használata nem lehetséges vagy nem oldható meg.

Mindig igyekezzünk erõs jelszavakat használni, melyek kellõen hosszúak és sokféle karaktert tartalmaznak, és így nehezebben fejthetõek meg vagy törhetõek fel.

Elõször az /etc/wpa_supplicant.conf állományban állítsuk be az SSID-t és a hálózatunkhoz tartozó elõre megosztott kulcsot:

```
network={
  ssid="freebsdap"
  psk="freebsdmall"
}
```

Ezután az /etc/rc.conf állományban jelezzük, hogy a vezeték nélküli eszközt a WPA segítségével állítjuk be és az IP-címet a DHCP szervertõl kérjük el:

```
wlans_ath0="wlan0"
ifconfig_ath0="WPA DHCP"
```

Innentõl már fel is tudjuk éleszteni a felületet:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 5
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 6
DHCPOFFER from 192.168.0.1
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.1
bound to 192.168.0.254 -- renewal in 300 seconds.
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      ether 00:11:95:d5:43:62
```

```
        inet 192.168.0.254 netmask 0xffffff00 broadcast 192.168.0.255
        media: IEEE 802.11 Wireless Ethernet autoselect OFDM/36Mbps mode 11g
        status: associated
        ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
        country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
        AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
        bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
        wme burst roaming MANUAL
```

Kézzel is megpróbálhatjuk elindítani az elõbb elkészített /etc/wpa_supplicant.conf állomány használatával:

```
# wpa_supplicant -i wlan0 -c /etc/wpa_supplicant.conf
Trying to associate with 00:11:95:c3:0d:ac (SSID='freebsdap' freq=2412 MHz)
Associated with 00:11:95:c3:0d:ac
WPA: Key negotiation completed with 00:11:95:c3:0d:ac [PTK=CCMP GTK=CCMP]
CTRL-EVENT-CONNECTED - Connection to 00:11:95:c3:0d:ac completed (auth) [id=0 id_str=]
```

A következõ parancs a dhclient indítása legyen, amivel megszerezzük a DHCP szervertõl az IP-címünket:

```
# dhclient wlan0
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.1
bound to 192.168.0.254 -- renewal in 300 seconds.
# ifconfig wlan0
ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether 00:11:95:d5:43:62
        inet 192.168.0.254 netmask 0xffffff00 broadcast 192.168.0.255
        media: IEEE 802.11 Wireless Ethernet OFDM/36Mbps mode 11g
        status: associated
        ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
        country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
        AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
        bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
        wme burst roaming MANUAL
```

Ha az /etc/rc.conf állományban szerepel a ifconfig_wlan0="DHCP" sor, akkor egyáltalán nem szükséges a dhclient parancs manuális kiadása, mivel a dhclient magától el fog indulni, miután a wpa_supplicant egyeztette a kulcsokat.

Amikor a DHCP nem használható, megadhatunk a statikus IP-címet is, miután a wpa_supplicant sikeresen lebonyolította a hitelesítést:

```
# ifconfig wlan0 inet 192.168.0.100 netmask 255.255.255.0
# ifconfig wlan0
ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
```

```
        ether 00:11:95:d5:43:62
        inet 192.168.0.100 netmask 0xffffff00 broadcast 192.168.0.255
        media: IEEE 802.11 Wireless Ethernet OFDM/36Mbps mode 11g
        status: associated
        ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
        country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
        AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
        bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
        wme burst roaming MANUAL
```

Ha egyáltalán nem használunk DHCP szervert, akkor nekünk kell beállítani az alapértelmezett átjárót és a névszervert is:

```
# route add default alapértelmezett_átjáró
# echo "nameserver névszerver" >> /etc/resolv.conf
```

WPA és EAP-TLS

A másik mód, ahogy a WPA használható, az a 802.1X hitelesítési szerveren keresztül történik, és ebben az esetben a WPA neve WPA-Enterprise. Ez sokkal biztonságosabb a WPA-Personal elõre kiosztott kulcsaival szemben. A WPA-Enterprise az EAP (Extensible Authentication Protocol, azaz Bõvíthetõ hitelesítési protokoll) használatán alapszik.

Az EAP önmaga nem végez titkosítást, mivel úgy alakították ki, hogy magát az EAP protokollt kell egy titkosított járaton keresztül bújtatni. Az EAP hitelesítési módszereinek több típusát is kidolgozták, melyek közül a legismertebbek az EAP-TLS, EAP-TTLS valamint a EAP-PEAP.

Az EAP-TLS (EAP szállítási rétegbeli védelemmel) a vezeték nélküli világban egy nagyon jól támogatott hitelesítési protokoll, mivel ez volt az elsõ EAP módszer, amit a Wi-fi szövetség jóváhagyott. Az EAP-TLS mûködéséhez három tanúsítvány kell: egy hitelesítõ hatóságtól (Certificate Authority, CA), egy a hitelesítést végzõ szervertõl és egy a klienstõl. Ezzel az EAP módszerrel mind a hitelesítõ szerver, mind a vezeték nélküli kliens külön képviselik a saját tanúsítványaikat, és ezeket a szervezetünket hitelesítõ hatóság aláírása alapján ellenõrzik.

A korábbiaknak megfelelõen a beállításokat szintén az /etc/wpa_supplicant.conf állományon keresztül végezzük el:

```
network={
  ssid="freebsdap" ①
  proto=RSN  ②
  key_mgmt=WPA-EAP ③
  eap=TLS ④
  identity="loader" ⑤
  ca_cert="/etc/certs/cacert.pem" ⑥
  client_cert="/etc/certs/clientcert.pem" ⑦
  private_key="/etc/certs/clientkey.pem" ⑧
  private_key_passwd="freebsdmallclient" ⑨
```

```
}
```

① Ez a mezõ adja meg a hálózat nevét (SSID).

② Itt az RSN (IEEE® 802.11i), vagyis a WPA2 protokollt használjuk.

③ A `key_mgmt` sor a kulcskezelési protokollt adja meg. A mi esetünkben ez a WPA lesz, EAP hitelesítéssel: `WPA-EAP`.

④ Ebben a mezõben az EAP módszert nevezzük meg a kapcsolathoz.

⑤ Az `identity` mezõ az EAP esetén használt azonosítót tartalmazza.

⑥ A `ca_cert` mezõ a hitelesítõ hatóság tanúsítványát tároló állomány elérési útvonalát adja meg. Ezt a szerver tanúsítványának hitelesítéséhez használjuk.

⑦ A `client_cert` sor a kliens tanúsítványát tartalmazó állomány elérési útvonalát adja meg. Ennek a vezeték nélküli hálózat minden egyes kliense esetében egyedinek kell lennie.

⑧ A `private_key` mezõ a kliens tanúsítványának privát kulcsát tároló állomány elérési útját adja meg.

⑨ A `private_key_passwd` mezõ a privát kulcshoz tartozó jelmondatot rögzíti.

Az /etc/rc.conf állományba vegyük fel a következõ sorokat:

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

A következõ lépés a felület felébresztése lesz az rc.d eszköz segítségével:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      ether 00:11:95:d5:43:62
      inet 192.168.0.254 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet DS/11Mbps mode 11g
      status: associated
      ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
      country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
      AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
      bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
      wme burst roaming MANUAL
```

Természetesen, ahogy azt már az elõbbiekben is megmutattuk, mindezt manuálisan is el tudjuk végezni a `wpa_supplicant` és az `ifconfig` parancsok segítségével.

WPA és EAP-TTLS

Az EAP-TLS használatakor mind a hitelesítést végzõ szervernek és kliensnek is kell tanúsítvány, azonban az EAP-TTLS (szállítási rétegbeli védelem EAP tunnelen keresztül) esetében a kliensnél ez elhagyható. Ez a módszer nagyjából olyan, mint amit a webes oldalak csinálnak, ahol a webszerverek egy védett SSL tunnelt képeznek még akkor is, amikor a látogatók nem rendelkeznek kliens oldali tanúsítvánnyal. Az EAP-TTLS egy titkosított TLS tunnelen keresztül védi le a hitelesítési adatok forgalmát.

Ezt ismét az /etc/wpa_supplicant.conf állományon keresztül tudjuk beállítani:

```
network={
  ssid="freebsdap"
  proto=RSN
  key_mgmt=WPA-EAP
  eap=TTLS ①
  identity="test" ②
  password="test" ③
  ca_cert="/etc/certs/cacert.pem" ④
  phase2="auth=MD5" ⑤
}
```

① Ebben a mezõben az EAP módszert állítjuk be a kapcsolathoz.

② Az `identity` mezõ a titkosított TLS tunnelen keresztül az EAP hitelesítésnél felhasznált azonosítót adja meg.

③ A `password` tartalmazza az EAP hitelesítésnél használt jelmondatot.

④ A `ca_cert` mezõ hivatkozik a hitelesítõ hatóság tanúsítványát tartalmazó állományra. Ez az állomány kell a szerver tanúsítványának ellenõrzéséhez.

⑤ Ebben a mezõben a titkosított TLS tunnelben használt hitelesítési módszer nevezzük meg. Jelen esetünkben ez az EAP MD5-Challenge használatával. A "belsõ hitelesítés" fázisát gyakran csak "phase2"-nak (2. fázisnak) hívják.

Mindezek mellett még a következõ sorokat is vegyük fel az /etc/rc.conf állományba:

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

Ezután hozzuk mûködésbe a felületet:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
```

```
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      ether 00:11:95:d5:43:62
      inet 192.168.0.254 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet DS/11Mbps mode 11g
      status: associated
      ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
      country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
      AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
      bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
      wme burst roaming MANUAL
```

WPA és EAP-PEAP

A PEAP (Védett EAP) az EAP-TTLS egyik alternatívájaként jött létre. A PEAP módszernek két változata van, melyek közül a leggyakoribb a PEAPv0/EAP-MSCHAPv2. A leírás további részében a PEAP elnevezéssel erre az EAP módszerre fogunk hivatkozni. A PEAP az EAP-TLS után a leginkább alkalmazott szabvány, más szóval, ha a hálózatunkban többféle operációs rendszer is megtalálható, akkor az EAP-TLS után valószínûleg a PEAP lesz a másik, amit mindegyik ismerni fog.

A PEAP hasonló az EAP-TTLS-hez: szerver oldali tanúsítványokkal hitelesíti a klienseket és titkosított TLS tunnelt hoz létre a kliens és a hitelesítést végzõ szerver között, amivel segíti megóvni a hitelesítési információkat. Biztonság szempontjából az EAP-TTLS és a PEAP között az a különbség, hogy a PEAP hitelesítés a felhasználói nevet titkosítatlanul küldi és csak a jelszó megy át a titkosított TLS tunnelen. Az EAP-TTLS egyaránt a TLS tunnelt használja mind a felhasználói név, mind a jelszó esetében.

Az EAP-PEAP beállításait az /etc/wpa_supplicant.conf állományba kell felvenni:

```
network={
  ssid="freebsdap"
  proto=RSN
  key_mgmt=WPA-EAP
  eap=PEAP ①
  identity="test" ②
  password="test" ③
  ca_cert="/etc/certs/cacert.pem" ④
  phase1="peaplabel=0" ⑤
  phase2="auth=MSCHAPV2" ⑥
}
```

① Ebben a mezõben megadjuk, az EAP módszert használjuk a kapcsolathoz.

② Az `identity` mezõ az EAP hitelesítés során a titkosított TLS tunnelben átküldött azonosítót tartalmazza.

③ A `password` mezõ az EAP hitelesítés során használt jelmondatot definiálja.

④ A `ca_cert` mezõ a hitelesítõ hatóság tanúsítványát tartalmazó állomány elérési útját adja meg. Ez az állomány kell a szerver tanúsítványának ellenõrzéséhez.

⑤ Ez a mezõ a hitelesítés elsõ fázisának (vagyis a TLS tunnel) paramétereit tartalmazza. A

hitelesítést végzõ szervertõl függõen a hitelesítéshez meg kell adnunk bizonyos címkéket. A legtöbb esetben a címke a "kliens oldali EAP titkosítás" lesz, amit a `peaplabel=0` használatával állítunk be. A részleteket a wpa_supplicant.conf(5) man oldalon olvashatjuk.

⑥ Ebben a mezõben a titkosított TLS tunnelben alkalmazott hitelesítést protokollt nevezzük meg. A PEAP esetében ez az `auth=MSCHAPV2` lesz.

A következõket kell még hozzátennünk az /etc/rc.conf állományhoz:

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

Ezután már mûködésbe is hozhatjuk a felületet:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      ether 00:11:95:d5:43:62
      inet 192.168.0.254 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet DS/11Mbps mode 11g
      status: associated
      ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
      country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
      AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
      bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
      wme burst roaming MANUAL
```

WEP

A WEP (Wired Equivalent Privacy, azaz kábellel egyenértékû titkosság) az eredeti 802.11 szabvány része. Nincs külön hitelesítési mechanizmusa, csupán a hozzáférés-vezérlés egy gyenge formájával találkozhatunk benne, amit azonban könnyen fel lehet törni.

A WEP `ifconfig` parancs használatán keresztül állítható be:

```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 inet 192.168.1.100 netmask 255.255.255.0 \
      ssid saját_hálózat wepmode on weptxkey 3 wepkey 3:0x3456789012
```

- A `weptxkey` utal arra, hogy a küldés során WEP kulcsot használunk. Itt most egy harmadik kulcsot használtunk, amelynek egyeznie kell a hozzáférési pont beállításaival. Ha nem tudjuk pontosan, hogy milyen kulcsot használ a hozzáférési pont, akkor próbálkozzunk az `1` érték (vagyis az elsõ kulcs) megadásával.

- A wepkey után következik a kiválasztott WEP kulcs. *index:kulcs* alakban kell megadni, és ha itt nem adunk meg indexet, akkor azzal az 1 indexû kulcsot állítjuk be. Úgyis fogalmazhatnánk, hogy az indexet csak olyankor kell megadni, amikor nem az elsõ kulcsot akarjuk használni.

A 0x3456789012 értéket a hozzáférési pontnál beállított kulcsra kell beállítani.

Ha érdekelnek minket a további részletek, akkor bátran lapozzuk fel az ifconfig(8) parancs man oldalát.

A wpa_supplicant segédprogramot is bevonhatjuk a vezeték nélküli felületek WEP alapú használatába. A fenti példát a következõ módon tudjuk leírni az /etc/wpa_supplicant.conf állományban:

```
network={
  ssid="sajat_halozat"
  key_mgmt=NONE
  wep_key3=3456789012
  wep_tx_keyidx=3
}
```

Majd:

```
# wpa_supplicant -i wlan0 -c /etc/wpa_supplicant.conf
Trying to associate with 00:13:46:49:41:76 (SSID='dlinkap' freq=2437 MHz)
Associated with 00:13:46:49:41:76
```

31.3.4. Az ad-hoc mûködési mód

Az IBSS vagy más néven ad-hoc módot pont-pont típusú kapcsolatok kialakítására tervezték. Például, ha az A és a B gépek között egy ad-hoc típusú hálózatot akarunk létesíteni, akkor egyszerûen csak ki kell választanunk két IP-címet és egy SSID-t.

Így állítjuk be az A gépet:

```
# ifconfig wlan0 create wlandev ath0 wlanmode adhoc
# ifconfig wlan0 inet 192.168.0.1 netmask 255.255.255.0 ssid freebsdap
# ifconfig wlan0
  wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
      ether 00:11:95:c3:0d:ac
      inet 192.168.0.1 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <adhoc>
      status: running
      ssid freebsdap channel 2 (2417 MHz) bssid 02:11:95:c3:0d:ac
      country US ecm authmode OPEN privacy OFF txpower 21.5 scanvalid 60
      protmode CTS wme burst
```

Az adhoc paraméterrel utalunk arra, hogy a felület most IBSS módban mûködik.

A B gépen ezután már képesek vagyunk észlelni az A gépet:

```
# ifconfig wlan0 create wlandev ath0 wlanmode adhoc
  SSID/MESH ID    BSSID              CHAN RATE  S:N     INT CAPS
  freebsdap       02:11:95:c3:0d:ac    2   54M -64:-96  100 IS   WME
```

A kimenetben szereplõ I is megerõsíti, hogy az A gépet ad-hoc módban érjük el. Így már csak a B gépet kell beállítanunk egy másik IP-címmel:

```
# ifconfig wlan0 inet 192.168.0.2 netmask 255.255.255.0 ssid freebsdap
# ifconfig wlan0
  wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
      ether 00:11:95:d5:43:62
      inet 192.168.0.2 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <adhoc>
      status: running
      ssid freebsdap channel 2 (2417 Mhz 11g) bssid 02:11:95:c3:0d:ac
      country US ecm authmode OPEN privacy OFF txpower 21.5 scanvalid 60
      protmode CTS wme burst
```

Most már mind az A és mind a B készen áll az adatok cseréjére.

31.3.5. FreeBSD alapú hozzáférési pontok

A FreeBSD képes hozzáférési pontként (Access Point, AP) is üzemelni, így nem kell külön hardveres hozzáférési pontot vásárolnunk vagy ad-hoc hálózatot használnunk. Ez különösen akkor hasznos, amikor a FreeBSD gépet egy másik hálózat (például az internet) felé állítottuk be átjárónak.

Alapvetõ beállítások

Mielõtt nekiállnánk a FreeBSD-s gépünket hozzáférési pontnak beállítani, egy olyan rendszermagra lesz szükségünk, amely tartalmazza a megfelelõ vezeték nélküli támogatást a kártyánkhoz. Emellett az alkalmazni kívánt biztonsági protokollok támogatását is bele kell építenünk. Ennek részleteit lásd a Kezdeti beállításokban.

Jelenleg az NDIS meghajtón keresztül használt Windows®-os meghajtók nem teszik lehetõvé hozzáférési pontok kialakítását. Egyedül a vezeték nélküli eszközök natív FreeBSD-s meghajtói ismerik a hozzáférési pont módot.

Ahogy betöltöttük a vezeték nélküli hálózatok támogatását, egybõl ellenõrizni is tudjuk, hogy a vezeték nélküli eszközünk használható-e hozzáférési pontként (avagy "hostap" módban):

```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 list caps
drivercaps=6f85edc1<STA,FF,TURBOP,IBSS,HOSTAP,AHDEMO,TXPMGT,SHSLOT,SHPREAMBLE,MONITOR,
```

```
MBSS,WPA1,WPA2,BURST,WME,WDS,BGSCAN,TXFRA>
crytocaps=1f<WEP,TKIP,AES,AES_CCM,TKIPMIC>
```

A fenti kimenetben láthatjuk a kártyánk tulajdonságait. A HOSTAP szó arról tanúskodik, hogy a vezeték nélküli kártyánk képes hozzáférési pontként viselkedni. Mellette még a különféle támogatott titkosítási módszerek is láthatóak: WEP, TKIP, AES stb. Ezekből az információkból tudjuk kideríteni, hogy a hozzáférési pontunkon milyen titkosítási protokollokat tudunk használni.

A vezeték nélküli eszközünket innentől már csak hozzáférési pontnak állíthatjuk át a viruális hálózati eszköz létrehozásakor, ezért a korábban létrehozott eszközt ehhez elõször meg kell semmisítenünk:

```
# ifconfig wlan0 destroy
```

Ezzel létrejön a megfelelõ beállításokkal, majd ezekhez állítjuk még be a többit:

```
# ifconfig wlan0 create wlandev ath0 wlanmode hostap
# ifconfig wlan0 inet 192.168.0.1 netmask 255.255.255.0 ssid freebsdap mode 11g
channel 1
```

Az ifconfig parancs ismételt használatával le is tudjuk kérdezni az wlan0 felület állapotát:

```
# ifconfig wlan0
  wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
      inet 192.168.0.1 netmask 0xffffff00 broadcast 192.168.0.255
      inet6 fe80::211:95ff:fec3:dac%ath0 prefixlen 64 scopeid 0x4
      ether 00:11:95:c3:0d:ac
      media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <hostap>
      status: running
      ssid freebsdap channel 1 (2412 Mhz) bssid 00:11:95:c3:0d:ac
      country US ecm authmode OPEN privacy OFF txpower 21.5 scanvalid 60
      protmode CTS wme burst dtimperiod 1 -dfs
```

A hostap paraméterbõl kiderül, hogy a felület hozzáférési pont módban van.

Ha az /etc/rc.conf állományban megadjuk a következõ sorokat, akkor a felület beállítása a rendszer indításakor magától megtörténik:

```
wlans_ath0="wlan0"
create_args_wlan0="wlanmode hostap"
ifconfig_wlan0="inet 192.168.0.1 netmask 255.255.255.0 ssid freebsdap mode 11g channel
1"
```

Hitelesítés vagy titkosítás nélküli hozzáférési pontok

Habár a hozzáférési pontok mûködtetése nem javasolt hitelesítés vagy titkosítás nélkül, ebben a módban könnyen meg tudunk gyõzõdni a hozzáférési pontunk használhatóságáról. Ez a típusú konfiguráció ezenkívül még fontos szerepet játszik a klienseken felbukkanó hibák kiszûrésében is.

Miután sikerült az elõbbiekben bemutatottak alapján beállítani a hozzáférési pontunkat, egy másik vezeték nélküli géprõl rögtön meg is kezdhetjük a keresését:

```
# ifconfig ath0 up scan
SSID/MESH ID    BSSID              CHAN RATE   S:N     INT CAPS
freebsdap       00:11:95:c3:0d:ac    1   54M -66:-96  100 ES   WME
```

Láthatjuk, hogy a kliens megtalálta a hozzáférési pontot és tudunk is rá kapcsolódni:

```
# ifconfig wlan0 inet 192.168.0.2 netmask 255.255.255.0 ssid freebsdap
# ifconfig wlan0
  wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
      inet 192.168.0.2 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet OFDM/54Mbps mode 11g
      status: associated
      ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
      country US ecm authmode OPEN privacy OFF txpower 21.5 bmiss 7
      scanvalid 60 bgscan bgscanintvl 300 bgscanidle 250 roam:rssi 7
      roam:rate 5 protmode CTS wme burst
```

WPA titkosítást használó hozzáférési pontok

Ebben a szakaszban a FreeBSD-s hozzáférési pontunkat WPA titkosítással állítjuk be. A WPA és a WPA alapú kliensek beállításának részleteit a WPAban találjuk.

A WPA titkosítást használó hozzáférési pontokon a hostapd démon foglalkozik a kliensek hitelesítésével és a kulcsok kezelésével.

A továbbiakban az összes beállítást egy olyan FreeBSD-s gépen végezzük el, amely hozzáférési pontként mûködik. Ahogy sikerült beállítanunk a hozzáférési pont módot, az /etc/rc.conf állományban a következõ sor segítségével könnyen meg tudjuk oldani, hogy az hostapd démon a rendszerrel együtt magától elinduljon:

```
hostapd_enable="YES"
```

Mielõtt megpróbálnánk beállítani a hostapd démont, ne felejtsük el elvégezni a Alapvetõ beállításokban említett alapvetõ beállításokat sem.

WPA-PSK

A WPA-PSK használatát olyan kis méretû hálózatok számára szánják, ahol egy külön hitelesítõ

szervert alkalmazása nem lehetséges vagy nem kívánatos.

A konfiguráció az /etc/hostapd.conf állományon keresztül történik:

```
interface=wlan0 ①
debug=1 ②
ctrl_interface=/var/run/hostapd ③
ctrl_interface_group=wheel ④
ssid=freebsdap ⑤
wpa=1 ⑥
wpa_passphrase=freebsdmall ⑦
wpa_key_mgmt=WPA-PSK ⑧
wpa_pairwise=CCMP TKIP ⑨
```

① Ebben a mezõben jelöljük ki a hozzáférési pontként használt vezeték nélküli felületet.

② Ebben a mezõben adjuk meg a hostapd futtatása során keletkezõ üzenetek részletességét. A példában szereplõ 1 érték ennek a legkisebb szintjét jelöli.

③ A `ctrl_interface` mezõ megadja a hostapd által használt könyvtár elérési útvonalát, amiben azokat a tartományokhoz tartozó socketeket tároljuk, amelyeken keresztül olyan programokkal tudunk kommunikálni, mint például a hostapd_cli(8). Itt az alapértelmezett értéket írtuk be.

④ A `ctrl_interface_group` sor beállítja azt a csoportot (ez jelen esetben a `wheel`), amin keresztül a vezérlõfelület (control interface) állományaihoz hozzá tudunk férni.

⑤ Ebben a mezõben a hálózat nevét állítjuk be.

⑥ A `wpa` mezõvel engedélyezzük a WPA használatát és megadjuk, hogy melyik WPA hitelesítési protokollt alkalmazzuk. Az itt szereplõ 1 érték a WPA-PSK hitelesítés állítja be a hozzáférési pont számára.

⑦ A `wpa_passphrase` mezõ a WPA hitelesítéshez szükséges ASCII jelmondatot tartalmazza.

⑧ A `wpa_key_mgmt` sor a kulcsok kezelésére használt protokollt definiálja. Ez a mi esetünk most a WPA-PSK.

⑨ A `wpa_pairwise` mezõ a hozzáférési pont által elfogadott titkosítási algoritmusokat határozza meg. A példában a TKIP (WPA) és CCMP (WPA2) titkosítást is támogatjuk. A CCMP titkosítás a TKIP egyik alternatívája, és lehetõség szerint használjuk ezt. A TKIP csak olyan állomások esetében javasolt, amelyek nem támogatják a CCMP használatát.

A következõ lépés a hostapd elindítása:

```
# /etc/rc.d/hostapd forcestart
```

```
# ifconfig wlan0
  wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 2290
      inet 192.168.0.1 netmask 0xffffff00 broadcast 192.168.0.255
      inet6 fe80::211:95ff:fec3:dac%ath0 prefixlen 64 scopeid 0x4
      ether 00:11:95:c3:0d:ac
      media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <hostap>
```

```
      status: associated
      ssid freebsdap channel 1 bssid 00:11:95:c3:0d:ac
      authmode WPA2/802.11i privacy MIXED deftxkey 2 TKIP 2:128-bit txpowmax 36
protmode CTS dtimperiod 1 bintval 100
```

A hozzáférési pont mostantól mûködik, innentõl a kliensek már képesek csatlakozni hozzá, bõvebben lásd a WPAban. A hozzáférési ponthoz tartozó állomásokat az `ifconfig wlan0 list sta` paranccsal tudjuk listázni.

WEP titkosítást használó hozzáférési pontok

A WEP titkosítást nem javasoljuk a hozzáférési pontok esetében, mivel nem tartalmaz semmilyen hitelesítési mechanizmust és könnyen feltörhetõ. Egyes régebbi vezeték nélküli kártyák azonban csak a WEP által nyújtott védelmet ismerik, ezért az ilyenek csak olyan hozzáférési pontokhoz tudnak csatlakozni, amelyek vagy nem használnank hitelesítést és titkosítást, vagy erre a WEP protokollt használják.

A vezeték nélküli eszközt tegyük hozzáférési pont módba és állítsuk be neki a megfelelõ SSID-t és IP-címet:

```
# ifconfig wlan0 create wlandev ath0 wlanmode hostap
# ifconfig wlan0 inet 192.168.0.1 netmask 255.255.255.0 \
    ssid freebsdap wepmode on weptxkey 3 wepkey 3:0x3456789012 mode 11g
```

- A `weptxkey` beállítás után adjuk meg a küldéshez használt WEP kulcsot. Itt a harmadik kulcsot adtuk meg (vegyük észre, hogy a kulcsok számozása az 1 értékkel kezdõdik). Ez a paramétert az adatok tényleges titkosításához kell megadni.
- A `wepkey` a kiválasztott WEP kulcs beállítását jelöli, aminek a formátuma *index:kulcs*. Ha itt nem adunk meg indexet, akkor automatikusan az elsõ kulcsot állítjuk be. Ezért talán mondanunk sem kell, hogy az indexet csak akkor kell megadni, ha nem az elsõ kulcsot akarjuk használni.

A wlan0 felület állapotának megtekintéséhez adjuk ki megint az `ifconfig` parancsot:

```
# ifconfig wlan0
  ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      inet 192.168.0.1 netmask 0xffffff00 broadcast 192.168.0.255
      ether 00:11:95:c3:0d:ac
      media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <hostap>
      status: running
      ssid freebsdap channel 4 (2427 Mhz) bssid 00:11:95:c3:0d:ac
      country US ecm authmode OPEN privacy ON deftxkey 3 wepkey 3:40-bit
      txpower 21.5 scanvalid 60 protmode CTS wme burst dtimperiod 1 -dfs
```

Egy másik vezeték nélküli géprõl most már megpróbálhatjuk megkeresni a hozzáférési pontot:

```
# ifconfig wlan0 create wlandev ath0
```

```
SSID           BSSID              CHAN RATE  S:N   INT CAPS
freebsdap      00:11:95:c3:0d:ac    1   54M 22:1   100 EPS
```

Láthatjuk, hogy a kliens megtalálta a hozzáférési pontot, és a megfelelõ paraméterekkel (kulcs stb.) képes kapcsolódni hozzá a WEPban leírtak szerint.

31.3.6. A vezetékes és vezeték nélküli hálózatok együttes használata

A vezetékes hálózatok általában jobb teljesítményt nyújtanak és megbízhatóbbak, miközben a vezeték nélküli hálózatok pedig nagyobb rugalmasságot és mozgásteret szolgáltatnak. Ezért a hordozható számítógépek tulajdonosaiban felmerülhet az igény, hogy egyszerre mind a kettõt használva, tetszõlegesen és problémamentesen válthassanak a hálózatok között.

FreeBSD rendszereken ún. "hibatûrõ" módon két vagy akár több hálózati interfészt össze tudunk vonni. Ennek köszönhetõen az aktív hálózati kapcsolat megszünésekor rendszerünk önállóan igyekszik mindig a fennmaradó elérhetõ hálózatok közül a leginkább preferáltabbra váltani.

A hálózati összeköttetések összefûzésével és a hibatûrés konkrét megvalósításával az Linkek összefûzése és hibatûrésében foglalkozunk, ahol a Hibatûrés beállítása vezetékes és vezeték nélküli hálózatok közöttben láthatjuk is a vezetékes és vezeték nélküli kapcsolatok együttes használatának beállítását.

31.3.7. Hibaelhárítás

Ha valamilyen gondunk lenne a vezeték nélküli hálózatok használatával, akad néhány lépés, amivel esetleg fel tudjuk deríteni a hiba okát.

- Ha nem látjuk a hozzáférési pontot a pásztázás után, ellenõrizzük, hogy a vezeték nélküli eszközt véletlenül nem korlátoztuk-e le bizonyos csatornákra.
- Ha nem tudunk csatlakozni a hozzáférési ponthoz, akkor egyeztessük vele az állomás egyes paramétereit, beleértve a hitelesítési sémát és a biztonsági protokollokat. Minél jobban egyszerûsítsük le a konfigurációkat. Ha WPA vagy WEP titkosítást használunk, akkor a hozzáférési ponton állítsunk be nyílt hitelesítést és kapcsoljuk ki a titkosítást, majd nézzük meg, hogy így eljut-e hozzánk valamilyen forgalom.
- Ahogy sikerült csatlakozunk a hozzáférési ponthoz, a biztonsági beállításokat olyan egyszerû eszközökkel próbáljuk meg diagnosztizálni, mint például a ping(8).

  A `wpa_supplicant` segédprogrammal tudunk nyomkövetést végezni. A `-dd` opció megadásával indítsuk el manuálisan és ellenõrizzük a rendszernaplókat.

- Vannak alacsonyabb szintû nyomkövetési lehetõségek is. A 802.11 protokollt támogató rétegben is tudunk engedélyezni nyomkövetési üzeneteket a /usr/src/tools/tools/net80211 könyvtárban található `wlandebug` program segítségével. Például a

  ```
  # wlandebug -i ath0 +scan+auth+debug+assoc
    net.wlan.0.debug: 0 => 0xc80000<assoc,auth,scan>
  ```

paranccsal a hozzáférési pontok kereséséhez és a 802.11 protokollon belül a kapcsolat megszervezéséhez szükséges kézfogásokhoz kapcsolódó konzolüzeneteket tudjuk engedélyezni.

A 802.11 rétegben rengeteg hasznos statisztikát találhatunk. Mindezeket a `wlanstats` eszközzel tudjuk kiíratni. Ezeknek a statisztikáknak a 802.11 réteg összes hibáját be kell tudniuk azonosítaniuk. Vigyázzunk azonban, mert az eszközmeghajtókban a 802.11 réteg alatt rejlõ bizonyos hibák ilyenkor nem jelennek meg. Az eszközfüggõ problémák felderítésével kapcsolatban a megfelelõ meghajtó dokumentációját olvassuk át.

Amennyiben a fenti tanácsok mentén sem sikerül orvosolnunk a hibát okát, küldjünk egy hibajelentést és mellékeljük hozzá a fentebb tárgyalt eszközök által gyártott kimeneteket.

31.4. Bluetooth

31.4.1. Bevezetés

A Bluetooth egy olyan vezeték nélküli technológia, amellyel a 2,4 GHz-es frekvenciatartományban tudunk személyi hálózatokat létrehozni 10 méteren belül. Az ilyen típusú hálózatok általában alkalmi jelleggel keletkeznek különféle hordozható eszközök, mint például mobiltelefonok, kézi számítógépek és laptopok között. Eltérõen más népszerû vezeték nélküli technológiáktól, például a wi-fitõl, a Bluetooth magasabb szintû szolgáltási profilokat is felajánl: FTP-szerû állományszervereket, az állományok áttolását, hang átküldését, soros vonali emulációt és még sok minden mást.

A FreeBSD-ben megvalósított Bluetooth protokollkészlet a Netgraph rendszerre építkezik (lásd netgraph(4)). A Bluetooth alapú USB-s hardverzárak széles körét támogatja az ng_ubt(4) meghajtó. A Broadcom BCM2033 chipre épített Bluetooth eszközöket az ubtbcmfw(4) és az ng_ubt(4) meghajtók támogatják. A 3Com Bluetooth PC Card 3CRWB60-A eszközt az ng_bt3c(4) meghajtó támogatja. A soros és UART alapú Bluetooth eszközöket a sio(4), ng_h4(4) és hcseriald(8) ismeri. Ebben a szakaszban a Bluetooth alapú USB-s hardverzárak használatát mutatjuk be.

31.4.2. Az eszköz csatlakoztatása

Alapértelmezés szerint a Bluetooth eszközmeghajtók modulként érhetõek el. Az eszköz csatlakoztatása elõtt a megfelelõ meghajtót be kell töltenünk a rendszermagba:

```
# kldload ng_ubt
```

Ha a Bluetooth eszköz már a rendszer indításakor is jelen van, akkor a modult az /boot/loader.conf állományon keresztül is betölthetjük:

```
ng_ubt_load="YES"
```

Dugjuk be az USB-s hardverzárunkat. Az alábbihoz hasonló kimenet fog keletkezni a konzolon (vagy a rendszernaplóban):

```
ubt0: vendor 0x0a12 product 0x0001, rev 1.10/5.25, addr 2
ubt0: Interface 0 endpoints: interrupt=0x81, bulk-in=0x82, bulk-out=0x2
ubt0: Interface 1 (alt.config 5) endpoints: isoc-in=0x83, isoc-out=0x3,
      wMaxPacketSize=49, nframes=6, buffer size=294
```

Az /etc/rc.d/bluetooth szkript fogja végezni a Bluetooth használatához szükséges protokollkészlet elindítását és leállítását. Jó ötlet leállítani az eszköz eltávolítása elõtt, de ha elhagyjuk, (általában) nem okoz végzetes hibát. Az indításkor a következõ kimenetet kapjuk:

```
# /etc/rc.d/bluetooth start ubt0
BD_ADDR: 00:02:72:00:d4:1a
Features: 0xff 0xff 0xf 00 00 00 00 00
<3-Slot> <5-Slot> <Encryption> <Slot offset>
<Timing accuracy> <Switch> <Hold mode> <Sniff mode>
<Park mode> <RSSI> <Channel quality> <SCO link>
<HV2 packets> <HV3 packets> <u-law log> <A-law log> <CVSD>
<Paging scheme> <Power control> <Transparent SCO data>
Max. ACL packet size: 192 bytes
Number of ACL packets: 8
Max. SCO packet size: 64 bytes
Number of SCO packets: 8
```

31.4.3. Host Controller Interface (HCI)

A Host Controller Interface (HCI) egy parancsfelületet nyújt a mûködési sáv vezérlõjéhez (baseband controller) és az összeköttetések kezelõjéhez (link manager), valamint hozzáférést a hardverállapot és -vezérlõ regiszterekhez. Ez a felület egy egységes módszert szolgáltat a Bluetooth mûködési sávjához tartozó tulajdonságok eléréséhez. Az eszközön üzemelõ HCI réteg a Bluetooth hardverben található HCI firmware-rel vált adatokat és parancsokat. A Host Controller Transport Layer (vagyis a fizikai busz) meghajtója mind a két HCI réteget és a kettejük közti információcserét is elérhetõvé teszi.

Az egyes Bluetooth eszközökhöz létrejön egy-egy *hci* típusú Netgraph-beli csomópont. Ez a HCI csomópont általában a Bluetooth eszközmeghajtó csomópontjához (lefelé) és az L2CAP csomóponthoz (felfelé) csatlakozik. Az összes HCI mûveletet a HCI csomóponton kell elvégezni és nem az eszközmeghajtóhoz tartozón. A HCI csomópont alapértelmezett neve a "devicehci". Ezekrõl többet az ng_hci(4) man oldalán tudhatunk meg.

Az egyik legáltalánosabb feladat a Bluetooth eszközök esetében a közelben levõ további eszközök felderítése. Ezt a mûveletet *tudakozódásnak* ("inquiry") nevezik. A tudakozódást és az összes többi HCI-hez kapcsolódó mûveletet a hccontrol(8) segédprogrammal tudjuk elvégezni. A lentebb látható példa azt mutatja meg, hogyan tudunk Bluetooth eszközöket keresni egy adott távolságon belül. Az elérhetõ eszközök listáját néhány másodpercen alatt megkapjuk. A távoli azonban eszközök csak akkor fognak válaszolni, ha *felderíthetõ* ("discoverable") módban vannak.

```
% hccontrol -n ubt0hci inquiry
Inquiry result, num_responses=1
```

```
Inquiry result #0
       BD_ADDR: 00:80:37:29:19:a4
       Page Scan Rep. Mode: 0x1
       Page Scan Period Mode: 00
       Page Scan Mode: 00
       Class: 52:02:04
       Clock offset: 0x78ef
Inquiry complete. Status: No error [00]
```

A BD_ADDR a Bluetooth eszköz egyedi címe, hasonló a hálózati kártyák MAC-céméhez. Erre a címre lesz szükség ahhoz, hogy a továbbiakban kommunikálni tudjunk az eszközzel. Emberek számára értelmezhetõ nevet is hozzá tudunk rendelni a BD_ADDR címhez. Az /etc/bluetooth/hosts állomány tartalmazza a Bluetooth eszközökre vonatkozó információkat. A következõ példában azt láthatjuk, hogyan tudunk beszédesebb nevet adni egy távoli eszköznek:

```
% hccontrol -n ubt0hci remote_name_request 00:80:37:29:19:a4
BD_ADDR: 00:80:37:29:19:a4
Name: Pav T39-ese
```

Amikor tudakozódni kezdünk a távoli Bluetooth eszközök jelenléte felõl, a gépünket "sajat.gep.nev (ubt0)" néven fogják látni. Ez a helyi eszközhöz rendelt név bármikor megváltoztatható.

A Bluetooth rendszer lehetõség ad pont-pont (természetesen csak két Bluetooth egység között) vagy pont-multipont típusú kapcsolatok kiépítésére. A pont-multipont kapcsolat esetén a kapcsolaton több Bluetooth eszköz osztozik. A most következõ példában megláthatjuk, hogyan kell az aktív mûködési sávban lekérdezni a helyi eszköz létrejött kapcsolatait:

```
% hccontrol -n ubt0hci read_connection_list
Remote BD_ADDR    Handle Type Mode Role Encrypt Pending Queue State
00:80:37:29:19:a4     41  ACL    0 MAST    NONE       0     0 OPEN
```

A *kapcsolat azonosítója* (connection handle) akkor hasznos, amikor egy sávbeli kapcsolatot akarunk lezárni. Ezt általában nem kell kézzel megcsinálni. A rendszer magától lezárja az inaktív sávbeli kapcsolatokat.

```
# hccontrol -n ubt0hci disconnect 41
Connection handle: 41
Reason: Connection terminated by local host [0x16]
```

A `hccontrol help` paranccsal tudjuk lekérdezni az elérhetõ HCI parancsokat. A legtöbb HCI parancs végrehajtásához nem kellenek rendszeradminisztrátori jogosultságok.

31.4.4. Logical Link Control and Adaptation Protocol (L2CAP)

A Logical Link Control and Adaptation Protocol (L2CAP) a kapcsolat-orientált és a kapcsolat nélküli adatszolgáltatásokért felelõs a felsõbb rétegek felé, valamit támogatja a protokollok többszörözését,

a darabolást és az összerakást. Az L2CAP a magasabb szintû protokollok és az alkalmazások számára egészen 64 kilobyte méretig lehetõvé teszi az adatcsomagok küldését és fogadását.

A L2CAP a *csatorna* (channel) fogalmára építkezik. A csatorna egy logikai kapcsolatot képvisel a mûködési sávon belüli kapcsolat felett. Mindegyik csatornához egyetlen protokoll kötõdik, egy a többhöz alapon. Több csatorna is tarthozhat ugyanahhoz a protokollhoz, de egy csatornán nem használhatunk több protokollt. A csatornákon keresztül érkezõ L2CAP csomagok ezután a megfelelõ felsõbb rétegbeli protokollokhoz kerülnek. Több csatorna osztozhat ugyanazon a sávbeli kapcsolaton.

Minden Bluetooth eszközhöz létrejön egy *l2cap* típusú Netgraph-csomópont. Az L2CAP csomópont általában egy Bluetooth HCI csomóponthoz (lefelé) és egy Bluetooth sockethez (felfelé) kapcsolódik. Az L2CAP csomópont alapértelmezett neve "devicel2cap". Errõl részletesebben az ng_l2cap(4) man oldal világosít fel minket.

Ezen a szinten hasznos parancsnak bizonyulhat az l2ping(8), amivel más eszközöket tudunk pingelni. Elõfordulhat, hogy egyes Bluetooth implementációk nem válaszolnak semmilyen feléjük küldött adatra, így az alábbi példában is szereplõ `0 bytes` teljesen normális.

```
# l2ping -a 00:80:37:29:19:a4
0 bytes from 0:80:37:29:19:a4 seq_no=0 time=48.633 ms result=0
0 bytes from 0:80:37:29:19:a4 seq_no=1 time=37.551 ms result=0
0 bytes from 0:80:37:29:19:a4 seq_no=2 time=28.324 ms result=0
0 bytes from 0:80:37:29:19:a4 seq_no=3 time=46.150 ms result=0
```

Az l2control(8) segédprogram használható az L2CAP csomópontok különbözõ mûveleteinek kivitelezésére. Ebben a példában a helyi eszközhöz tartozó logikai kapcsolatokat (csatornák) és sávokat kérdezzük le:

```
% l2control -a 00:02:72:00:d4:1a read_channel_list
L2CAP channels:
Remote BD_ADDR     SCID/ DCID   PSM  IMTU/ OMTU State
00:07:e0:00:0b:ca    66/   64     3   132/  672 OPEN
% l2control -a 00:02:72:00:d4:1a read_connection_list
L2CAP connections:
Remote BD_ADDR    Handle Flags Pending State
00:07:e0:00:0b:ca     41 O           0 OPEN
```

Másik ugyanilyen diagnosztikai eszköz a btsockstat(1). Ha a viselkedését tekintjük, akkor leginkább a netstat(1) programra hasonlít, de a Bluetooth hálózatban megjelenõ adatszerkezetekkel dolgozik. Az alábbi példa az iménti l2control(8) parancs kimenetében szereplõ logikai kapcsolatokat mutatja:

```
% btsockstat
Active L2CAP sockets
PCB      Recv-Q Send-Q Local address/PSM       Foreign address   CID   State
c2afe900      0      0 00:02:72:00:d4:1a/3     00:07:e0:00:0b:ca 66    OPEN
Active RFCOMM sessions
```

```
L2PCB    PCB      Flag MTU   Out-Q DLCs State
c2afe900 c2b53380 1    127   0     Yes  OPEN
Active RFCOMM sockets
PCB      Recv-Q Send-Q Local address     Foreign address   Chan DLCI State
c2e8bc80      0    250 00:02:72:00:d4:1a 00:07:e0:00:0b:ca 3    6    OPEN
```

31.4.5. Az RFCOMM protokoll

Az RFCOMM protokoll a soros portok emulációját valósítja meg az L2CAP protokollon keresztül. A protokoll az ETSI TS 07.10. RFCOMM szabványán alapszik, és egy egyszerû átviteli protokoll, amelyet a 9 tûs RS-232 (EIATIA-232-E) soros portok emulációjára készítettek fel. Az RFCOMM protokoll legfeljebb 60 kapcsolat (RFCOMM csatorna) párhuzamos használatát támogatja két Bluetooth eszköz között.

Az RFCOMM számára a teljes kommunikációs útvonal két különbözõ eszközön futó alkalmazást (kommunikációs végpontot) és köztük levõ kommunikációs szegments foglalja magában. Az RFCOMM az adott eszközön a soros portot használó alkalmazások részére készült. A kommunikációs szegmens az egyik eszköztõl a másikig vezetõ Bluetooth alapú összeköttetés (közvetlen kapcsolat).

Közvetlen kapcsolat esetén az RFCOMM csak az eszközök közti kapcsolattal foglalkozik, valamint hálózati kapcsolat esetén az eszköz és a modem közti kapcsolattal. Az RFCOMM más konfigurációkat is támogat, például olyan modulokat, amelyek az egyik oldalon a Bluetooth vezeték nélküli technológián keresztül kommunikálnak, míg a másik oldalon egy vonalas felületet nyújtanak.

A FreeBSD-ben az RFCOMM protokollt Bluetooth foglalatok rétegében valósították meg.

31.4.6. Az eszközök párosítása

Alapértelmezés szerint a Bluetooth kommunikáció nem hitelesítõdik és bármelyik eszköz képes bármelyik másikkal felvenni a kapcsolatot. Egy Bluetooth eszköz (például egy mobiltelefon) egy adott szolgáltatáshoz igényelhet hitelesítést (például betárcsázáshoz). A Bluetooth alapú hitelesítés többnyire *PIN kódokkal* történik. A PIN kód egy legfeljebb 16 karakterbõl álló ASCII karakterlánc. A felhasználóknak mind a két eszközön ugyanazt a PIN kódot kell megadniuk. Miután megadtuk a PIN kódot, az eszközök létrehoznak hozzájuk egy *összekötettésbeli kulcsot* (link key). Ezután ezt a kulcsot vagy az eszközökön tároljuk vagy pedig valamilyen tartós tárolón. A következõ alkalommal mind a két eszközt ezt a korábban elkészített kulcsot fogja használni. Ezt az eljárást nevezik *párosításnak* (pairing). Ha valamelyik eszköz elveszti az össszeköttetés kulcsát, akkor a párosítást meg kell ismételni.

A hcsecd(8) démon felelõs az összes Bluetooth alapú hitelesítési kérés lekezeléséért. Az alapértelmezett konfigurációs állománya az /etc/bluetooth/hcsecd.conf. Például így tudjuk benne egy mobiltelefonhoz megadni az "1234" PIN kódot:

```
device {
        bdaddr  00:80:37:29:19:a4;
        name    "Pav T39-ese";
```

```
        key     nokey;
        pin     "1234";
      }
```

Semmilyen korlátozás nincs a PIN kódokra (a méretüktõl eltekintve). Egyes eszközökbe (például a Bluetooth fejhallgatók) elõre rögzített PIN kódot építettek bele. A `-d` kapcsoló hatására a hcsecd(8) démont az elõtérben lehet futtatni, így könnyebben láthatjuk mi történik. A távoli eszközt állítsuk be a párosítás elfogadására és kezdeményezzünk felé egy Bluetooth kapcsolatot. A távoli eszköznek erre azt kell válaszolnia, hogy elfogadta a párosítást, majd kérni fogja a PIN kódot. Adjuk meg ugyanazt a PIN kódot, mint amit a hcsecd.conf állományba is beírtunk. Most már a gépünk és a távoli eszköz párban vannak. A párosítást a távoli eszközrõl is kezdeményezhetjük.

A FreeBSD 5.5, 6.1 és újabb változataiban az /etc/rc.conf állományba a következõ sort kell felvenni a hcsecd automatikus indításához:

```
hcsecd_enable="YES"
```

Ez pedig a hcsecd démon által generált kimenetre példa:

```
hcsecd[16484]: Got Link_Key_Request event from 'ubt0hci', remote bdaddr
0:80:37:29:19:a4
hcsecd[16484]: Found matching entry, remote bdaddr 0:80:37:29:19:a4, name 'Pav's T39',
link key doesn't exist
hcsecd[16484]: Sending Link_Key_Negative_Reply to 'ubt0hci' for remote bdaddr
0:80:37:29:19:a4
hcsecd[16484]: Got PIN_Code_Request event from 'ubt0hci', remote bdaddr
0:80:37:29:19:a4
hcsecd[16484]: Found matching entry, remote bdaddr 0:80:37:29:19:a4, name 'Pav's T39',
PIN code exists
hcsecd[16484]: Sending PIN_Code_Reply to 'ubt0hci' for remote bdaddr 0:80:37:29:19:a4
```

31.4.7. Service Discovery Protocol (SDP)

A Service Discovery Protocol (SDP) segítségével a kliens alkalmazások képes felderíteni, hogy a szerver alkalmazások részérõl milyen szolgáltatások érhetõek el, valamint ezek a szolgáltatások milyen tulajdonságokkal rendelkeznek. A szolgáltatások tulajdonsági közé soroljuk többek között a felajánlott szolgáltatás típusát vagy osztályát, illetve a szolgáltatás kihasználásához szükséges mechanizmusra vagy protokollra vonatkozó információkat.

Az SDP az SDP szerver és az SDP kliens közti kommunikációt foglalja magában. A szerver karbantart egy listát azokról a szolgáltatási rekordokról, amelyek a szerverhez tartozó szolgáltatások jellemzõit írják le. Mindegyik ilyen szolgáltatási rekord egyetlen szolgáltatás adatait tartalmazza. A kliensek egy SDP kéréssel ezeket a szolgáltatási rekordokat kérhetik el az SDP szervertõl. Amennyiben a kliens, vagy a hozzá tartozó alkalmazás a szolgáltatás használata mellett dönt, akkor a szolgáltatás használatához a megfelelõ szolgáltató felé nyitnia kell egy külön kapcsolatot. Az SDP csak a szolgáltatások és azok tulajdonságainak felderítéséhez ad segítséget, de semmilyen eszközt nem tartalmaz a felhasználásukra.

Általában az SDP kliensek általában valamilyen számunkra kellõ tulajdonság alapján keresnek szolgáltatásokat. Ráadásul adódhatnak olyan alkalmak is, amikor a szolgáltatások elõzetes ismerete nélkül szeretnénk felderíteni a rendelkezésre álló szolgáltatások típusait. A felajánlott szolgáltatások ilyen típusú feldolgozását nevezzük *böngészésnek* (browsing).

Az sdpd(8) Bluetooth SDP szerver és a parancssoros sdpcontrol(8) kliens az alap FreeBSD telepítés része. Az alábbi példában egy SDP böngészési kérést adunk ki:

```
% sdpcontrol -a 00:01:03:fc:6e:ec browse
Record Handle: 00000000
Service Class ID List:
        Service Discovery Server (0x1000)
Protocol Descriptor List:
        L2CAP (0x0100)
                Protocol specific parameter #1: u/int/uuid16 1
                Protocol specific parameter #2: u/int/uuid16 1

Record Handle: 0x00000001
Service Class ID List:
        Browse Group Descriptor (0x1001)

Record Handle: 0x00000002
Service Class ID List:
        LAN Access Using PPP (0x1102)
Protocol Descriptor List:
        L2CAP (0x0100)
        RFCOMM (0x0003)
                Protocol specific parameter #1: u/int8/bool 1
Bluetooth Profile Descriptor List:
        LAN Access Using PPP (0x1102) ver. 1.0
```

és így tovább. Mindegyik szolgáltatáshoz hozzátartozik a tulajdonságok egy listája (például RFCOMM csatorna). Lehetséges, hogy szolgáltatástól függõen bizonyos tulajdonságokat kell figyelnünk. Egyes Bluetooth implementációk nem támogatják a szolgáltatások böngészését és ezért egy üres listát adnak vissza. Ebben az esetben egy konkrét szolgáltatásra tudunk rákeresni. A következõ példában az OBEX Object Push (OPUSH) szolgáltatást keressük:

```
% sdpcontrol -a 00:01:03:fc:6e:ec search OPUSH
```

FreeBSD alatt az sdpd(8) szerverrel tudunk szolgáltatásokat felajánlani a Bluetooth klienseknek. A FreeBSD 5.5, 6.1 vagy késõbbi változataiban ehhez a következõ sort kell megadnunk az /etc/rc.conf állományban:

```
sdpd_enable="YES"
```

Ezután az sdpd démon így indítható el:

```
# /etc/rc.d/sdpd start
```

A távoli kliensek részére Bluetooth szolgáltatásokat felajánlani kívánó helyi szerver alkalmazásoknak regisztrálniuk kell magukat a helyi SDP démonnál. Például az egyik ilyen alkalmazás az rfcomm_pppd(8), és elindítása után regisztrálni fogja a Bluetooth LAN szolgáltatást a helyi SDP démonnál.

A helyi SDP szerveren regisztrált szolgáltatásokat a helyi vezérlési csatornán keresztül egy `browse` kéréssel tudjuk lekérdezni:

```
# sdpcontrol -l browse
```

31.4.8. A betárcsázós hálózati és a PPP hálózati hozzáférési (LAN) profilok

A betárcsázós hálózati (Dial-Up Networking, DUN) profil leggyakrabban a modemek és mobiltelefonok között tûnik fel. Ez a profil a következõ forgatókönyveket dolgozza fel:

- A számítógépünkkel egy mobiltelefont vagy modemet vezeték nélküli modemként használunk, amivel az internethez vagy más hálózatokhoz csatlakozunk betárcsázással.
- A számítógépünkkel egy mobiltelefonon vagy modemen keresztül fogadunk adathívásokat.

A PPP hálózati hozzáférési (LAN) profil a következõ helyezetekben alkalmazható:

- LAN hozzáférés egyetlen Bluetooth eszközhöz
- LAN hozzáférés több Bluetooth eszközhöz
- Két gép összekötése (a soros vonali kapcsolat emulációval PPP-n keresztül)

FreeBSD alatt mind a két profilt a ppp(8) és az rfcomm_pppd(8) valósítja meg - egy olyan wrapper eszköz, amely az RFCOMM Bluetooth kapcsolatokat a PPP számára is értelmessé alakítja át. Mielõtt még bármelyik profilt elkezdenénk használni, egy új PPP címkét kell létrehozni az /etc/ppp/ppp.conf állományban. Erre példát az rfcomm_pppd(8) man oldalon találhatunk.

A következõ példában az rfcomm_pppd(8) programot fogjuk használni arra, hogy egy RFCOMM típusú kapcsolatot nyissunk a 00:80:37:29:19:a4 címmel rendelkezõ távoli Bluetooth eszköz felé. A tényleges RFCOMM csatorna számát SDP-n keresztül a távoli eszköztõl kapjuk. Az RFCOMM csatorna kézzel is megadható, és ilyen esetekben az rfcomm_pppd(8) nem fog SDP kérést küldeni. A sdpcontrol(8) használatával tudjuk lekérdezni a távoli eszközön létrejött RFCOMM csatornát.

```
# rfcomm_pppd -a 00:80:37:29:19:a4 -c -C dun -l rfcomm-dialup
```

A PPP hálózati elérés (LAN) szolgáltatás beindításához futni kell a sdpd(8) szervernek. A helyi hálózaton keresztül csatlakozó kliensekhez létre kell hozni egy új bejegyzést az /etc/ppp/ppp.conf állományban. Az rfcomm_pppd(8) man oldalon találhatunk erre példákat. Végezetül indítsuk el az RFCOMM PPP szervert egy érvényes RFCOMM csatornaszámmal. Az RFCOMM PPP szerver ekkor automatikusan regisztrálja a Bluetooth LAN szolgáltatást a helyi SDP démonnál. A következõ

példában megmutatjuk, hogyan lehet elindítani egy RFCOMM PPP szervert:

```
# rfcomm_pppd -s -C 7 -l rfcomm-server
```

31.4.9. Az OBEX Object Push (OPUSH) profil

Az OBEX egy széles körben alkalmazott protokoll a mobileszközök közti egyszerû állományvitelre. Legfõképpen az infravörös kommunikációban alkalmazzák, ahol a laptopok vagy PDA-k közti általános állományátvitelre használják, illetve névjegykártyák vagy naptárbejegyzések átküldésére mobiltelefonok között és egyéb PIM alkalmazást futtató eszközök esetében.

Az OBEX szervert és klienst egy külsõ csomag, az obexapp valósítja meg, amelyet az comms/obexapp portból érhetünk el.

Az OBEX kliens használható objektumok áttolására vagy lehúzására az OBEX szerverhez. Ez az objektum lehet például egy névjegykártya vagy egy megbeszélt találkozó. Az OBEX kliens SDP-n keresztül tud magának RFCOMM csatornaszámot szerezni. Ezt úgy tehetjük meg, ha a szolgáltatás neve helyett egy RFCOMM csatorna számát adjuk meg. A támogatott szolgáltatások: IrMC, FTRN és OPUSH. Számként RFCOMM csatorna is megadható. Az alábbi példában egy OBEX munkamenetet láthatunk, ahol az eszköz információs objektumát húzzuk le a mobiltelefonról és egy új objektumot (egy névjegykártyát) tolunk fel a telefon könyvtárába.

```
% obexapp -a 00:80:37:29:19:a4 -C IrMC
obex> get telecom/devinfo.txt devinfo-t39.txt
Success, response: OK, Success (0x20)
obex> put new.vcf
Success, response: OK, Success (0x20)
obex> di
Success, response: OK, Success (0x20)
```

Az OBEX objektumok tologatásának támogatásához az sdpd(8) szervernek kell futnia. Továbbá a beérkezõ objektumok tárolásához létre kell hoznunk még egy könyvtárat is. Ez az könyvtár alapértelmezés szerint a /var/spool/obex. Végül indítsuk el az OBEX szervert egy érvényes RFCOMM csatorna számának megadásával. Az OBEX szerver ezután automatikusan regisztrálja az "OBEX Object Push" nevû szolgáltatást a helyi SDP démonnál. Ebben a példában láthatjuk az OBEX szerver indítását:

```
# obexapp -s -C 10
```

31.4.10. Soros vonali profil (SPP)

A soros vonali profil (Serial Port Profile, SPP) használatával RS232 (vagy ahhoz hasonló) vonali adatátvitelt tudunk emulálni. Ez a profil a régebben fejlesztett alkalmazásokkal birkózik meg, és a Bluetooth technológiával valódi kábel helyett egy virtuális soros portot képez le.

Az rfcomm_sppd(1) segédprogram ezt a soros vonali profilt valósítja meg. Így egy pszeudo

terminált tudunk virtuális soros portként használni. Ha nem adunk meg RFCOMM csatornát, akkor az rfcomm_sppd(1) képes SDP-n keresztül kérni egyet magának a távoli eszköztõl. Ha ezt felül kívánjuk bírálni, akkor a parancssorban megadhatunk akár egy konkrét RFCOMM csatornát is.

```
# rfcomm_sppd -a 00:07:E0:00:0B:CA -t /dev/ttyp6
rfcomm_sppd[94692]: Starting on /dev/ttyp6...
```

Miután csatlakoztunk, a pszeudo terminált tudjuk soros portként használni:

```
# cu -l ttyp6
```

31.4.11. Hibaelhárítás

Nem tudunk csatlakozni a távoli eszközzel

Egyes Bluetooth eszközök nem támogatják a szerepek cseréjét (role switch). Alapértelmezés szerint amikor a FreeBSD elfogad egy új kapcsolatot, megpróbál rajta szerepet cserélni és mesterré válni. Azok az eszközök, amelyek ezt nem támogatják, nem lesznek képesek emiatt csatlakozni. Ez a szerepváltás az új kapcsolatok felépítése során zajlik le, ezért egy távoli eszköztõl nem lehet megtudni, hogy ismeri-e ezt a lehetõséget. A helyi oldalon a következõ HCI opcióval lehet kikapcsolni a szerepcserét:

```
# hccontrol -n ubt0hci write_node_role_switch 0
```

Valami nem megy. Lehet látni valahogy, pontosan mi is történik?

Persze, igen. Egy külsõ csomag, a hcidump segítségével, amely a comms/hcidump portból érhetõ el. A hcidump segédprogram a tcpdump(1) programhoz hasonlítható. Ezzel lehet a Bluetooth csomagok tartalmát megnézni a terminálon vagy elmenteni ezeket egy állományba.

31.5. Hálózati hidak

31.5.1. Bevezetés

Gyakran hasznos lehet anélkül felosztani egy fizikai hálózatot (például egy Ethernet szegmenst) két külön hálózati szegmensre, hogy külön IP-alhálózatot kellene létrehozunk és összekötnünk ezeket egy útválasztóval. A két ilyen módon kialakított hálózatot összekötõ eszközt nevezzük "hálózati hídnak" (bridge). A legalább két hálózati felülettel rendelkezõ FreeBSD rendszerek képesek hálózati híd szerepét betölteni.

A hálózati híd az eszközök adatkapcsolati rétegben a hozzá tartozó felületein megjelenõ (vagyis Ethernet) címének megtanulásával mûködik. A két hálózat között csak akkor közvetít forgalmat, amikor a forrás és cél nem ugyanabban a hálózatban található.

A hálózati hidak bizonyos szempontból lényegében nagyon kevés porttal rendelkezõ Ethernet

switch-ek.

31.5.2. A hálózati hidak tipikus alkalmazásai

Napjainkban akad néhány igen jellemzõ szituáció, ahol szükség van a hálózati hidak alkalmazására.

Hálózatok összekötése

A hálózati hidak alapvetõ feladata két vagy több hálózati szegmens összekötése. Az egyszerû hálózati környezet felállítása helyett több okból is felmerülhet a hidak létrehozása: kábelezési megszorítások, tûzfalazás vagy pszeudo hálózatok, például virtuális gépek felületének csatlakoztatása miatt. Egy híd használatával ráadásul össze tudunk kötni egy vezeték nélküli hozzáférési pontként üzemelõ felületet egy vezetékes hálózattal.

Szûrés vagy forgalomkorlátozás tûzfallal

Sokszor elõfordulhat, hogy útválasztás vagy hálózati címfordítás (NAT) nélkül szeretnénk tûzfalat használni.

Példaként képzeljünk el egy olyan kis méretû céget, amely egy DSL vagy ISDN vonalon kapcsolódik az internet-szolgáltatójához. A szolgáltatótól 13, mindenki által használható IP-címet kaptak és a hálózatukban 10 gép van. Ebben a helyzetben egy útválasztást végzõ tûzfal mûködtetése nehézkessé válna az alhálózatok problémái miatt.

Egy hídként viselkedõ tûzfallal azonban minden IP számozási probléma nélkül egyszerûen be tudjuk dobni a gépeket a DSL/ISDN útválasztó mögé.

A hálózat megcsapolása

Egy hálózati híddal úgy kapcsolunk össze két hálózati szegmenst, hogy közben meg tudjuk vizsgálni a kettejük között mozgó Ethernet kereteket. Ezt a híd felületen a bpf(4) valamint a tcpdump(1) segítségével tudjuk megoldani, vagy úgy, ha egy másik felületen elküldjük az összes keret másolatát (span, vagyis feszítõ port).

VPN az adatkapcsolati rétegben

A két Ethernet hálózatot egy IP alapú összeköttetésen keresztül is össze tudunk kötni, ha a hálózatokat egy EtherIP járaton keresztül kötjük össze híddal, vagy egy OpenVPN-hez hasonló tap(4) alapú megoldással.

Redundancia az adatkapcsolati rétegben

A hálózatokat több linken keresztül kötjük össze és a redundáns útvonalakat a feszítõfa protokollal (Spanning Tree Protocol, STP). Az Ethernetes hálózatok esetében a megfelelõ mûködéshez a két eszköz között csak egyetlen aktív útvonal létezhet, így a feszítõfa protokoll észleli a hurkokat és a redundáns összeköttetéseket blokkolt állapotba teszi. Amikor azonban az aktív linkek egyike meghibásodik, akkor a protokoll újraszámolja a fát és a hálózati pontjai közti konnektivitást megpróbálja helyreállítani az addig blokkolt linkek ismételt engedélyezésével.

31.5.3. A rendszermag beállításai

Ebben a szakaszban az if_bridge(4) hálózati híd implementációval foglalkozunk, de a Netgraph segítségével is tudunk hidakat építeni. Ez utóbbiról az ng_bridge(4) man oldalon olvashatunk.

Amikor létrehozunk egy hálózati hidat, az ifconfig(8) automatikusan betölti a hozzá tartozó meghajtót. Ha viszont a rendszermag beállításait tartalmazó állományba felvesszük a `device if_bridge` sort, akkor akár be is építhetjük a rendszermagba.

A csomagszûrés minden olyan tûzfallal használható, amely a pfil(9) rendszerre kapcsolódik. Maga a tûzfal is betölthetõ modulként, vagy belefordítható a rendszermagba.

A hálózati híddal forgalmat is tudunk szabályozni az altq(4) vagy a dummynet(4) segítségével.

31.5.4. A hálózati híd engedélyezése

Hálózati hidak felületek klónozásával hozhatóak létre. A híd létrehozásához használjuk az ifconfig(8) programot, és a megfelelõ meghajtó automatikusan betöltõdik, ha nem lenne még elérhetõ a rendszermagban.

```
# ifconfig bridge create
bridge0
# ifconfig bridge0
bridge0: flags=8802<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 96:3d:4b:f1:79:7a
        id 00:00:00:00:00:00 priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 100 timeout 1200
        root id 00:00:00:00:00:00 priority 0 ifcost 0 port 0
```

Ekkor létrejön a hálózati hídhoz tartozó felület és véletlenszerûen generálódik hozzá egy Ethernetes cím. A `maxaddr` és a `timeout` paraméterek vezérlik, hogy a híd mennyi MAC-címet tartson meg a keretek továbbításáért felelõs táblázatban és mennyi másodperc után töröljön automatikusan egy bejegyzést a legutolsó használat után. A többi paraméter a feszítõfa mûködését irányítja.

Vegyük fel a hídhoz tartozó hálózati tagfelületeket. A híd csak akkor fog a tagfelületek között csomagokat továbbküldeni, amikor a híd és a tagok is `up` állapotban vannak:

```
# ifconfig bridge0 addm fxp0 addm fxp1 up
# ifconfig fxp0 up
# ifconfig fxp1 up
```

A híd most már átküldi az Ethernet kereteket a fxp0 és fxp1 felületek között. Az iméntiekkel megegyezõ konfigurációt az /etc/rc.conf állományban így alakíthatjuk ki:

```
cloned_interfaces="bridge0"
ifconfig_bridge0="addm fxp0 addm fxp1 up"
ifconfig_fxp0="up"
```

```
ifconfig_fxp1="up"
```

Ha a hídhoz IP-címet is rendelni akarunk, akkor inkább magánál a hídnál adjuk meg, ne a tagoknál. Ezt statikusan vagy DHCP használatával is megtehetjük:

```
# ifconfig bridge0 inet 192.168.0.1/24
```

A hídhoz IPv6 címet is hozzá tudunk rendelni.

31.5.5. Tûzfalazás

Ha engedélyezzük a csomagszûrést, a hídon áthaladó csomagok elõször a küldõ felület érkezési oldalára kerülnek, majd a hídra, végül a megfelelõ irányban levõ felület küldési oldalára. Bármelyik fázis letiltható. Amikor a csomagok áramlásának iránya fontos számunkra, akkor jobban járunk, ha nem magára a hídra, hanem csak a tagfelületekre állítjuk be a tûzfalat.

A híd számos módosítható beállítással rendelkezik a nem-IP és ARP csomagok átküldésére, valamint arra, hogy az IPFW tûzfal adatkapcsolati réteg szintjén mûködhessen. Az if_bridge(4) man oldal ennek részleteit tárja fel.

31.5.6. Feszítõfák

A híd meghajtója a gyors feszítõfa protokollt (Rapid Spanning Tree Protocol, RSTP avagy 802.1w) valósítja meg, ami visszafelé kompatibilis a korábban említett feszítõfa protokollal. A feszítõfákat a hálózati topológiában felbukkanó hurkok észlelésére és eltávolítására alkalmazzák. Az RSTP azonban a hagyományos STP-nél valamivel gyorsabb konvergenciát ígér, mivel itt a szomszédos switch-ek kicserélik egymás között az adataikat, és így újabb hurkok létrehozása nélkül képesek viszonylag gyorsan egyik állapotból átváltani a másikba.

Az alábbi táblázat a támogatott mûködési módokat láthatjuk:

Operációs rendszer	STP módok	Alapértelmezés
FreeBSD 5.4-FreeBSD 6.2	STP	STP
FreeBSD 6.3+	RSTP vagy STP	STP
FreeBSD 7.0+	RSTP vagy STP	RSTP

A tagfelületeken az `stp` paranccsal tudjuk engedélyezni a feszítõfák használatát. Az fxp0 és fxp1 felületeket összekötõ hídfelület esetében tehát így:

```
# ifconfig bridge0 stp fxp0 stp fxp1
bridge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether d6:cf:d5:a0:94:6d
        id 00:01:02:4b:d4:50 priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 100 timeout 1200
        root id 00:01:02:4b:d4:50 priority 32768 ifcost 0 port 0
        member: fxp0 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
```

```
                port 3 priority 128 path cost 200000 proto rstp
                role designated state forwarding
        member: fxp1 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                port 4 priority 128 path cost 200000 proto rstp
                role designated state forwarding
```

Láthatjuk, hogy a híd a feszítõfában megkapta a `00:01:02:4b:d4:50`-es azonosítót és a `32768`-as prioritást. Mivel `root id` értéke is ugyanez, elmondhatjuk, hogy ez a fa gyökereként funkcionáló híd.

Ha a hálózaton már valahol létezik egy másik híd:

```
bridge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 96:3d:4b:f1:79:7a
        id 00:13:d4:9a:06:7a priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 100 timeout 1200
        root id 00:01:02:4b:d4:50 priority 32768 ifcost 400000 port 4
        member: fxp0 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                port 4 priority 128 path cost 200000 proto rstp
                role root state forwarding
        member: fxp1 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                port 5 priority 128 path cost 200000 proto rstp
                role designated state forwarding
```

A `root id 00:01:02:4b:d4:50 priority 32768 ifcost 400000 port 4` sor mutatja, hogy a fa gyökerét képezõ híd most a `00:01:02:4b:d4:50` azonosítóval rendelkezik, és ezt a hidat `400000`-res költséggel éri el a `port 4` (a 4. porton) keresztül, amely jelen esetben az fxp0 felület.

31.5.7. Komolyabb hidak építése

A forgalom áramlásának átszerkesztése

A hidak támogatják az ún. megfigyelési módot, ahol a csomagokat a bpf(4) feldolgozásuk után eldobja, így nem folytatódik a feldolgozásuk vagy nem haladnak tovább. Ennek kihasználásával a két vagy több felületen érkezõ adatokat egyetlen bpf(4) folyammá tudjuk alakítani. Ez olyan hálózati csapok forgalmának átszerkesztésében hasznos, ahol a két különbözõ felületen keresztül küldjük ki az RX/TX (fogadás/küldés) jeleket.

Az alábbi paranccsal tudjuk megoldani, hogy négy felületrõl érkezõ adatot legyünk képesek egyetlen folyamként olvasni:

```
# ifconfig bridge0 addm fxp0 addm fxp1 addm fxp2 addm fxp3 monitor up
# tcpdump -i bridge0
```

Feszítõ portok

A hídhoz befutó Ethernet keretek mindegyikérõl készül egy másolat, ami egy megadott feszítõ porton keresztül megy tovább. Hidanként végtelen számú ilyen feszítõ port létezhet, és ha egy

felületet feszítõ portnak adtunk meg, akkor hagyományos portként már nem használhatjuk. Ez leginkább akkor hasznos, amikor passzívan akarjuk megfigyelni a híddal rendelkezõ hálózatot a híd valamelyik feszítõ portjára csatlakozó géprõl.

Küldessük az összes keretrõl egy másolatot az fxp4 felületre:

```
# ifconfig bridge0 span fxp4
```

Privát felületek

A privát felületek (private interface) csak más privát felületek felé küldenek tovább adatot. Így feltétel nélkül tudjuk korlátozni a forgalmat, és sem Ethernet keretek, sem pedig ARP nem megy keresztül rajtuk. Ha viszont szelektíven akarjuk korlátozni a forgalmat, akkor helyette használjunk tûzfalat.

Tapadós felületek

Ha a híd egyik tagfelületét tapadósnak (sticky) adjuk meg, akkor a dinamikusan megtanult címek bejegyzései a gyorsítótárba kerülésük után állandósulnak. A tapadós bejegyzések soha nem évülnek el vagy cserélõdnek le, még abban az esetben sem, ha utána az adott címet egy másik felületrõl látjuk. Így a továbbításra vonatkozó táblázatot nem kell elõre feltöltenünk, és a híd egyik oldalán meglátott kliensek nem képesek átvándorolni egy másik hálózati szegmensbe.

Másik ilyen példa a tapadós címek használatára az lehetne, amikor a hidat VLAN-nal kombináljuk, és így egy olyan útválasztót hozunk létre, ahol az ügyfeleink az IP-címtartomány pocséklása nélkül zárhatóak el egymástól. Tegyük fel, hogy az `A-ugyfel` a `vlan100`, és a `B-ugyfel` a `vlan101` felületen csatlakozik. A híd IP-címe `192.168.0.1`, amely maga is egy internet felé mutató útválasztó.

```
# ifconfig bridge0 addm vlan100 sticky vlan100 addm vlan101 sticky vlan101
# ifconfig bridge0 inet 192.168.0.1/24
```

Mind a két kliens a `192.168.0.1` címet látja alapértelmezett átjáróként, és mivel a híd gyorsítótára tapadós bejegyzéseket tartalmaz, a MAC-címeik meghamisításával nem tudják elcsípni a másikuk forgalmát.

A VLAN-ok közti bárminemû kommunikációt privát felületek létrehozásával akadályozzuk meg (vagy egy tûzfallal):

```
# ifconfig bridge0 private vlan100 private vlan101
```

Ezzel a megoldással az ügyfeleinket teljesen elszigeteljük egymástól úgy, hogy közben az egész `/24` címtartomány külön alhálózatok kialakítása nélkül kiosztható.

Címek korlátozása

Korlátozhatóak az egy felület mögül küldeni képes egyedi MAC-címek. Amikor ezen a határon felül érkeznek ismeretlen feladótól csomagok, egészen addig eldobjuk ezeket, amíg egy korábban már

regisztrált bejegyzést a rendszer ki nem töröl vagy ki nem veszünk a gyorsítótárból.

A következõ példában az `vlan100` felületen csatlakozó `A-ugyfel` számára korlátozzuk le 10-re az Ethernet eszközök számát:

```
# ifconfig bridge0 ifmaxaddr vlan100 10
```

SNMP felügyelet

A hidak és az STP paraméterei az alap FreeBSD rendszerben megtalálható SNMP démonnal felügyelhetõek. A hídhoz exportált felügyeleti információk (Management Information Base, MIB) megfelelnek az IETF által elõírt szabványoknak, így akár tetszõleges SNMP kliens vagy bármilyen más felügyeleti szoftver alkalmas az olvasásukra.

A hidat mûködtetõ gépen az /etc/snmp.config állományban engedélyezzük a `begemotSnmpdModulePath."bridge" = "/usr/lib/snmp_bridge.so"` sort és indítsuk el a bsnmpd démont. Itt még szükség lehet más beállítások, például a közösségek nevének (community name) vagy a hozzáférési listák (access list) módosítására is. Ezzel kapcsolatban a bsnmpd(1) és az snmp_bridge(3) man oldalakat lapozzuk fel.

A következõ példában a Net-SNMP nevû szoftver (net-mgmt/net-snmp) fogjuk használni a híd elérésére, de ugyanerre a net-mgmt/bsnmptools port is alkalmas. Az SNMP klienst használó gépen egészítsük ki az $HOME/.snmp/snmp.conf állományt a híd felügyeleti információinak importálásával az Net-SNMP rendszerébe:

```
mibdirs +/usr/shared/snmp/mibs
mibs +BRIDGE-MIB:RSTP-MIB:BEGEMOT-MIB:BEGEMOT-BRIDGE-MIB
```

Az IETF BRIDGE-MIB (RFC 4188) használatán keresztül így tudjuk elindítani egy híd felügyeletét:

```
% snmpwalk -v 2c -c public bridge1.example.com mib-2.dot1dBridge
BRIDGE-MIB::dot1dBaseBridgeAddress.0 = STRING: 66:fb:9b:6e:5c:44
BRIDGE-MIB::dot1dBaseNumPorts.0 = INTEGER: 1 ports
BRIDGE-MIB::dot1dStpTimeSinceTopologyChange.0 = Timeticks: (189959) 0:31:39.59 centi-
seconds
BRIDGE-MIB::dot1dStpTopChanges.0 = Counter32: 2
BRIDGE-MIB::dot1dStpDesignatedRoot.0 = Hex-STRING: 80 00 00 01 02 4B D4 50
...
BRIDGE-MIB::dot1dStpPortState.3 = INTEGER: forwarding(5)
BRIDGE-MIB::dot1dStpPortEnable.3 = INTEGER: enabled(1)
BRIDGE-MIB::dot1dStpPortPathCost.3 = INTEGER: 200000
BRIDGE-MIB::dot1dStpPortDesignatedRoot.3 = Hex-STRING: 80 00 00 01 02 4B D4 50
BRIDGE-MIB::dot1dStpPortDesignatedCost.3 = INTEGER: 0
BRIDGE-MIB::dot1dStpPortDesignatedBridge.3 = Hex-STRING: 80 00 00 01 02 4B D4 50
BRIDGE-MIB::dot1dStpPortDesignatedPort.3 = Hex-STRING: 03 80
BRIDGE-MIB::dot1dStpPortForwardTransitions.3 = Counter32: 1
RSTP-MIB::dot1dStpVersion.0 = INTEGER: rstp(2)
```

A példában látszik, hogy a dot1dStpTopChanges.0 értéke kettõ, ami arra utal, hogy az STP híd topológiája kétszer változott. A topológia változása pedig azt jelenti, hogy a hálózaton belül egy vagy több link állapota megváltozott vagy egyszerûen meghibásodott és ezért egy új fát kellett számolni. A dot1dStpTimeSinceTopologyChange.0 érték adja meg, hogy ez pontosan mikor is történt.

Több híd felületének felügyeletéhez a belsõ BEGEMOT-BRIDGE-MIB parancsot is használhatjuk:

```
% snmpwalk -v 2c -c public bridge1.example.com
enterprises.fokus.begemot.begemotBridge
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseName."bridge0" = STRING: bridge0
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseName."bridge2" = STRING: bridge2
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseAddress."bridge0" = STRING: e:ce:3b:5a:9e:13
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseAddress."bridge2" = STRING: 12:5e:4d:74:d:fc
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseNumPorts."bridge0" = INTEGER: 1
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseNumPorts."bridge2" = INTEGER: 1
...
BEGEMOT-BRIDGE-MIB::begemotBridgeStpTimeSinceTopologyChange."bridge0" = Timeticks:
(116927) 0:19:29.27 centi-seconds
BEGEMOT-BRIDGE-MIB::begemotBridgeStpTimeSinceTopologyChange."bridge2" = Timeticks:
(82773) 0:13:47.73 centi-seconds
BEGEMOT-BRIDGE-MIB::begemotBridgeStpTopChanges."bridge0" = Counter32: 1
BEGEMOT-BRIDGE-MIB::begemotBridgeStpTopChanges."bridge2" = Counter32: 1
BEGEMOT-BRIDGE-MIB::begemotBridgeStpDesignatedRoot."bridge0" = Hex-STRING: 80 00 00 40
95 30 5E 31
BEGEMOT-BRIDGE-MIB::begemotBridgeStpDesignatedRoot."bridge2" = Hex-STRING: 80 00 00 50
8B B8 C6 A9
```

Így tudjuk megadni, hogy a hidat mib-2.dot1dBridge részfán keresztül akarjuk megfigyelni:

```
% snmpset -v 2c -c private bridge1.example.com
BEGEMOT-BRIDGE-MIB::begemotBridgeDefaultBridgeIf.0 s bridge2
```

31.6. Linkek összefûzése és hibatûrése

31.6.1. Bevezetés

A lagg(4) felület lehetõvé teszi, hogy több hálózati felületet egyetlen virtuális felületként fûzzünk össze, és ezzel egy hibatûrõ és nagysebességû összeköttetést alakítsunk ki.

31.6.2. Mûködési módok

failover

Csak az elsõdlegesként kijelölt porton keresztül fogad és küld adatokat. Amikor ez az elsõdleges port elérhetetlenné válik, a következõ aktív portot fogja használni. Az elsõként felvett felület válik automatikusan az elsõdleges porttá, és az utána felvett összes többit pedig csak hiba esetén használjuk.

Cisco® Fast EtherChannel®

A Cisco® Fast EtherChannel® (FEC) technológia támogatása. Ez egy statikus beállítás, és nem egyezteti az összefûzést a többiekkel vagy a linkek felügyeletéhez nem vált kereteket. Ha a switch támogatja az LACP használatát, akkor inkább azt válasszuk.

A FEC a kimenõ forgalmat a fejlécekben szereplõ protokollok alapján számolt hasítókóddal próbálja szétosztani az aktív portok között, és tetszõleges aktív porton fogad beérkezõ adatokat. Az említett hasítókódban egy Ethernetes forrás- és célcím szerepel, valamint ha elérhetõ, akkor egy VLAN címke, illetve az IPv4/IPv6 forrás- és célcím.

LACP

Az IEEE® 802.3ad Link Aggregation Control Protocol (LACP) és a Marker Protcol támogatása. Az LACP megpróbálja egyeztetni a többi géppel az összefûzhetõ linkeket egy vagy több csoportban (Link Aggregated Group, LAG). Mindegyik ilyen csoportban ugyanolyan sebességû portokat találunk, full-duplex mûködési módban. A forgalmat így a legnagyobb összsebességgel rendelkezõ csoportban megtalálható portok között osztja el, ami a legtöbb esetben az összes portot magában foglaló csoport. A fizikai konnektivitás megváltozása esetén a linkek összefûzõdése igen gyorsan alkalmazkodik az új konfigurációhoz.

Az LACP a kimenõ forgalmat az aktív portok között osztja szét fejlécekben szereplõ protokollok alapján számolt hasítókóddal, és bármelyik aktív portról fogad bejövõ forgalmat. A hasítókódban megtalálható az Ethernetes forrás- és célcím, valamint ha elérhetõ, akkor a VLAN címke, illetve az IPv4/IPv6 forrás- és célcímek.

Loadbalance

Ez a *FEC* mód másik neve.

Round-Robin

A kimenõ forgalmat egy körkörös (Round-Robin) elvû ütemezõvel osztja szét az aktív portok között és tetszõleges aktív portról fogad bejövõ forgalmat. Ez a mûködési mód megsérti az Ethernet keretek rendezését és csak nagy körültekintés mellett alkalmazzuk.

31.6.3. Példák

Példa 37. LACP alapú összefûzés egy Cisco® switch-csel

Ebben a példában egy FreeBSD-s gép két felületét kapcsoljuk össze switch-csel egy egyszerû terhelés-kiegyenlítéssel és hibatûréssel beállított linken keresztül. Mivel az Ethernet keretek sorrendje döntõ fontosságú, ezért a két állomás között egyazon fizikai linken zajló forgalom maximális sebességét az adott felület kapacitása korlátozza. A küldési algoritmus a lehetõ legtöbb információ alapján próbálja egymástól megkülönböztetni a forgalmakat és elosztani ezeket a rendelkezésre álló felületek között.

A Cisco® switch-en vegyünk fel a *FastEthernet0/1* és *FastEthernet0/2* interfészeket az *1* csoportba (channel group):

```
interface FastEthernet0/1
 channel-group 1 mode active
```

```
 channel-protocol lacp
!
interface FastEthernet0/2
 channel-group 1 mode active
 channel-protocol lacp
```

A FreeBSD-s gépen pedig a *fxp0* és *fxp1* használatával hozzunk létre a lagg(4) interfészt:

```
# ifconfig lagg0 create
# ifconfig lagg0 up laggproto lacp laggport fxp0 laggport fxp1
```

Ellenõrizzük a felület állapotát:

```
# ifconfig lagg0
```

A *ACTIVE* jelzésû, vagyis aktív állapotú portok az összefûzéshez kialakított csoport azon tagjai, amelyeknél felépült a kapcsolat a távoli switch felé és készen állnak a küldésre és fogadásra. Ha az ifconfig(8) programtól részletesebb kimenetet kérünk, akkor láthatjuk a csoportok azonosítóit is:

```
lagg0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=8<VLAN_MTU>
        ether 00:05:5d:71:8d:b8
        media: Ethernet autoselect
        status: active
        laggproto lacp
        laggport: fxp1 flags=1c<ACTIVE,COLLECTING,DISTRIBUTING>
        laggport: fxp0 flags=1c<ACTIVE,COLLECTING,DISTRIBUTING>
```

A `show lacp neighbor` paranccsal kérdezhetjük le a portok állapotát:

```
switch# show lacp neighbor
Flags:  S - Device is requesting Slow LACPDUs
        F - Device is requesting Fast LACPDUs
        A - Device is in Active mode       P - Device is in Passive mode

Channel group 1 neighbors

Partner's information:

                  LACP port                        Oper    Port     Port
Port      Flags   Priority  Dev ID         Age     Key     Number   State
Fa0/1     SA      32768     0005.5d71.8db8  29s    0x146   0x3      0x3D
Fa0/2     SA      32768     0005.5d71.8db8  29s    0x146   0x4      0x3D
```

Részletesebb kijelzést a `show lacp neighbor detail` paranccsal kaphatunk.

Példa 38. A hibatûrés beállítása

A hibatûrési mód arra alkalmas, hogy amikor az elsõdleges porton elvesztjük a kapcsolatot, helyette egy másodlagos interfész használatára tudunk áttérni. Hozzuk létre és állítsuk be a *lagg0* interfészt, ahol az *fxp0* legyen a fõinterfész, az *fxp1* pedig a tartalék interfész:

```
# ifconfig lagg0 create
# ifconfig lagg0 up laggproto failover laggport fxp0 laggport fxp1
```

Az így létrejövõ interfész nagyjából az alábbi lesz, ahol eltérés a MAC-cím és az eszköz neve:

```
# ifconfig lagg0
lagg0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=8<VLAN_MTU>
        ether 00:05:5d:71:8d:b8
        media: Ethernet autoselect
        status: active
        laggproto failover
        laggport: fxp1 flags=0<>
        laggport: fxp0 flags=5<MASTER,ACTIVE>
```

A forgalom kezdetben az *fxp0* felületen keresztül érkezik és távozik. Ha az *fxp0* felületen valamiért megszakadna a kapcsolat, helyette az *fxp1* lesz az aktív link. Ha késõbb helyreáll a kapcsolat az elsõdleges felületen, akkor újra az lesz aktív link.

Példa 39. Hibatûrés beállítása vezetékes és vezeték nélküli hálózatok között

Hordozható számítógépek használata esetén általában érdemesebb a vezeték nélküli kapcsolatot másodlagos interfészként beállítani, így csak akkor használja a rendszer, ha vezetékes hálózat nem érhetõ el. A lagg(4) segítségével egyetlen IP-címmel tudjuk használni mind a két interfészt: a teljesítmény és biztonságosság miatt elsõsorban a vezetékes hálózatot használjuk, miközben megmarad a lehetõség az adatok továbbítására a vezeték nélküli kapcsolaton keresztül is.

A beállítás során a vezeték nélküli interfész MAC-címét úgy kell módosítanunk, hogy megegyezzen a lagg(4) címével. A lagg(4) interfész a saját MAC-címét az elsõdleges interfésztõl örökli, amely jelen esetünkben a vezetékes interfész lesz.

A most következõ példában a vezetékes hálózatunk lesz az elsõdleges interfész (*bge0*), míg a vezeték nélküli (*wlan0*) a másodlagos. A *wlan0* interfészt az *iwn0* interfészbõl hoztuk létre, és a vezetékes kapcsolat MAC-címét állítjuk be neki. Elsõ lépésként tehát le kell kérdeznünk a vezetékes interfész MAC-címét:

```
# ifconfig bge0
```

```
bge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=19b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,TSO4>
    ether 00:21:70:da:ae:37
    inet6 fe80::221:70ff:feda:ae37%bge0 prefixlen 64 scopeid 0x2
    nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
    media: Ethernet autoselect (1000baseT <full-duplex>)
    status: active
```

A *bge0* helyett természetesen a saját vezetékes hálózati interfészünket kell megadni, és az `ether` kezdetû sorban is saját kártyánk MAC-címe fog megjelenni. Ezután már meg is tudjuk változtatni az *iwn0* címét:

```
# ifconfig iwn0 ether 00:21:70:da:ae:37
```

Aktiváljuk a vezeték nélküli interfészt, de ne állítsunk be neki semmilyen IP-címet:

```
# ifconfig wlan0 create wlandev iwn0 ssid wlan_hálózat up
```

Hozzuk létre a lagg(4) interfészt a *bge0* mint elsõdleges interfész megadásával, valamint a *wlan0* legyen a szükség esetén használható tartalék:

```
# ifconfig lagg0 create
# ifconfig lagg0 up laggproto failover laggport bge0 laggport wlan0
```

Az így létrehozott interfész nagyjából így fog megjelenni, egyedüli fontosabb eltérések a MAC-címek és az eszközök nevei:

```
# ifconfig lagg0
lagg0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=8<VLAN_MTU>
        ether 00:21:70:da:ae:37
        media: Ethernet autoselect
        status: active
        laggproto failover
        laggport: wlan0 flags=0<>
        laggport: bge0 flags=5<MASTER,ACTIVE>
```

Hogy ne kelljen a rendszer minden egyes indítása után ezt a mûveletet megismételni, vegyük fel a következõ sorokat az /etc/rc.conf állományba:

```
ifconfig_bge0="up"
ifconfig_iwn0="ether 00:21:70:da:ae:37"
wlans_iwn0="wlan0"
ifconfig_wlan0="WPA"
cloned_interfaces="lagg0"
```

```
ifconfig_lagg0="laggproto failover laggport bge0 laggport wlan0 DHCP"
```

31.7. Lemez nélküli mûködés

A FreeBSD képes hálózaton keresztül elindulni és helyi lemez nélkül egy NFS szerver által megosztott állományrendszer csatlakoztatásával mûködni. Ehhez a szabványos konfigurációs állományok módosításán kívül semmi másra nincs szükségünk. Egy ilyen rendszert viszonylag könnyû beállítani, mivel az összes hozzávaló szinte készen elérhetõ:

- Rögtön adott legalább két módszer, ha a rendszermagot hálózaton keresztül akarjuk betölteni:
  - PXE: az Intel® által fejlesztett Preboot eXecution Environment ("indítás elõtti végrehajtási környezet") nevû rendszer a hálózati kártyákba vagy alaplapokba épített ROM segítségével teszi lehetõvé az intelligens rendszerindítást. A pxeboot(8) man oldalán olvashatunk errõl részletesebben.
  - Az Etherboot port (net/etherboot) olyan ROM-ba programozható kódot készít, amellyel rendszermagokat tudunk hálózaton keresztül betölteni. Ez a kód egyaránt felhasználható egy hálózati rendszerindító PROM beégetéséhez, vagy betölthetõ a helyi floppy (esetleg merev)lemezrõl, illetve MS-DOS® rendszer alól. Elég sok hálózati kártya támogatja ezt a módot.
- Egy mintaszkript (/usr/shared/examples/diskless/clone_root) is próbálja megkönnyíteni a szerveren a munkaállomás rendszerindító állományrendszerének létrehozását és karbantartását. Ezt a szkriptet valószínûleg némileg módosítani kell, de így is sokat segít az elindulásban.
- Az /etc könyvtárban található szabványos rendszerindításhoz használt állományok, amelyekkel a lemez nélküli indulást lehet detektálni és segíteni.
- A lapozás, amennyiben szükséges, NFS vagy helyi lemez segítségével oldható meg.

Számos módon állíthatunk be egy lemez nélküli munkaállomást. Rengeteg részbõl tevõdik össze, és ezek legtöbbje remekül testreszabható az igényeinknek. A továbbiakban egy teljes rendszer összeállításának lehetséges variációit ismertetjük, különös hangsúlyt fektetünk arra, hogy egyszerûek és a hagyományos FreeBSD indítószkriptekkel kompatibilisek maradjanak. A bemutatandó rendszer a következõ jellemzõkkel bír:

- A lemez nélküli munkaállomások megosztott / és /usr állományrendszereket használnak.

  A rendszer indításához használt gyökér állományrendszer a szabvány FreeBSD-s gyökér (ez általában a szerveré), ahol néhány állományt felülírtunk a lemez nélküli mûködéshez vagy azért, mert egyszerûen az adott munkaállomáshoz tartozik.

  A gyökér azon részeit, amelyeket írhatóvá kívánunk tenni, md(4) alapú állományrendszerekkel lapoljuk felül. Ilyenkor azonban bármilyen rajtuk ejtett változtatás a rendszer újraindításával elveszik.

- A rendszermagot vagy az Etherboot vagy a PXE használatával küldessük át és töltsük be, mivel egyes helyzetekben ezekre szükség lesz.

A bemutatott rendszer nem biztonságos. Helyezzük a hálózatunk egy jól védett részére, és a többi gép ne tekintse megbízhatónak.

A szakaszban szereplõ összes információt a FreeBSD 5.2.1-RELEASE változatával teszteltük.

31.7.1. Háttérinformációk

A lemez nélküli munkaállomások beállítása egyszerre adja magát és könnyen is elvéthetõ. Az elkövetett hibákat olykor számos okból kifolyólag nehéz felismerni. Például:

- A fordítási idõben megadott beállítások mást eredményeznek futási idõben.
- A hibaüzenetek gyakran titokzatosak vagy esetleg teljesen el is maradnak.

Ezért ha valamennyire tisztában vagyunk a háttérben zajló folyamatokkal, akkor sokkal több eséllyel leszünk képesek megoldani a menet közben felmerülõ problémákat.

A rendszernek a sikeres felkapaszkodáshoz több mûveletet is végre kell hajtania:

- A gépnek szüksége van olyan induló paraméterekhez, mint például az IP-cím, a végrehajtható állomány neve, a szerver neve, a gyökér elérési útja. Ezeket a DHCP vagy a BOOTP protokollok használatával adhatjuk meg. A DHCP a BOOTP kompatibilis kiterjesztése, ezért ugyanazokat a portokat és alapvetõ csomagformátumot alkalmazza.

  A rendszerüket kizárólag BOOTP használatával is beállíthatjuk. A bootpd(8) szerver az alap FreeBSD rendszer része.

  A DHCP azonban rengeteg elõnnyel rendelkezik a BOOTP protokollal szemben (áttekinthetõbb konfigurációs állományok, a PXE használatának lehetõsége, illetve sok minden más, ami nem csak a lemez nélküli mûködéshez kellhet), ezért itt alapvetõen egy DHCP alapú konfigurációt mutatunk be, de ahol megoldható, megemlítjük a bootpd(8) esetén alkalmas példákat is. A mintaként szolgáló konfiguráció az ISC DHCP szoftvercsomagot használja (a tesztszerverre ennek a 3.0.1.r12 verzióját telepítetük fel).

- A gépnek egy vagy több programot kell a saját memóriájába áttöltenie. Erre vagy a TFTP vagy pedig az NFS alkalmas. A TFTP és az NFS között sok helyen fordítási idõben tudunk választani. Gyakori hibaforrás a protokollhoz rosszul megadott állománynevek használata: a TFTP általában az összes állományt a szerverrõl egyetlen könyvtárból tölti át, ezért arra számít, hogy a neveiket ehhez viszonyítva adjuk meg. Az NFS használata során azonban abszolút elérési utakat kell megadnunk.
- A rendszer indítását lehetõvé tevõ közbensõ programokat és a rendszermagot valahogy inicializálni kell és elindítani. Ezen a területen több fontos változat kapott helyet:
  - A PXE a pxeboot(8) kódját fogja betölteni, ez lényegében a FreeBSD betöltõ harmadik fokozatának egy módosított változata. A loader(8) a mûködéséhez szükséges paramétereket a rendszer indításakor kapja meg, majd a vezérlés átadása elõtt ezeket a rendszermag környezetében hagyja. Ebben az esetben akár a GENERIC rendszermag is használható.
  - Az Etherboot kevesebb elõkészítéssel közvetlenül magát a rendszermagot tölti be. Ehhez azonban egy saját rendszermagot kell építeni, külön beállításokkal.

A PXE és az Etherboot egyaránt jól használható. Mivel azonban a rendszermagok általában a loader(8) kódjára hagyják a munka legnagyobb részét, ezért ahol lehetséges, a PXE megoldását érdemes alkalmazni.

Tehát ha az alaplapi BIOS és a hálózati kártya is támogatja a PXE használatát, akkor válasszunk inkább azt.

- Végezetül a gépnek valamilyen módon hozzá kell tudnia férnie az állományrendszerekhez. Erre többnyire az NFS jöhet szóba.

A további részleket lásd a diskless(8) man oldalon.

31.7.2. Beállítási útmutató

Beállítás a ISC DHCP használatával

Az ISC DHCP szervere képes a BOOTP és DHCP kéréseket is megválaszolni.

Az ISC DHCP 3.0 nem az alaprendszer része, ezért a használatához elõször telepítenünk kell a net/isc-dhcp30-server portot vagy a neki megfelelõ csomagot.

Ahogy feltelepítettük, le kell futtatnunk az ISC DHCP konfigurációs állományát (ezt általában /usr/local/etc/dhcpd.conf néven találjuk meg). A most következõ, megjegyzésekkel kiegészített példában egy `margaux` nevû gép az Etherboot, valamint egy `corbieres` nevû gép PXE használatával akar kapcsolódni:

```
default-lease-time 600;
max-lease-time 7200;
authoritative;

option domain-name "minta.com";
option domain-name-servers 192.168.4.1;
option routers 192.168.4.1;

subnet 192.168.4.0 netmask 255.255.255.0 {
  use-host-decl-names on; ①
  option subnet-mask 255.255.255.0;
  option broadcast-address 192.168.4.255;

  host margaux {
    hardware ethernet 01:23:45:67:89:ab;
    fixed-address margaux.minta.com;
    next-server 192.168.4.4; ②
    filename "/data/misc/kernel.diskless"; ③
    option root-path "192.168.4.4:/data/misc/diskless"; ④
  }
  host corbieres {
    hardware ethernet 00:02:b3:27:62:df;
    fixed-address corbieres.minta.com;
    next-server 192.168.4.4;
```

```
    filename "pxeboot";
    option root-path "192.168.4.4:/data/misc/diskless";
  }
}
```

① Ez a beállítás arra utasítja a dhcpd démont, hogy a lemez nélküli gép hálózati neveként a `host` deklarációban megadott értéket küldje el. Ezt úgyis meg lehet csinálni, hogy felvesszünk egy `option host-name margaux` részt a `host` deklarációk közé.

② A `next-server` direktíva a betöltõ vagy a rendszermag betöltéséért felelõs TFTP vagy NFS szervert jelöli ki (alapértelmezés szerint ez megegyezik a DHCP szerverrel).

③ A `filename` direktíva azt az állományt adja meg, amelyet az Etherboot vagy a PXE a következõ végrehajtási lépésben betölt. Ezt a kiválasztott átviteli módnak megfelelõen kell megadni. Az Etherboot lefordítható az NFS vagy a TFTP használatával is. A FreeBSD port alapból az NFS támogatását tartalmazza. A PXE a TFTP protokollt használja, ezért itt relatív állományneveket adunk meg (ez persze a TFTP szerver beállításaitól függ, de általában ez a jellemzõ). Sõt, a PXE a pxeboot állományt tölti be, nem is a rendszermagot. Léteznek további érdekes lehetõségek is, mint például a pxeboot állomány betöltése a FreeBSD CD-jén található /boot könyvtárból (mivel a pxeboot(8) a GENERIC rendszermagot képes betölteni, ezért a PXE használatával akár egy távoli CD-meghajtóról is indíthatjuk a rendszert).

④ A `root-path` opció a rendszer indításához használt gyökér állományrendszert nevezi meg, amelyet többnyire az NFS jelölési módszere szerint kell megadni. A PXE használata során el lehet hagyni a gép IP-címét egészen addig, amíg nem engedélyezzük a rendszermagban a BOOTP beállítást. Az NFS szerver ekkor megegyzik a TFTP szerverrel.

Beállítás a BOOTP használatával

Itt a bootpd (egyetlen kliensre korlátozott) beállítását láthatjuk. Ezt az /etc/bootptab állományba tegyük.

Ne feledjük, hogy a BOOTP használatához az Etherboot portot a `NO_DHCP_SUPPORT` beállítással kell fordítanunk, miközben a PXE esetében kell a DHCP. Egyébként a bootpd egyedüli nyilvánvaló elõnye csupán annyi, hogy az alaprendszer része.

```
.def100:\
  :hn:ht=1:sa=192.168.4.4:vm=rfc1048:\
  :sm=255.255.255.0:\
  :ds=192.168.4.1:\
  :gw=192.168.4.1:\
  :hd="/tftpboot":\
  :bf="/kernel.diskless":\
  :rp="192.168.4.4:/data/misc/diskless":

margaux:ha=0123456789ab:tc=.def100
```

A rendszer elõkészítése az Etherboot számára

Az Etherboot honlapján találhatunk egy minden részletre kiterjedõ dokumentációt (angolul),

amely elsõsorban ugyan a Linux típusú rendszerek számára íródott, de ettõl függetlenül még hasznos információkat tartalmaz. A továbbiakban csak annyit szeretnénk körvonalazni, hogy az Etherboot miként bírható mûködésre FreeBSD rendszerekkel.

Elõször telepítenünk kell a net/etherboot csomagot vagy portot.

Az Etherboot beállítását (vagyis a TFTP használatának megadását az NFS helyett) az Etherboot forrását tartalmazó könyvtárban található Config állomány megfelelõ átírásával tudjuk megtenni.

Itt most floppyról fogjuk indítani a rendszert. A többi módszerrel (PROM vagy MS-DOS® program) kapcsolatban olvassuk el az Etherboot dokumentációját.

A rendszerindító lemez elkészítéséhez tegyünk egy lemezt annak a gépnek a meghajtójába, ahová az Etherboot felkerült. Váltsunk az Etherboot könyvtárán belül az src alkönyvtárba és gépeljük be:

```
# gmake bin32/eszköztípus.fd0
```

Az *eszköztípus* a lemez nélküli munkaállomás Ethernet kártyájától függ. Az ugyanebben a könyvtárban található NIC állományból tudjuk kiolvasni, hogy az adott kártyához melyik *eszköztípus* tartozik.

A rendszer indítása PXE használatával

Alapértelmezés szerint a pxeboot(8) betöltõ a rendszermagot NFS-en keresztül tölti be. Ha az /etc/make.conf állományban a `LOADER_TFTP_SUPPORT` beállítást adjuk meg, akkor TFTP támogatással is lefordítható. Ezzel kapcsolatban a /usr/shared/examples/etc/make.conf állományban található megjegyzéseket érdemes elolvasnunk.

A make.conf állományban még további két másik hasznos opciót is találhatunk a soros vonali konzollal üzemelõ lemez nélküli gépek számára: az egyik a `BOOT_PXELDR_PROBE_KEYBOARD`, a másik pedig a `BOOT_PXELDR_ALWAYS_SERIAL`.

A gép indításakor úgy tudjuk beüzemelni a PXE használatát, ha a BIOS beállításai között a `Boot from network` opciót választjuk ki, vagy a gép bekapcsolása után lenyomjuk hozzá a megfelelõ funkcióbillentyût.

A TFTP és NFS szerverek beállítása

Ha a PXE vagy az Etherboot a TFTP protokollt használja, akkor az állományszerveren a tftpd démont kell elindítani:

1. Készítsünk egy könyvtárat, ahonnan majd a tftpd küldi az állományokat, például legyen ez a /tftpboot.
2. Vegyük fel a következõ sort az /etc/inetd.conf állományunkba:

   ```
   tftp    dgram   udp wait    root    /usr/libexec/tftpd  tftpd -l -s /tftpboot
   ```

A tapasztalat szerint egyes PXE verziók a TFTPTCP alapú változatát használják. Ebben az esetben vegyünk fel még egy második sort is, ahol a `dgram udp` részt `stream tcp`-re cseréljük.

3. Mondjuk meg az inetd démonnak, hogy olvassa újra a konfigurációs állományát. Az alábbi parancs megfelelõ mûködéséhez Az `inetd_enable="YES"` sornak szerepelnie kell az /etc/rc.conf állományban:

```
# /etc/rc.d/inetd restart
```

A tftpboot könyvtárat bárhova rakhatjuk a szerveren. Viszont az inetd.conf és dhcpd.conf állományokban ezt ne felejtsük fel megadni.

Minden esetben engedélyeznünk kell az NFS használatát és vele együtt exportálni az NFS szerverrõl elérni kívánt állományrendszereket.

1. Az /etc/rc.conf állományba tegyük bele a következõt:

```
nfs_server_enable="YES"
```

2. Az /etc/exports állományban a lemez nélküli rendszereknek szánt gyökérkönyvtárat tegyük elérhetõvé (a példában írjuk át a kötet csatlakozási pontját és a *margaux corbieres* helyére állítsuk be a saját lemez nélküli munkaállomásaink neveit:

```
/data/misc -alldirs -ro margaux corbieres
```

3. Kérjük meg a mountd démont, hogy olvassa újra a konfigurációs állományát. Elõfordulhat azonban, hogy ehhez elõször az NFS szolgáltatást kell engedélyezni az /etc/rc.conf állományból és újraindítani a gépet.

```
# /etc/rc.d/mountd restart
```

Lemez nélküli rendszermag fordítása

Ha az Etherboot használata mellett döntünk, akkor a lemez nélküli kliensek számára a rendszermagot a következõ beállítások használatával kell újrafordítani (a megszokottak mellett):

```
options     BOOTP          # BOOTP-n keresztül kérünk IP-címet és hálózati nevet
options     BOOTP_NFSROOT  # a BOOTP-tõl kapott információk alapján csatoljuk a
gyökeret NFS-en keresztül
```

Ezek mellett valószínûleg szükségünk lesz a BOOTP_NFSV3, BOOT_COMPAT és BOOTP_WIRED_TO beállítások megadására is (lásd a NOTES állományt).

A beállítások nevei régrõl származnak és némileg félrevezetõek lehetnek, mivel valójában semmit sem változtatnak a rendszermagban levõ DHCP vagy a BOOTP rutinok használatában (egyébként meg lehet adni vagy az egyik vagy a másik protokoll kizárólágos használatát is).

Fordítsuk le a rendszermagot (lásd A FreeBSD rendszermag testreszabása), és másoljuk a dhcpd.conf állományban megadott helyre.

Amikor a PXE protokollt használjuk, a rendszermagot nem fontos az imént felsorolt paraméterekkel fordítanunk (habár ajánlatos). Az engedélyezésükkel több DHCP kérés keletkezik a rendszermag elindulása közben, ezért kisebb a kockázata annak, hogy a pxeboot(8) által bizonyos esetekben megszerzett és az új értékek között valamilyen ellentmondás jön létre. A használatuk egyik elõnye, hogy így mellékhatásként a hálózati nevünket is megkapjuk. Ellenkezõ esetben erre is találnunk kellene valamilyen módot, például fenntartani egy-egy rc.conf állományt minden kliensen.

Az Etherboot csak akkor lesz képes betölteni a rendszermagot, ha device hinteket is beépítünk. Ezt a következõ beállítással tudjuk megoldani (errõl bõvebben lásd a NOTES állomány megjegyzéseit):

```
hints       "GENERIC.hints"
```

A rendszerindító állományrendszer elõkészítése

A dhcpd.conf állomány root-path beállításának megfelelõen hozzunk létre a rendszer indítására alkalmas gyökér állományrendszert.

Az állományrendszer feltöltése a make world paranccsal

Ezzel a módszerrel a DESTDIR könyvtárba pillanatok alatt telepíteni tudunk egy teljes szûz rendszert (és nem csak a rendszerindító állományrendszert). Ehhez mindössze csak annyit kell tenni, hogy lefuttatjuk a következõ szkriptet:

```
#!/bin/sh
export DESTDIR=/data/misc/diskless
mkdir -p ${DESTDIR}
cd /usr/src; make buildworld && make buildkernel
make installworld && make installkernel
cd /usr/src/etc; make distribution
```

Miután végzett, már csak a DESTDIR könyvtárban található /etc/rc.conf és /etc/fstab állományokat kell az igényeinkhez igazítani.

A lapozóterület beállítása

Amennyiben szükséges, a szerveren található lapozóállományt NFS-en keresztül el tudjuk érni.

Lapozás NFS-sel

A rendszermag maga nem támogatja az NFS alapú lapozás engedélyezését a rendszer indításakor. A lapozóállományt ezért a rendszerindító szkripteken keresztül aktiváljuk, amelyekben csatlakoztatunk egy írható állományrendszert, ahol létrehozzuk és engedélyezzük a lapozóállományt. Tetszõleges méretû lapozóállományt például így tudunk készíteni:

```
# dd if=/dev/zero of=/a/lapozóállomány/helye bs=1k count=1 oseek=100000
```

Az engedélyezéséhez pedig a következõ sort kell felvenni az rc.conf állományba:

```
swapfile=/a/lapozóállomány/helye
```

Egyéb problémák

Írásvédett /usr használata

Ha a lemez nélküli munkaállomáson X szervert akarunk futtatni, akkor az XDM konfigurációs állományait kicsit módosítanunk kell, mert alapértelmezés szerint a /usr könyvtárban hozza létre a naplókat.

Nem FreeBSD-s szerver használata

Amikor a rendszer indításához használt állományrendszert nem egy FreeBSD alapú számítógépen tároljuk, akkor elõször ezt egy FreeBSD-s gépen kell elkészíteni, majd a `tar` vagy `cpio` segítségével átmásolni a megfelelõ helyre.

Ilyen helyzetekben gyakran gondok adódhatnak olyan speciális állományokkal, mint például amelyek a /dev könyvtárban találhatóak, mivel a fõ- és aleszközazonosítók tárolására szánt méret különbözhet. Ezt úgy oldhatjuk meg, ha exportálunk egy könyvtárat a nem FreeBSD alapú szerveren, ezt csatlakoztatjuk a FreeBSD-s gépen, majd a devfs(5) segítségével a eszközleírókat a felhasználó számára észrevétlen módon foglaljuk le.

31.8. ISDN

Az ISDN technológiai és hardveres hátterérõl sokat megtudhatunk Dan Kegel ISDN-rõl szóló oldalán (angolul).

Az ISDN használatát röviden így foglalhatnánk össze:

- Ha Európában élünk, akkor minden bizonnyal az ISDN kártyákkal foglalkozó szakaszt érdemes elolvasnunk.
- Ha elsõsorban betárcsázós ISDN-nel szeretnénk csatlakozni az internetre egy internet-szolgáltatón keresztül, akkor a terminál adaptereket tárgyaló szakaszt nézzük meg. A

szolgáltatók váltásakor ezzel jár a legtöbb rugalmasság és a legkevesebb probléma.

- Ha két helyi hálózat összekötésére használjuk, vagy az internethez egy bérelt ISDN vonalon keresztül kapcsolódunk, akkor egy önálló útválasztó vagy hálózati híd beállításában érdemes gondolkodnunk.

A költség fontos szerepet játszik az elfogadható megoldás kiválasztásában. A most következõ lehetõségeket a legolcsóbbtól indulva kezdjük el felsorolni egészen a legdrágábbig.

31.8.1. ISDN kártyák

A FreeBSD-ben megtalálható ISDN implementáció csak a DSS1/Q.931 (más néven Euro-ISDN) szabvány szerint gyártott passzív kártyákat támogatja. Ismer azonban egyes olyan aktív kártyákat is, amelyeknél a firmware további más jelkezelési protokollokat is támogat. Ilyen többek közt az elsõként támogatott Primary Rate (PRI) ISDN kártya.

Az isdn4bsd szoftver segítségével kapcsolódni tudunk más ISDN útválasztókhoz IP-n keresztül a nyers HDLC felett, vagy szinkron PPP használatával. Mindezeket a rendszermagban található PPP-re vagy az `isppp`-re építkezik.

FreeBSD alatt egyre több PC-s ISDN kártyához készül el a támogatás, és a visszajelzések azt mutatják, hogy Európában és a világ minden részén sikerrel használják ezeket.

A passzív ISDN kártyák közül is leginkább az Infineon (korábban Siemens) gyártmányú ISAC/HSCX/IPAC ISDN chipkészletek támogatottak, de a Cologne chippel rendelkezõ (de csak ISA buszos) ISDN kártyák, a Winbond W6692 chipes PCI buszos kártyák, és a Tiger300/320/ISAC chipkészletek egyes változatai, valamint néhány gyártófüggõ chipkészlettel rendelkezõ kártya, mint például az AVM Fritz!Card PCI V.1.0 és az AVM Fritz!Card PnP is remekül mûködik.

Jelenleg a következõ aktív ISDN kártyákat támogatja a rendszer: AVM B1 (ISA és PCI) BRI kártyák és az AVM T1 PCI PRI kártyák.

Az isdn4bsd dokumentációját a rendszerünkön belül a /usr/shared/examples/isdn/ könyvtárban találhatjuk meg, vagy közvetlenül az isdn4bsd honlapján, ahol több hivatkozást is találunk tippekre, hibajegyzékekre és bõségesebb dokumentációra, például az isdn4bsd saját kézikönyvére.

Ha szeretnénk egy másik ISDN protokoll támogatásának kifejlesztésében résztvenni, vagy egy jelenleg még nem támogatott ISDN kártyát használhatóvá tenni, esetleg valamilyen más módon segíteni az isdn4bsd ügyét, vegyük fel a kapcsolatot Hellmuth Michaelis <hm@FreeBSD.org> fejlesztõvel.

Az isdn4bsd telepítésével, beállításával és hibaelhárításával kapcsolatos kérdéseinket a freebsd-isdn levelezési listán tehetjük fel.

31.8.2. ISDN terminál adapterek

Az ISDN számára olyanok a terminál adapterek, mint a hagyományos telefonvonalak számára a modemek.

A legtöbb terminál adapter a Hayes-modemek szabványos AT parancskészletét használja, és könnyen be lehet iktatni egy modem helyett.

A terminál adapterek alapvetõen ugyanúgy mûködnek, mint a modemek, kivéve, hogy egy átlagos modemnél jóval nagyobb adatátviteli sebességre képesek. Ezért a PPP kapcsolatunkat pontosan ugyanúgy kell beállítani, mint a modemek esetében. Ne felejtsük a soros pont sebességét a maximális értékre állítani.

A terminál adapterek használatának egyik legnagyobb elõnye, hogy segítségükkel dinamikus PPP-n keresztül tudunk az internet-szolgáltatónkhoz kapcsolódni. Mivel az IP-címtartomány egyre inkább szûkösebb, a legtöbb szolgáltató nem szívesen oszt ki bárkinek is statikus IP-címet. A legtöbb önálló útválasztó azonban nem képes alkalmazkodni az IP-címek dinamikus kiosztásához.

A terminál adapter az elérhetõ lehetõségeket és a kapcsolat stabilitását tekintve teljesen a PPP démontól függ. Emiatt egy FreeBSD-s gépet könnyû modemrõl átállítani az ISDN használatára, ha már egyszer beállítottuk a PPP démont. Ezzel együtt azonban a PPP használata során tapasztalt problémák ugyanúgy ismét felmerülnek.

Ha a maximális stabilitásra van szükségünk, akkor a rendszermag PPP beállítását használjuk, és ne a felhasználói PPP megoldást.

A FreeBSD hivatalosan az alábbi terminál adaptereket ismeri:

- Motorola BitSurfer és Bitsurfer Pro
- Adtran

Valószínûleg a többi terminál adapterrel is képes együttmûködni, mivel a terminál adapterek gyártói általában igyekeznek a termékeiket a szabványos modemes AT parancskészletével kompatibilissá tenni.

Az igazi probléma a külsõ terminál adapterekkel adódik, mivel, akárcsak a modemek esetében, egy nagyon jó soros kártyát igényelnek.

A soros eszközök mûködésének részleteit valamint az aszinkron és szinkron soros portok közti különbségeket a FreeBSD soros hardverekrõl szóló cikkében olvashatjuk.

A terminál adaptereken keresztül elérhetõ sebességet a PC-kben található szabványos (aszinkron) soros port 115,2 Kb/mp-re korlátozza, még 128 Kb/mp-es adatátvitelû kapcsolatok esetében is. Az ISDN által nyújtott 128 Kb/mp kihasználásához a terminál adaptert egy szinkron soros kártyával kell összekötnünk.

Ne higyjük, hogy egy belsõ terminál adapter megvásárlásával megmenekülünk ettõl a gondtól. A belsõ terminál adapterekbe egyszerûen csak egy sima szabványos PC-s soros portot építettek bele. Mindössze egy soros kábelt és egy konnektort takarítunk meg velük.

A terminál adapterhez csatlakozó szinkron kártyák legalább olyan gyorsak, mint egy önálló útválasztó, és egy egyszerû 386-osra épülõ FreeBSD rendszerrel talán még rugalmasabban is kezelhetõek.

A terminál adapter plusz szinkron kártya kontra önálló útválasztó kérdése már hitkérdéssé fajult, amirõl igen sokat vitatkoztak szerte a levelezési listákon. A teljes okfejtés elolvasásához az archívum böngészését javasoljuk.

31.8.3. Önálló ISDN hálózati hidak és útválasztók

Az ISDN hidak vagy útválasztók nem egészen a FreeBSD vagy operációs rendszerek területéhez tartoznak. Az útválasztás és a hálózatok hidak alapjainak a számítógépes hálózatokról szóló szakirodalomban járhatunk utána.

Ebben a szakaszban a hálózati híd és az útválasztó kifejezéseket egymás szinonímájaként fogjuk használni.

Ahogy az olcsóbb ISDN útválasztók és hidak árai egyre jobban csökkennek, ezért egyre inkább népszerûbbé válnak. Az ISDN útválasztó egy apró doboz, amelyet közvetlenül a helyi Ethernet hálózatunkra tudunk csatlakoztatni, és a többi útválasztóhoz vagy hídhoz kapcsolódik. A benne található szoftverrel képes kommunikálni a PPP vagy más egyéb népszerû protokollokon keresztül.

Az útválasztó egy szabványos terminál adapternél sokkal nagyobb adatátvitelt tesz lehetõvé, mivel a teljes szinkron ISDN kapcsolatot képes kihasználni.

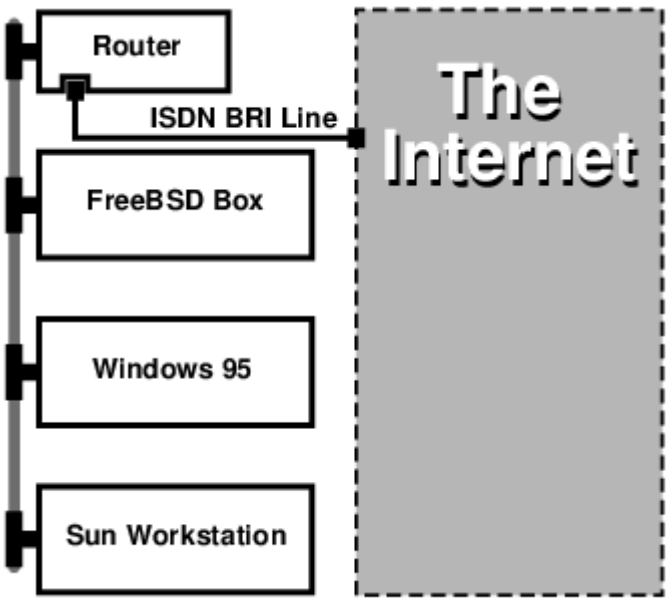
Az ISDN útválasztókkal és hidakkal kapcsolatban az egyik legnagyobb problémát a különbözõ gyártók közti eltérések jelenthetik. Ha egy szolgáltatóhoz akarunk ezen a módon csatlakozni, akkor érdemes elõzetesen egyeztetni az igényeinket velük.

Ha két helyi hálózati szegmenst akarunk összekapcsolni, mint például az otthoni és az irodai hálózatot, akkor ez a megoldás jár a legkevesebb karbantartási költséggel. Mivel ekkor mi magunk vásároljuk a kapcsolat mind a két oldalára a felszerelést, biztosak lehetünk benne, hogy az így létrehozott összekötettés mûködni fog.

Például, ha egy otthon vagy a vállalat egy fiókjánál levõ gépet akarjuk összekötni az igazgatóság hálózatával, akkor a következõ felállást érdemes követnünk:

Példa 40. Egy otthoni vagy egy fiókbeli hálózat

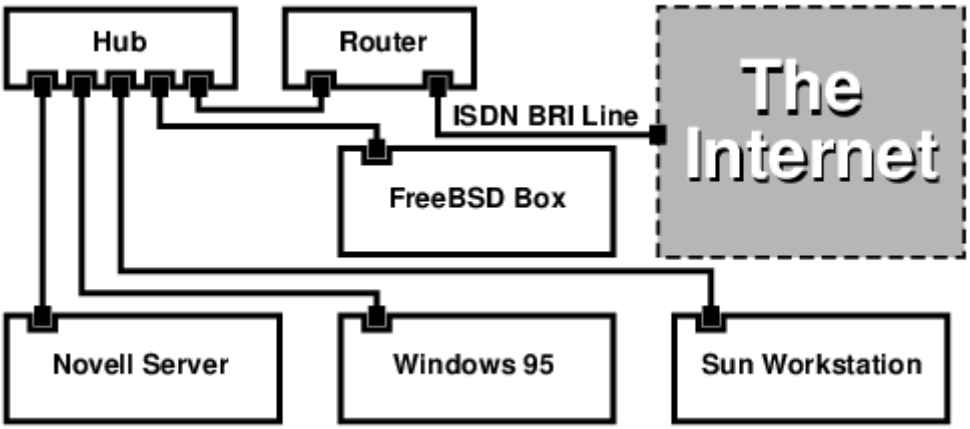
A hálózat busz topológiájú és 10 Base 2 Ethernetet használ ("thinnet"). Ha szükséges, akkor az útválasztót egy AUI/10BT adó-vevõvel csatlakoztassuk a hálózati kábelre.



Ha az otthoni vagy fiókbeli számítógép az egyedüli, akkor egy keresztkötésû sodrott érpár kábellel akár közvetlenül is csatlakozhatunk az útválasztóhoz.

Példa 41. Az igazgatósági iroda vagy egy másik helyi hálózat

A hálózat csillag topológiájú, és 10 Base T Ethernet kábelezésû ("sodrott érpár").



A legtöbb útválasztó/híd elõnye, hogy *egyszerre* 2 *egymástól független* PPP kapcsolatot tudunk felépíteni velük 2 egymástól független géppel. Ezt a legtöbb terminál adapter nem támogatja, kivéve azok a (általában drága) típusok, amelyek két soros porttal rendelkeznek. Ezt ne tévesszük össze a csatornák nyalábolásával, az MPP-vel és a többivel.

Ez nagyon hasznos lehet például olyan esetekben, amikor van egy dedikált ISDN kapcsolatunk az irodában, amelyet ugyan szeretnénk megcsapolni, de nem szeretnénk a másik ISDN vonalat is elrabolni. Az irodában levõ A útválasztó képes a dedikált B csatornájú kapcsolaton (64 Kb/mp) keresztül elérni az internetet, miközben a másik B csatornát ettõl független adatkapcsolatra használja. A második B csatorna így használható betárcsázásra, kitárcsázásra vagy a másik B csatornával együtt dinamikus nyalábolásra (MPP stb.) a nagyobb sávszélesség elérése érdekében.

Az Ethernetes híd nem IP alapú forgalmat is képes továbbítani, ezért rajta keresztül akár IPX vagy SPX és más egyéb protokollokat is használni tudunk.

31.9. Hálózati címfordítás

31.9.1. Áttekintés

A FreeBSD hálózati címfordításért felelõs démonprogramja, a natd(8) (Network Address Translation daemon), a beérkezõ nyers IP csomagokat dolgozza fel, és a helyi gépek forráscímét kicserélve visszailleszti ezeket a csomagokat a kimenõ folyamba. A natd(8) mindezt úgy teszi a forrás IP-címekkel és portokkal, hogy amikor az adat visszaérkezik, akkor képes lesz megmondani a csomag eredeti küldõjét és visszaküldeni neki a választ.

A hálózati címfordítást általában az internet-kapcsolatok megosztásánál alkalmazzuk.

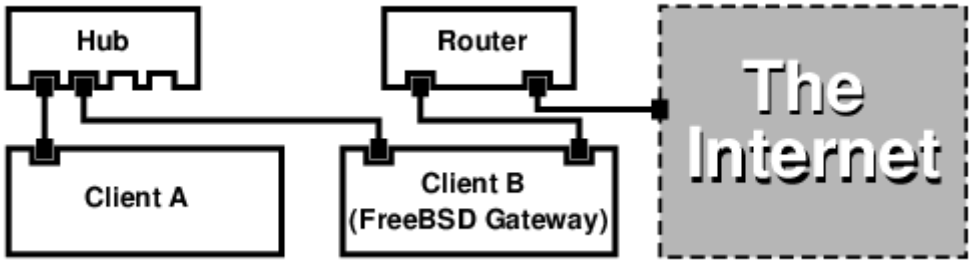
31.9.2. A hálózat felépítése

Az IPv4 világában egyre jobban fogyó IP-címek és az egyre növekvõ számú, nagysebességre vágyó, például kábeles vagy DSL-es fogyasztók miatt az igény is egyre nagyobb az internet-kapcsolatok megosztására. Ha több számítógéppel szeretnénk egyetlen kapcsolaton és egy IP-címen keresztül kapcsolódni az internetre, akkor ehhez a natd(8) tökéletes választás.

Az esetek többségében a felhasználók egy kábeles vagy DSL vonalra csatlakoznak, melyhez egyetlen IP-cím tartozik, és ezen a gépen keresztül szeretnék elérni az internetet a helyi hálózaton levõ többi géprõl.

Ezt úgy tudjuk elérni, ha az internethez kapcsolódó FreeBSD-s gépet átjárónak állítjuk be. Ebben az átjáróban legalább két hálózati felületnek kell léteznie - az egyikkel az internetes útválasztóhoz, a másikkal pedig a helyi hálózathoz kapcsolódik. A belsõ hálózaton levõ gépek egy hub vagy egy switch segítségével csatlakoznak egymáshoz.

Több módon is el tudjuk érni a belsõ hálózatról az internetet egy FreeBSD-s átjárón keresztül. Ebben a példában most csak olyan átjárókkal foglalkozunk, amelyekben legalább két hálózati kártya található.



Egy ehhez hasonló beállítás igen gyakori a megosztott internet-kapcsolatok esetében. A helyi hálózat egyik gépe csatlakozik az internetre. A többi gép ezen az "átjárón" keresztül éri el az internetet.

31.9.3. A rendszerbetöltõ beállítása

A natd(8) mûködéséhez szükséges címfordítási támogatást a GENERIC típusú rendszermagok nem tartalmazzák, viszont a /boot/loader.conf megfelelõ paraméterezésével a rendszer betöltése közben ezt hozzá tudjuk adni:

```
ipfw_load="YES"
ipdivert_load="YES"
```

Valamint a `net.inet.ip.fw.default_to_accept` változót állítsuk az `1` értékre.

```
net.inet.ip.fw.default_to_accept="1"
```

Ez utóbbi beállítást leginkább a tûzfal és a címfordítást végzõ átjáró

> próbálgatásakor érdemes alkalmazni. Ilyenkor ugyanis az ipfw(8) alapértelmezett módon az `allow ip from any to any` (minden forgalom engedélyezett) szabályt követi, és nem pedig a kevésbé barátságos `deny ip from any to any` (minden forgalom tiltott) szabályt. A rendszer újraindításakor így valamivel nehezebb lesz kizárnunk magunkat a szabályok megadása során.

31.9.4. A rendszermag beállítása

Amikor viszont nincs lehetõségünk modulok használatára, vagy szeretnénk minden igényelt funkciót beépíteni a rendszermagba, akkor a rendszermag beállításait tartalmazó állományban a következõket kell megadnunk:

```
options IPFIREWALL
options IPDIVERT
```

A fentiek mellett még ezeket a lehetõségeket tudjuk választani:

```
options IPFIREWALL_DEFAULT_TO_ACCEPT
options IPFIREWALL_VERBOSE
```

31.9.5. A rendszerindítás beállítása

A tûzfal és a hálózati címfordítás beindításához a következõknek kell az /etc/rc.conf állományban lennie:

```
gateway_enable="YES" ①
firewall_enable="YES" ②
firewall_type="OPEN" ③
natd_enable="YES"
natd_interface="fxp0" ④
natd_flags="" ⑤
```

① A gépet átjárónak állítja be. Hatása megegyezik a `sysctl net.inet.ip.forwarding=1` parancs kiadásával.

② A rendszer indításakor engedélyezi az /etc/rc.firewall állományban szereplõ tûzfalszabályok használatát.

③ Egy olyan elõre definiált tûzfalat ad meg, amely alapból mindent beenged. Az /etc/rc.firewall állományban találhatjuk a többi típust.

④ Megadja, hogy melyik felületen továbbítsunk csomagokat az internet felé (ez a felület csatlakozik az internetre).

⑤ Itt szerepel minden további paraméter, amelyet még az indításkor át kell adnunk a natd(8) démonnak.

Amikor megadjuk ezeket a beállításokat az /etc/rc.conf állományban, pontosan ugyanaz történik,

mintha a `natd -interface fxp0` parancsot adtunk volna ki a rendszer indításakor. Ez tehát manuálisan is elindítható.

Ha túlságosan sok paramétert akarunk egyszerre beállítani natd(8) használatához, akkor akár egy külön konfigurációs állományt is megadhatunk. Ebben az esetben a konfigurációs állományt a következõ módon kell megjelölni az /etc/rc.conf állományban:

```
natd_flags="-f /etc/natd.conf"
```

Ekkor a /etc/natd.conf állomány fogja tartalmazni a beállításokat, soronként egyet. Például a következõ szakaszban ez lesz a tartalma:

```
redirect_port tcp 192.168.0.2:6667 6667
redirect_port tcp 192.168.0.3:80 80
```

A konfigurációs állományról és az `-f` opció használatával kapcsolatban olvassuk el a natd(8) man oldalát.

A helyi hálózaton mindegyik gépnek az RFC 1918 által megadott privát IP-címterekbõl származó címet kell használnia, és az alapértelmezett átjárónak mindenhol a natd démont futtató gép IP-címét kell megadni.

Például a belsõ hálózaton található `A` és `B` kliensek IP-címei rendre `192.168.0.2` és `192.168.0.3`, míg a natd(8) démont futtató gép belsõ címe `192.168.0.1`. Az `A` és a `B` kliens alapértelmezett átjáróját a natd gépre, vagyis a `192.168.0.1` címre kell beállítanunk. A natd gép külsõ, avagy internetes felülete semmilyen további módosítást nem igényel a natd(8) mûködéséhez.

31.9.6. A portok átirányítása

A natd(8) alkalmazásának hátránya, hogy a belsõ hálózatra csatlakozó kliensek az internetrõl nem érhetõek el. Tehát a helyi hálózat kliensei képesek elérni a külvilágot, de az visszafelé már nem igaz. Ez akkor jelent igazából problémát, ha az egyik belsõ kliensen szolgáltatásokat akarunk futtatni. A probléma egyik egyszerû megoldása, ha a natd használatával az internet felõl egyszerûen átirányítunk bizonyos portokat a megfelelõ belsõ kliensre.

Például tegyük fel, hogy az `A` kliens egy IRC szervert, míg a `B` kliens egy webszervert futtat. Ez akkor fog mûködni, ha a szolgáltatásokhoz tartozó 6667 (IRC) és 80 (web) portokat átirányítjuk a hozzájuk tartozó gépek felé.

Ehhez a natd(8) démonnak a `-redirect_port` paramétert kell átadni. A pontos felírás így néz ki:

```
    -redirect_port protokoll célIP:célPORT[-célPORT]
                [külsõIP:]külsõPORT[-külsõPORT]
                [távoliIP[:távoliPORT[-távoliPORT]]]
```

A fenti példában tehát ezt kell megadnunk:

```
    -redirect_port tcp 192.168.0.2:6667 6667
    -redirect_port tcp 192.168.0.3:80 80
```

Így az egyes külsõ *tcp* portokat átirányítjuk a belsõ hálózat gépei felé.

A `-redirect_port` paraméternek akár egész porttartományokat is megadhatunk. Például a *tcp 192.168.0.2:2000-3000 2000-3000* megadásával az összes 2000-tõl 3000-ig terjedõ port csatlakozását leképezzük az `A` kliens 2000 és 3000 közti portjaira.

Ezek a beállítások a natd(8) közvetlen futtatásakor adhatóak meg, esetleg az /etc/rc.conf állományban az `natd_flags=""` opció keresztül, vagy egy külön konfigurációs állományban.

A többi beállítási lehetõséget a natd(8) man oldalán ismerhetjük meg.

31.9.7. A címek átirányítása

A címek átirányítása abban az esetben hasznos, amikor több IP-cím áll rendelkezésünkre, de ezek egy géphez tartoznak. Ilyenkor az natd(8) képes a belsõ hálózat egyes gépeihez saját külsõ IP-címet rendelni. A natd(8) a belsõ hálózat kliensei által küldött csomagokban kicseréli a címüket a megfelelõ külsõ IP-címmel, illetve az ezekre a címekre érkezõ forgalmat továbbítja a megfelelõ belsõ kliens irányába. Ezt a megoldást statikus hálózati címfordításnak is nevezzük. Például a `128.1.1.2` és a `128.1.1.3` IP-címek a natd démont futtató átjáróhoz tartoznak. A `128.1.1.1` cím használható a natd alapú átjáró külsõ IP-címeként, miközben a `128.1.1.2` és a `128.1.1.3` címeket a belsõ hálózaton elérhetõ `A` és `B` kliensek felé közvetítjük.

A `-redirect_address` felírása tehát a következõ:

```
-redirect_address helyiIP publikusIP
```

helyiIP	A helyi hálózaton található kliens saját IP-címe.
publikusIP	A klienshez tartozó megfelelõ külsõ IP-cím.

Az iménti példában a pontos paraméterek ezek lesznek:

```
-redirect_address 192.168.0.2 128.1.1.2
-redirect_address 192.168.0.3 128.1.1.3
```

A `-redirect_port` opcióhoz hasonlóan ez is megadható az /etc/rc.conf állományban az `natd_flags=""` beállításon keresztül vagy egy külön konfigurációs állományban. A címek átirányításával nincs szüksége a portok átirányítására, mivel az adott IP-címhez tartozó összes forgalmat átirányítjuk.

A natd démont futtató gépen a külsõ IP-címeket aktiválni kell és a külsõ felületéhez kell rendelni. A rc.conf(5) man oldalon járhatunk utána, hogy mindezt hogyan is tudjuk megcsinálni.

31.10. Párhuzamos vonali IP (PLIP)

A párhuzamos vonali IP (Parallel Line IP, PLIP) a TCP/IP protokoll használatát valósítja meg párhuzamos porton keresztül. Olyan gépek számára lehet hasznos, amelyekben nincs hálózati kártya, vagy esetleg laptopoknál. Ebben a szakaszban a következõket tárgyaljuk:

- Párhuzamos (laplink) kábel készítése
- Két számítógép összekapcsolása a PLIP segítségével

31.10.1. Párhuzamos kábel készítése

Párhuzamos kábelt a legtöbb számítástechnikai boltban tudunk vásárolni. Ha mégsem tudnánk sehol sem beszerezni, vagy egyszerûen tudni szeretnénk, hogyan lehet ilyet készíteni, akkor az alábbi táblázatban láthatjuk, hogy miként tudunk egy hétköznapi nyomtatókábelt átalakítani a céljainkra.

Táblázat 12. A párhuzamos kábel hálózati használatra alkalmas bekötése

A-név	A-vég	B-vég	Leírás	Post/Bit
.... DATA0 -ERROR	.... 2 15	.... 15 2	Adat	.... 0/0x01 1/0x08 ....
.... DATA1 +SLCT	.... 3 13	.... 13 3	Adat	.... 0/0x02 1/0x10 ....
.... DATA2 +PE	.... 4 12	.... 12 4	Adat	.... 0/0x04 1/0x20 ....
.... DATA3 -ACK	.... 5 10	.... 10 5	Vál. imp.	.... 0/0x08 1/0x40 ....
.... DATA4 BUSY	.... 6 11	.... 11 6	Adat	.... 0/0x10 1/0x80 ....
GND	18-25	18-25	Föld	-

31.10.2. A PLIP beállítása

Elõször is szereznünk kell valahonnan egy laplink kábelt. Ha ez megvan, akkor mind a két gépen ellenõrizzük, hogy a rendszermag tartalmazza az lpt(4) meghajtót:

```
# grep lp /var/run/dmesg.boot
lpt0: <Printer> on ppbus0
lpt0: Interrupt-driven port
```

A párhuzamos portnak megszakítással vezéreltnek kell lennie ("interrupt driven"), és az /boot/device.hints állományban szerepelnie kell nagyjából a következõ soroknak:

```
hint.ppc.0.at="isa"
hint.ppc.0.irq="7"
```

Ezután nézzük meg, hogy a rendszermag beállításait tartalmazó állományban megjelenik-e a `device plip` sor, vagy a plip.ko modul betöltõdött-e. Akármelyik is történt, a párhuzamos hálózati felület most már a rendelkezésünkre áll, és az ifconfig(8) paranccsal ezt meg is tudjuk nézni:

```
# ifconfig plip0
plip0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> mtu 1500
```

A laplink kábelt csatlakoztassuk mind a két számítógéphez.

Mind a két a hálózati felület paramétereit `root` felhasználóként hangoljuk be. Például, ha az `egyikgép` nevû gépet akarjuk a `másikgép` nevû géphez csatlakoztatni:

```
             egyikgép <-----> másikgép
IP-cím       10.0.0.1      10.0.0.2
```

Az `egyikgép` felületét így állítsuk be:

```
# ifconfig plip0 10.0.0.1 10.0.0.2
```

A `másikgép` felületét így állítsuk be:

```
# ifconfig plip0 10.0.0.2 10.0.0.1
```

Ezt követõen már egy mûködõ kapcsolatnak kell felépülnie. Az egyéb részletek kapcsán az lp(4) és az lpt(4) man oldalait nézzük át.

Ezt a két gépet vegyük fel az /etc/hosts állományba is:

```
127.0.0.1               localhost.saját.tartomány localhost
10.0.0.1                egyikgép.saját.tartomány egyikgép
10.0.0.2                másikgép.saját.tartomány
```

A kapcsolat mûködõképességérõl úgy tudunk meggyõzõdni, ha az egyik géprõl megpróbáljuk pingelni a másikat. Például az `egyikgép` esetében:

```
# ifconfig plip0
plip0: flags=8851<UP,POINTOPOINT,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet 10.0.0.1 --> 10.0.0.2 netmask 0xff000000
# netstat -r
Routing tables

Internet:
Destination        Gateway           Flags     Refs     Use      Netif Expire
másikgép           egyikgép          UH          0       0       plip0
# ping -c 4 másikgép
PING másikgép (10.0.0.2): 56 data bytes
64 bytes from 10.0.0.2: icmp_seq=0 ttl=255 time=2.774 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=255 time=2.530 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=255 time=2.556 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=255 time=2.714 ms

--- másikgép ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.530/2.643/2.774/0.103 ms
```

31.11. Az IPv6

Az IPv6 (másik néven az IPng, vagy a "az internet következõ generációs protokollja", "IP next generation") a jól ismert IP protokoll (avagy az IPv4) új változata. Hasonlóan a jelenleg mûködõ összes többi BSD rendszerhez, a FreeBSD is tartalmazza a KAME IPv6 referencia implementációt. Ezért ha ezzel szeretnénk kísérletezni, akkor ehhez a FreeBSD minden eszköz biztosít számunkra. Ez a szakasz az IPv6 beállítását és használatát mutatja be.

Az 1990-es évek elején az IPv4-es címterek rohamos mértékû kimerülését figyelték meg. Az internet jelenlegi bõvülési üteme mellett két nagyobb aggodalomnak adott okot:

- A címek elfogyása. Napjainkban efelõl egyre kevesebb a kétség, mivel az RFC 1918 által megfogalmazott privát címterek (`10.0.0.0/8`, `172.16.0.0/12`, és `192.168.0.0/16`), valamint a hálózati címfordítás (Network Address Translation, NAT) használata igen elterjedt.
- Az útválasztási táblázatok méretének növekedése. Ez még manapság is aggasztó.

Az IPv6 ezeket és még más egyéb problémákat a következõ módon igyekszik megoldani:

- A 128 bites címtér használata. Más szóval, elméletben összesen 340 282 366 920 938 463 463 374 607 431 768 211 456 darab címet képes kiosztani. Ez azt jelenti, hogy bolygónk minden egyes négyzetméterére megközelítõleg 6,67 * 10^27 IPv6 típusú cím jut.
- Az útválasztók a saját táblázataikban csak a hálózatok összevont címeit tárolják el, ezáltal egy átlagos útválasztási táblázatban található bejegyzések száma 8192 alá csökken.

Az IPv6 emellett még rengeteg más elõnyös lehetõséget is kínál:

- A címek automatikus beállítása (lásd RFC 2462)
- Anycast (bárkiküldés, vagyis "egy a sokból")
- Kötelezõ (mandatory) multicast
- IPsec (IP szintû védelem)
- Egyszerûsített fejléc
- Mobil IP
- IPv6-IPv4 közti átjárhatóság

Ha mindezekrõl többet szeretnénk megtudni, akkor erre érdemes továbblépnünk:

- Az IPv6 áttekintése a playground.sun.com honlapon
- KAME.net

31.11.1. Az IPv6 címek háttere

Az IPv6 címeknek több típusa létezik: a unicast (egyesküldés), az anycast (bárkiküldés) és a multicast (többesküldés).

A unicasthez használt címek jól ismert címek. Az így elküldött csomag pontosan ahhoz a felülethez érkezik meg, amelyhez az adott cím tartozik.

Az anycasthez használt címek felírásukban tökéletesen megegyeznek a unicast esetével, de valójában felületek egy csoportját címezik. Az anycastre beállított címekre küldött csomagok mindig a(z útválasztó szerinti) legközelebb levõ felülethez érkeznek meg. Az anycastet az útválasztók számára találták ki.

A multicasthez használt címek felületek egy csoportját nevezik meg. A multicast címekre érkezõ csomagokat a csoport minden egyes tagja megkapja.

Az IPv4 esetében az üzenetszórásra szánt (általában az `xxx.xxx.xxx.255` formátumú) címeket az IPv6 esetében multicast címekkel fejezzük ki.

Táblázat 13. Fenntartott IPv6 címek

IPv6 cím	Az elõtag hossza (bitekben)	Leírás	Megjegyzés
`::`	128 bit	nem specifikált	Vö. a `0.0.0.0` címmel az IPv4 esetében.
`::1`	128 bit	saját cím	Vö. a `127.0.0.1` címmel az IPv4 esetében.
`::00:xx:xx:xx:xx`	96 bit	IPv4 beágyazása	Az alsó 32 bit egy IPv4 formátumú cím. Ezt "IPv4 kompatibilis IPv6 címnek" is nevezik.

IPv6 cím	Az elõtag hossza (bitekben)	Leírás	Megjegyzés
`::ff:xx:xx:xx:xx`	96 bit	IPv4-re leképzett IPv6 címek	Az alsó 32 bit egy IPv4 címet jelöl. Olyan gépeknél használatos, amelyek nem támogatják az IPv6 protokollt.
`fe80::` - `feb::`	10 bit	helyi összeköttetés	Vö. az IPv4 loopback címeivel.
`fec0::` - `fef::`	10 bit	helyi cím	
`ff::`	8 bit	multicast	
`001` (2-es alapú)	3 bit	globális unicast	Az összes globális unicast címet ebbõl a tartományból osztjuk ki. Az elsõ 3 bit értéke"001".

31.11.2. Az IPv6 címek olvasása

Az IPv6 címek kanonikus formája így ábrázolható: `x:x:x:x:x:x:x:x`, ahol mindegyik "x" egy 16 bites hexadecimális érték. Például: `FEBC:A574:382B:23C1:AA49:4592:4EFE:9982`.

Gyakran a címek hosszú nullákból álló sorozatokat tartalmaznak, ezért mindegyik ilyen sorozatot rövidíteni tudjuk a "::" jelöléssel. Rajtuk kívül még az egyes hexadecimális csoportokban a bevezetõ nullák is elhagyhatóak. Például az `fe80::1` cím kanonikus formája: `fe80:0000:0000:0000:0000:0000:0000:0001`.

A harmadik forma szerint az utolsó 32 bites részt írjuk fel a megszokott (decimális) IPv4 stílusú pontozással, ahol tehát a "." választja el a tagokat. Így például a `2002::10.0.0.1` felírás a `2002:0000:0000:0000:0000:0000:0a00:0001` kanonikus (hexadecimális) ábrázolásnak feleltethetõ meg, ami pedig egyszerûen `2002::a00:1` alakban is megadható.

Mostanra már minden bizonnyal a kedves olvasó érteni fogja a következõt:

```
# ifconfig
```

```
rl0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
        inet 10.0.0.10 netmask 0xffffff00 broadcast 10.0.0.255
        inet6 fe80::200:21ff:fe03:8e1%rl0 prefixlen 64 scopeid 0x1
        ether 00:00:21:03:08:e1
        media: Ethernet autoselect (100baseTX )
        status: active
```

A `fe80::200:21ff:fe03:8e1%rl0` cím az automatikusan beállított helyi összeköttetés címe. Ez az

automatikus beállítás részeként a MAC-címbõl jött létre.

Az IPv6 címek szerkezetérõl további részleteket az RFC 3513-ban találunk.

31.11.3. Kapcsolódás

Jelenleg négy módon tudunk más IPv6-os géphez és hálózathoz csatlakozni:

- Kérjünk a hálózati elérésünkért felelõs illetékesektõl IPv6 alapú hálózatot. A részletek tekintetében vegyük fel a kapcsolatot az internet-szolgáltatónkkal.
- A SixXS a világ minden táján kínál végpontokkal rendelkezõ tunneleket.
- Egy 6-ból-4 (RFC 3068) típusú tunnellel.
- Ha betárcsázós kapcsolatunk van, akkor használjuk a net/freenet6 portot.

31.11.4. A nevek feloldása az IPv6 világában

IPv6 alatt régebben két típusa volt a nevek feloldásáért felelõs rekordoknak. Az IETF az A6 rekordokat idõközben elavultnak nyilvánította. Ezért manapság már az AAAA rekordok tekinthetõek szabványosnak.

Az AAAA rekordok használata magától értetõdik. A hálózati nevükhöz az alábbi módon tudunk IPv6 címet rendelni az elsõdleges zónát leíró állományban:

```
SAJÁTNÉV          AAAA    SAJÁTIPv6CÍM
```

Ha nem rendelkezünk saját névfeloldási zónával, akkor erre kérjük meg a névfeloldást végzõ szolgáltatónkat. A bind jelenlegi változatai (8.3 és 9), valamint a dns/djbdns (IPv6 támogatására vonatkozó javítással) támogatják az AAAA rekordokat.

31.11.5. Az /etc/rc.conf szükséges módosításai

Az IPv6 kliensek beállításai

Ezek a beállítások egy helyi hálózaton levõ gépre vonatkoznak, nem pedig egy útválasztóra. Az rtsol(8) az alábbi megadásával fogja automatikusan beállítani a felületeinket a rendszer indításakor:

```
ipv6_enable="YES"
```

Ha az fxp0 felülethez statikusan akarunk IP-címet rendelni, például a `2001:471:1f11:251:290:27ff:fee0:2093` címet, akkor ehhez a következõt kell megadni:

```
ipv6_ifconfig_fxp0="2001:471:1f11:251:290:27ff:fee0:2093"
```

Az /etc/rc.conf állományban az alapértelmezett átjárót a következõ módon tudjuk a

2001:471:1f11:251::1 címre beállítani:

```
ipv6_defaultrouter="2001:471:1f11:251::1"
```

Az IPv6 útválasztók és átjárók beállítása

Itt most a tunnelt biztosító szolgáltató által mutatott irányt követjük, és olyan formára alakítjuk, amely megmarad az újraindítás után is. A rendszer indításakor az /etc/rc.conf állományban valami ilyesmit kell megadni a járat visszaállításához:

Soroljuk fel a beállítandó általános tunnel alapú felületeket, ilyen lehet például a gif0:

```
gif_interfaces="gif0"
```

A felületnek állítsunk be egy helyi végpontot a *SAJÁT_IPv4_CÍM* megadásával, valamint egy távoli végpontot a *TÁVOLI_IPv4_CÍM* megadásával:

```
gifconfig_gif0="SAJÁT_IPv4_CÍM TÁVOLI_IPv4_CÍM"
```

Az IPv6 tunnelünk végpontjához kapott cím aktiválásához az alábbit kell még megadnunk:

```
ipv6_ifconfig_gif0="SAJÁT_KAPOTT_IPv6_TUNNEL_VÉGPONTJÁNAK_CÍME"
```

Ezután már csak az alapértelmezett útvonalat kell beállítani az IPv6 számára. Ez az IPv6 járat másik oldala:

```
ipv6_defaultrouter="SAJÁT_IPv6_TÁVOLI_TUNNEL_VÉGPONTJÁNAK_CÍME"
```

Az IPv6 tunnel beállításai

Amennyiben a szerver IPv6 alapú forgalmat közvetít a hálózatunk és a világ között, az /etc/rc.conf állományba a következõt kell felvennünk:

```
ipv6_gateway_enable="YES"
```

31.11.6. Az útválasztók kihirdetése és automatikus konfigurációja

Ebben a szakaszban az rtadvd(8) beállításával fogjuk az alapértelmezett IPv6 útvonalat kihirdetni.

Az rtadvd(8) engedélyezéséhez az alábbi sort kell betennünk az /etc/rc.conf állományba:

```
rtadvd_enable="YES"
```

Emellett még fontos megadnunk azt a felületet, ahol az IPv6 útválasztó kérelmezését végezzük. Ha erre a feladatra például az fxp0 felületet választjuk, akkor erről az rtadvd(8) így értesíthetõ:

```
rtadvd_interfaces="fxp0"
```

Most pedig készítenünk kell hozzá egy konfigurációt is, vagyis az /etc/rtadvd.conf állományt. Íme erre egy példa:

```
fxp0:\
    :addrs#1:addr="2001:471:1f11:246::":prefixlen#64:tc=ether:
```

Az fxp0 felületet természetesen cseréljük ki a sajátunkkal.

Ezután a `2001:471:1f11:246::` címre helyére írjuk be a saját kiosztásunk elõtagját.

Egy egész `/64` alhálózat esetén nem is kell többet megadni. Minden más helyezetben az elõtag hosszára `prefixlen#` vonatkozó értéket is be kell még állítanunk.

31.12. Az Aszinkron adatátviteli mód (ATM)

31.12.1. A klasszikus IP-címek beállítása ATM felett (állandó)

A klasszikus IP ATM felett (Classical IP over ATM, CLIP) a legegyszerûbb módszer az IP-címek használatára az Aszinkron adatátviteli móddal (Asynchronous Transfer Mode, ATM) együtt. Kapcsolt és állandó kapcsolatok (Switched Virtual Channel, SVC és Permanent Virtual Channel, PVC) esetén egyaránt megfelelõ. Ebben a szakaszban ez utóbbival fogunk foglalkozni.

A teljesen hálószerû konfigurációk

A CLIP beállítását állandó csatornákon például úgy tudjuk megoldani, ha az összes gépet külön ezekre a célokra szánt állandó csatornákkal összekapcsoljuk egymással. Ez az egyszerû megoldás azonban nagyobb számú gép esetében már nem eléggé hatékony. A következõ példában csupán négy gépet kötünk hálózatba, melyik mindegyike egy ATM kártyával csatlakozik az ATM hálózatra. Ehhez elsõként tervezzük meg az IP-címek kiosztását és a gépek közti ATM kapcsolatokat. A példában ez az alábbiak szerint alakul:

Gép	IP-cím
A-gep	192.168.173.1
B-gep	192.168.173.2
C-gep	192.168.173.3
D-gep	192.168.173.4

A teljes hálózat felépítéséhez minden egyes pár között egy-egy ATM kapcsolatra lesz szükségünk:

Gépek	VPI.VCI pár
A-gep - B-gep	0.100
A-gep - C-gep	0.101
A-gep - D-gep	0.102
B-gep - C-gep	0.103
B-gep - D-gep	0.104
C-gep - D-gep	0.105

A kapcsolatok egyes végein szereplõ VPI és VCI értékek természetesen eltérhetnek, de ezeket mi most az egyszerûség kedvéért egyenlõnek tekintettük. A következõ lépésben minden gépen állítsuk be az ATM felület:

```
A-gep# ifconfig hatm0 192.168.173.1 up
B-gep# ifconfig hatm0 192.168.173.2 up
C-gep# ifconfig hatm0 192.168.173.3 up
D-gep# ifconfig hatm0 192.168.173.4 up
```

Ha feltételezzük, hogy minden gépen a hatm0 az ATM felület neve. Most pedig az A-gep-en állítsuk be az állandó csatornákat. (Itt most feltesszük, hogy az ATM switch-eken mindezt már elvégeztük. A switch kézikönyvében errõl részletesebb leírást is találhatunk.)

```
A-gep# atmconfig natm add 192.168.173.2 hatm0 0 100 llc/snap ubr
A-gep# atmconfig natm add 192.168.173.3 hatm0 0 101 llc/snap ubr
A-gep# atmconfig natm add 192.168.173.4 hatm0 0 102 llc/snap ubr

B-gep# atmconfig natm add 192.168.173.1 hatm0 0 100 llc/snap ubr
B-gep# atmconfig natm add 192.168.173.3 hatm0 0 103 llc/snap ubr
B-gep# atmconfig natm add 192.168.173.4 hatm0 0 104 llc/snap ubr

C-gep# atmconfig natm add 192.168.173.1 hatm0 0 101 llc/snap ubr
C-gep# atmconfig natm add 192.168.173.2 hatm0 0 103 llc/snap ubr
C-gep# atmconfig natm add 192.168.173.4 hatm0 0 105 llc/snap ubr

D-gep# atmconfig natm add 192.168.173.1 hatm0 0 102 llc/snap ubr
D-gep# atmconfig natm add 192.168.173.2 hatm0 0 104 llc/snap ubr
D-gep# atmconfig natm add 192.168.173.3 hatm0 0 105 llc/snap ubr
```

Természetesen nem csak UBR használható, hanem minden más olyan forgalmazási beállítás, amit az ATM kártyáink ismernek. Itt most a forgalmi beállítás nevét a hozzá tartozó konkrét paraméterek követik. Az atmconfig(8) segédprogram használatához így kérhetünk segítséget:

```
# atmconfig help natm add
```

Olvassuk el az atmconfig(8) man oldalát.

Ugyanez a beállítás az /etc/rc.conf állomány használatával is elvégezhetõ. Az A-gep esetében mindez így nézne ki:

```
network_interfaces="lo0 hatm0"
ifconfig_hatm0="inet 192.168.173.1 up"
natm_static_routes="B-gep C-gep D-gep"
route_B-gep="192.168.173.2 hatm0 0 100 llc/snap ubr"
route_C-gep="192.168.173.3 hatm0 0 101 llc/snap ubr"
route_D-gep="192.168.173.4 hatm0 0 102 llc/snap ubr"
```

A CLIP útvonalak pillanatnyi állapota így kérdezhetõ le:

```
A-gep# atmconfig natm show
```

31.13. A Közös cím redundancia protokoll (CARP)

A Közös cím redundancia protokoll (Common Address Redundancy Protocol, avagy CARP) segítségével több gép képes egyazon IP-címen osztozni. Bizonyos konfigurációkban ez a terhelés elosztására (terhelés-kiegyenlítésre) vagy a rendelkezésre állás növelésére (hibatûrésre) alkalmazható. A benne szereplõ gépek akár eltérõ IP-címmel is rendelkezhetnek, ahogy azt majd a példában is láthatjuk.

A CARP támogatásának engedélyezéséhez a FreeBSD rendszermagját a következõ beállítással kell újrafordítanunk:

```
device  carp
```

A CARP által biztosított lehetõségek ezután már elérhetõek, és számos sysctl változón keresztül állíthatóak:

Változó	Leírás
net.inet.carp.allow	A beérkezõ CARP csomagok elfogadása. Alapértelmezés szerint engedélyezett.
net.inet.carp.preempt	Ezzel a beállítással az adott gépen az összes CARP felület leáll, ha közülük bármelyik is mûködésképtelenné válik. Alapértelmezés szerint tiltott.
net.inet.carp.log	A 0 értékkel kikapcsoljuk a naplózást. Az 1 értékkel a rossz CARP csomagok naplózását engedélyezzük. Az ettõl nagyobb értékek esetén pedig a CARP felületek változásait naplózzuk. Az alapértelmezett értéke az 1.

Változó	Leírás
net.inet.carp.arpbalance	Az ARP protokoll segítségével próbálja meg a helyi hálózati forgalmat mentesíteni a terheléstõl. Alapértelmezés szerint tiltott.
net.inet.carp.suppress_preempt	Ez a változó írásvédett, és a megszakítás elnyomásának állapotát mutatja. A megszakítás elnyomható, ha a felület egyik linkje nem mûködik. A 0 érték arra utal, hogy a megszakítást nem nyomták el. Minden probléma növeli ennek a változónak az értékét.

A CARP eszközök maguk az ifconfig paranccsal készíthetõek el:

```
# ifconfig carp0 create
```

Egy valós környezetben az ilyen felületeknek egy VHID néven ismert egyedi azonosítóval kell rendelkezniük. Ez a VHID vagy más néven a virtuális gépazonosító (azaz Virtual Host Identification) fogja a gépünket a hálózat többi elemétõl megkülönböztetni.

31.13.1. A CARP felhasználása a rendelkezésre állás javításában

A CARP használatának egyik módja, ahogy arra már korábban is utaltunk, a szerverek rendelkezésre állásának feljavítása. Ebben a példában három géppel fogunk hibatûrést biztosítani, melyik mindegyike egyedi IP-címmel rendelkezik és ugyanazt a webes tartalmat szolgáltatják. A gépeket egy Round Robin rendszerû (körbejáró) névfeloldással együtt használjuk. A tartalék gépünknek lesz még további két CARP felülete, külön a szerver IP-címeihez tartozó egyes webes tartalmakhoz. Amikor valami meghibásodik, a tartalék szerver átveszi a meghibásodott gép IP-címét. Ilyenkor a hiba teljesen észrevétlen marad a felhasználók számára. A tartalék szerveren a többi szerverrel egyezõ tartalomnak és szolgáltatásoknak kell megjelennie, hogy bármikor át tudja tõlük venni a forgalmat.

A hálózati neveiktõl és a virtuális azonosítóiktól eltekintve a két gépet ugyanúgy kell beállítani. Ebben a példában a gépeket most az a-gep.minta.org és b-gep.minta.org nevekkel láttuk el. Elõször is a CARP beállításához el kell helyeznünk a megfelelõ hivatkozásokat az rc.conf állományban. Az a-gep.minta.org esetében az rc.conf állomány a következõ sorokat tartalmazza:

```
hostname="a-gep.minta.org"
ifconfig_fxp0="inet 192.168.1.3 netmask 255.255.255.0"
cloned_interfaces="carp0"
ifconfig_carp0="vhid 1 pass testpass 192.168.1.50/24"
```

Miközben a b-gep.minta.org az rc.conf állományában ezeket adjuk meg:

```
hostname="b-gep.minta.org"
ifconfig_fxp0="inet 192.168.1.4 netmask 255.255.255.0"
```

```
cloned_interfaces="carp0"
ifconfig_carp0="vhid 2 pass testpass 192.168.1.51/24"
```

Nagyon fontos, hogy az `ifconfig` parancs `pass` paraméterével megadott jelszavak megegyezzenek. A carp eszközök csak a megfelelõ jelszót birtokló gépeket fogadják el. A virtuális gépazonosítónak azonban minden esetben el kell térnie.

A harmadik, `szolgaltato.minta.org` címmel rendelkezõ gépet fogjuk felkészíteni az elõbbi gépek meghibásodására felkészíteni. Ennek a gépnek két carp eszközre lesz szüksége, melyek az egyes gépeket kezelik. Az ehhez illeszkedõ sorok valahogy így fognak kinézni az rc.conf állományban:

```
hostname="szolgaltato.minta.org"
ifconfig_fxp0="inet 192.168.1.5 netmask 255.255.255.0"
cloned_interfaces="carp0 carp1"
ifconfig_carp0="vhid 1 advskew 100 pass testpass 192.168.1.50/24"
ifconfig_carp1="vhid 2 advskew 100 pass testpass 192.168.1.51/24"
```

Két carp eszköz használatával a `szolgaltato.minta.org` képes észlelni és átvenni bármelyik olyan gép IP-címét, amely nem válaszol.

Az alap FreeBSD rendszermag használata esetén *elõfordulhat*, hogy a megszakítás (a "preemption" opció) engedélyezett. Amennyiben így lenne, a `szolgaltato.minta.org` nem fogja minden esetben fogja rendesen visszaadni az IP-címet az eredeti tulajdonosának. Ilyenkor a rendszergazdának kell ezt manuálisan megtennie. Tehát a következõ parancsot kell kiadnia a `szolgaltato.minta.org` gépen:

```
# ifconfig carp0 down && ifconfig carp0 up
```

Ezt az adott géphez tartozó carp felülettel kell megcsinálni.

Innentõl a CARP már teljesen engedélyezhetõ és készen áll a tesztelésre. A teszteléshez vagy a hálózati rendszert kell újraindítani, vagy a gépeket.

További információkat a carp(4) man oldalán találhatunk.

Part V: Függelék

Függelék A: A FreeBSD beszerzése

A.1. CD és DVD kiadók

A.1.1. Kiskereskedelmi dobozos termékek

A FreeBSD beszerezhetõ számos kiskereskedõtõl dobozos termék formájában is (FreeBSD CD-k, egyéb szoftverek és nyomtatott dokumentáció):

- CompUSA
  WWW: http://www.compusa.com/
- Frys Electronics
  WWW: http://www.frys.com/

A.1.2. CD- és DVD-készletek

FreeBSD CD- és DVD-készletek rengeteg helyrõl rendelhetõek:

- FreeBSD Mall, Inc.
  700 Harvest Park Ste F
  Brentwood, CA 94513
  Egyesült Államok
  Telefon: +1 925 240-6652
  Fax: +1 925 674-0821
  e-mail: <info@freebsdmall.com>
  WWW: http://www.freebsdmall.com/
- Dr. Hinner EDV
  St. Augustinus-Str. 10
  D-81825 München
  Németország
  Telefon: (089) 428 419
  WWW: http://www.hinner.de/linux/freebsd.html
- Ikarios
  22-24 rue Voltaire
  92000 Nanterre
  Franciaország
  WWW: http://ikarios.com/form/#freebsd
- JMC Software
  Írország
  Telefon: 353 1 6291282
  WWW: http://www.thelinuxmall.com
- The Linux Emporium
  Hilliard House, Lester Way
  Wallingford
  OX10 9TA

Egyesült Királyság
Telefon: +44 1491 837010
Fax: +44 1491 837016
WWW: http://www.linuxemporium.co.uk/products/bsd/

- Linux+ DVD Magazine
  Lewartowskiego 6
  Warsaw
  00-190
  Lengyelország
  Telefon: +48 22 860 18 18
  e-mail: <editors@lpmagazine.org>
  WWW: http://www.lpmagazine.org/
- Linux System Labs Australia
  21 Ray Drive
  Balwyn North
  VIC - 3104
  Ausztrália
  Telefon: +61 3 9857 5918
  Fax: +61 3 9857 8974
  WWW: http://www.lsl.com.au
- LinuxCenter.Ru
  Galernaya utca, 55
  Szentpétervár
  190000
  Oroszország
  Telefon: +7-812-3125208
  e-mail: <info@linuxcenter.ru>
  WWW: http://linuxcenter.ru/shop/freebsd

A.1.3. Terjesztõk

Ha viszonteladók vagyunk és szeretnénk CD-s FreeBSD termékeket forgalmazni, akkor az alábbi terjesztõk valamelyikével vegyük fel a kapcsolatot:

- Cylogistics
  809B Cuesta Dr., #2149
  Mountain View, CA 94040
  Egyesült Államok
  Telefon: +1 650 694-4949
  Fax: +1 650 694-4953
  e-mail: <sales@cylogistics.com>
  WWW: http://www.cylogistics.com/
- Ingram Micro
  1600 E. St. Andrew Place
  Santa Ana, CA 92705-4926
  Egyesült Államok

Telefon: 1 (800) 456-8000
WWW: http://www.ingrammicro.com/

- Kudzu, LLC
  7375 Washington Ave. S.
  Edina, MN 55439
  Egyesült Államok
  Telefon: +1 952 947-0822
  Fax: +1 952 947-0876
  e-mail: <sales@kudzuenterprises.com>

- LinuxCenter.Kz
  Uszty-Kamenogorszk
  Kazahsztán
  Telefon: +7-705-501-6001
  e-mail: <info@linuxcenter.kz>
  WWW: http://linuxcenter.kz/page.php?page=fr

- LinuxCenter.Ru
  Galernaya utca, 55
  Szentpétervár
  190000
  Oroszország
  Telefon: +7-812-3125208
  e-mail: <info@linuxcenter.ru>
  WWW: http://linuxcenter.ru/freebsd

- Navarre Corp
  7400 49th Ave South
  New Hope, MN 55428
  Egyesült Államok
  Telefon: +1 763 535-8333
  Fax: +1 763 535-0341
  WWW: http://www.navarre.com/

A.2. FTP oldalak

A FreeBSD hivatalos forrásai anonim FTP-n keresztül is elérhetõek különféle tükrözésekrõl. Az ftp://ftp.FreeBSD.org/pub/FreeBSD/ oldal ugyan jó minõségû kapcsolattal rendelkezik és rengeteg felhasználót is enged egyidejûleg kapcsolódni, azonban valószínûleg jobban járunk, ha egy "hozzánk közelebbi" tükrözést választunk (különösen abban az esetben, amikor mi magunk is egy tükrözést akarunk készíteni).

A FreeBSD tükrözések adatbázisában az itt megtalálhatónál sokkal pontosabb leltárt kaphatunk az elérhetõ tükrözésekrõl, mivel közvetlenül a névfeloldás segítségével állapítja meg a szükséges adatokat és nem egy rögzített listát tárol.

Emellett az alábbi tükrözésekrõl a FreeBSD elérhetõ anonim FTP-n keresztül is. Amennyiben az anonim FTP használata mellett döntenénk, igyekezzünk a hozzánk legközelebb levõ szervert használni. Az "Elsõdleges tükrözésekként" feltüntetett oldalak általában a teljes FreeBSD

archívumot tartalmazzák (az összes jelenleg elérhetõ változatot az összes architektúrára), de a környékünkön vagy országunkban elhelyezkedõ tükörszerverekrõl többnyire gyorsabban tudunk majd letölteni. A regionális oldalakon gyakorta csak a népszerûbb architektúrákon futó népszerûbb változatokat találjuk meg, nem a teljes FreeBSD archívumot. Minden szerver elérhetõ anonim FTP-vel, de közülük néhány még további más módszereket is támogat. Az egyes oldalak által ismert konkrét módszereket a nevük után zárójelben közüljük.

Central Servers, Primary Mirror Sites, Armenia, Australia, {mirrors-austria}, Brazil, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hong Kong, Ireland, Japan, Korea, Latvia, Lithuania, Netherlands, New Zealand, Norway, Poland, Russia, Saudi Arabia, Slovenia, South Africa, Spain, Sweden, Switzerland, Taiwan, Ukraine, United Kingdom, United States of America.

(as of UTC)

Central Servers

ftp://ftp.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 / http://ftp.FreeBSD.org/pub/FreeBSD/ / http://ftp.FreeBSD.org/pub/FreeBSD/)

Primary Mirror Sites

In case of problems, please contact the hostmaster <mirror-admin@FreeBSD.org> for this domain.

- ftp://ftp1.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 / http://ftp4.FreeBSD.org/pub/FreeBSD/ / http://ftp4.FreeBSD.org/pub/FreeBSD/)
- ftp://ftp5.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp6.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp7.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp10.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 / http://ftp10.FreeBSD.org/pub/FreeBSD/ / http://ftp10.FreeBSD.org/pub/FreeBSD/)
- ftp://ftp11.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp13.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp14.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp14.FreeBSD.org/pub/FreeBSD/)

Armenia

In case of problems, please contact the hostmaster <hostmaster@am.FreeBSD.org> for this domain.

- ftp://ftp1.am.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp1.am.FreeBSD.org/pub/FreeBSD/ / rsync)

Australia

In case of problems, please contact the hostmaster <hostmaster@au.FreeBSD.org> for this domain.

- ftp://ftp.au.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.au.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.au.FreeBSD.org/pub/FreeBSD/ (ftp)

{mirrors-austria}

In case of problems, please contact the hostmaster <{mirrors-austria-email}> for this domain.

- {mirrors-austria-ftp} (ftp / ftpv6 / {mirrors-austria-ftp-http} / {mirrors-austria-ftp-httpv6})

Brazil

In case of problems, please contact the hostmaster <hostmaster@br.FreeBSD.org> for this domain.

- ftp://ftp2.br.FreeBSD.org/FreeBSD/ (ftp / http://ftp2.br.FreeBSD.org/)
- ftp://ftp3.br.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)
- ftp://ftp4.br.FreeBSD.org/pub/FreeBSD/ (ftp)

Czech Republic

In case of problems, please contact the hostmaster <hostmaster@cz.FreeBSD.org> for this domain.

- ftp://ftp.cz.FreeBSD.org/pub/FreeBSD/ (ftp / ftp://ftp.cz.FreeBSD.org/pub/FreeBSD/ / http://ftp.cz.FreeBSD.org/pub/FreeBSD/ / http://ftp.cz.FreeBSD.org/pub/FreeBSD/ / rsync / rsyncv6)
- ftp://ftp2.cz.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp2.cz.FreeBSD.org/pub/FreeBSD/)

Denmark

In case of problems, please contact the hostmaster <staff@dotsrc.org> for this domain.

- ftp://ftp.dk.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 / http://ftp.dk.FreeBSD.org/pub/FreeBSD/ / http://ftp.dk.FreeBSD.org/pub/FreeBSD/)

Estonia

In case of problems, please contact the hostmaster <hostmaster@ee.FreeBSD.org> for this domain.

- ftp://ftp.ee.FreeBSD.org/pub/FreeBSD/ (ftp)

Finland

In case of problems, please contact the hostmaster <hostmaster@fi.FreeBSD.org> for this domain.

- ftp://ftp.fi.FreeBSD.org/pub/FreeBSD/ (ftp)

France

In case of problems, please contact the hostmaster <hostmaster@fr.FreeBSD.org> for this domain.

- ftp://ftp.fr.FreeBSD.org/pub/FreeBSD/ (ftp)

- ftp://ftp1.fr.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp1.fr.FreeBSD.org/pub/FreeBSD/ / rsync)
- ftp://ftp3.fr.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp6.fr.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp6.fr.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)
- ftp://ftp7.fr.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp8.fr.FreeBSD.org/pub/FreeBSD/ (ftp)

Germany

In case of problems, please contact the hostmaster <de-bsd-hubs@de.FreeBSD.org> for this domain.

- ftp://ftp.de.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp1.de.FreeBSD.org/freebsd/ (ftp / http://www1.de.FreeBSD.org/freebsd/ / rsync://rsync3.de.FreeBSD.org/freebsd/)
- ftp://ftp2.de.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp2.de.FreeBSD.org/pub/FreeBSD/ / rsync)
- ftp://ftp4.de.FreeBSD.org/FreeBSD/ (ftp / http://ftp4.de.FreeBSD.org/pub/FreeBSD/)
- ftp://ftp5.de.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp7.de.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp7.de.FreeBSD.org/pub/FreeBSD/)

Greece

In case of problems, please contact the hostmaster <hostmaster@gr.FreeBSD.org> for this domain.

- ftp://ftp.gr.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.gr.FreeBSD.org/pub/FreeBSD/ (ftp)

Hong Kong

ftp://ftp.hk.FreeBSD.org/pub/FreeBSD/ (ftp)

Ireland

In case of problems, please contact the hostmaster <hostmaster@ie.FreeBSD.org> for this domain.

- ftp://ftp3.ie.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)

Japan

In case of problems, please contact the hostmaster <hostmaster@jp.FreeBSD.org> for this domain.

- ftp://ftp.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp5.jp.FreeBSD.org/pub/FreeBSD/ (ftp)

- ftp://ftp6.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp7.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp8.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp9.jp.FreeBSD.org/pub/FreeBSD/ (ftp)

Korea

In case of problems, please contact the hostmaster <hostmaster@kr.FreeBSD.org> for this domain.

- ftp://ftp.kr.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)
- ftp://ftp2.kr.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp2.kr.FreeBSD.org/pub/FreeBSD/)

Latvia

In case of problems, please contact the hostmaster <hostmaster@lv.FreeBSD.org> for this domain.

- ftp://ftp.lv.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp.lv.FreeBSD.org/pub/FreeBSD/)

Lithuania

In case of problems, please contact the hostmaster <hostmaster@lt.FreeBSD.org> for this domain.

- ftp://ftp.lt.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp.lt.FreeBSD.org/pub/FreeBSD/)

Netherlands

In case of problems, please contact the hostmaster <hostmaster@nl.FreeBSD.org> for this domain.

- ftp://ftp.nl.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp.nl.FreeBSD.org/os/FreeBSD/ / rsync)
- ftp://ftp2.nl.FreeBSD.org/pub/FreeBSD/ (ftp)

New Zealand

- ftp://ftp.nz.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp.nz.FreeBSD.org/pub/FreeBSD/)

Norway

In case of problems, please contact the hostmaster <hostmaster@no.FreeBSD.org> for this domain.

- ftp://ftp.no.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)

Poland

In case of problems, please contact the hostmaster <hostmaster@pl.FreeBSD.org> for this domain.

- ftp://ftp.pl.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp2.pl.FreeBSD.org

Russia

In case of problems, please contact the hostmaster <hostmaster@ru.FreeBSD.org> for this domain.

- ftp://ftp.ru.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp.ru.FreeBSD.org/FreeBSD/ / rsync)
- ftp://ftp2.ru.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp2.ru.FreeBSD.org/pub/FreeBSD/ / rsync)
- ftp://ftp5.ru.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp5.ru.FreeBSD.org/pub/FreeBSD/ / rsync)
- ftp://ftp6.ru.FreeBSD.org/pub/FreeBSD/ (ftp)

Saudi Arabia

In case of problems, please contact the hostmaster <ftpadmin@isu.net.sa> for this domain.

- ftp://ftp.isu.net.sa/pub/ftp.freebsd.org (ftp)

Slovenia

In case of problems, please contact the hostmaster <hostmaster@si.FreeBSD.org> for this domain.

- ftp://ftp.si.FreeBSD.org/pub/FreeBSD/ (ftp)

South Africa

In case of problems, please contact the hostmaster <hostmaster@za.FreeBSD.org> for this domain.

- {mirrors-south-africa-ftp} (ftp)
- ftp://ftp2.za.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.za.FreeBSD.org/pub/FreeBSD/ (ftp)

Spain

In case of problems, please contact the hostmaster <hostmaster@es.FreeBSD.org> for this domain.

- ftp://ftp.es.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp.es.FreeBSD.org/pub/FreeBSD/)
- ftp://ftp3.es.FreeBSD.org/pub/FreeBSD/ (ftp)

Sweden

In case of problems, please contact the hostmaster <hostmaster@se.FreeBSD.org> for this domain.

- ftp://ftp.se.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.se.FreeBSD.org/pub/FreeBSD/ (ftp / rsync://ftp2.se.FreeBSD.org/)
- ftp://ftp3.se.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.se.FreeBSD.org/pub/FreeBSD/ (ftp / ftp://ftp4.se.FreeBSD.org/pub/FreeBSD/ / http://ftp4.se.FreeBSD.org/pub/FreeBSD/ / http://ftp4.se.FreeBSD.org/pub/FreeBSD/ / rsync://ftp4.se.FreeBSD.org/pub/FreeBSD/ / rsync://ftp4.se.FreeBSD.org/pub/FreeBSD/)
- ftp://ftp6.se.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp6.se.FreeBSD.org/pub/FreeBSD/)

Switzerland

In case of problems, please contact the hostmaster <hostmaster@ch.FreeBSD.org> for this domain.

- ftp://ftp.ch.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp.ch.FreeBSD.org/pub/FreeBSD/)

Taiwan

In case of problems, please contact the hostmaster <hostmaster@tw.FreeBSD.org> for this domain.

- ftp://ftp.ch.FreeBSD.org/pub/FreeBSD/ (ftp / ftp://ftp.tw.FreeBSD.org/pub/FreeBSD/ / rsync / rsyncv6)
- ftp://ftp2.tw.FreeBSD.org/pub/FreeBSD/ (ftp / ftp://ftp2.tw.FreeBSD.org/pub/FreeBSD/ / http://ftp2.tw.FreeBSD.org/pub/FreeBSD/ / http://ftp2.tw.FreeBSD.org/pub/FreeBSD/ / rsync / rsyncv6)
- ftp://ftp4.tw.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp5.tw.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp6.tw.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp6.tw.FreeBSD.org/ / rsync)
- ftp://ftp7.tw.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp8.tw.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp11.tw.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp11.tw.FreeBSD.org/FreeBSD/)
- ftp://ftp12.tw.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp13.tw.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp14.tw.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp15.tw.FreeBSD.org/pub/FreeBSD/ (ftp)

Ukraine

- ftp://ftp.ua.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp.ua.FreeBSD.org/pub/FreeBSD/)
- ftp://ftp6.ua.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp6.ua.FreeBSD.org/pub/FreeBSD / rsync://ftp6.ua.FreeBSD.org/FreeBSD/)
- ftp://ftp7.ua.FreeBSD.org/pub/FreeBSD/ (ftp)

United Kingdom

In case of problems, please contact the hostmaster <hostmaster@uk.FreeBSD.org> for this domain.

- ftp://ftp.uk.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.uk.FreeBSD.org/pub/FreeBSD/ (ftp / rsync://ftp2.uk.FreeBSD.org/ftp.freebsd.org/pub/FreeBSD/)
- ftp://ftp3.uk.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.uk.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp5.uk.FreeBSD.org/pub/FreeBSD/ (ftp)

United States of America

In case of problems, please contact the hostmaster <hostmaster@us.FreeBSD.org> for this domain.

- ftp://ftp1.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.us.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 / http://ftp4.us.FreeBSD.org/pub/FreeBSD/ / http://ftp4.us.FreeBSD.org/pub/FreeBSD/)
- ftp://ftp5.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp6.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp8.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp10.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp11.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp13.us.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp13.us.FreeBSD.org/pub/FreeBSD/ / rsync)
- ftp://ftp14.us.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp14.us.FreeBSD.org/pub/FreeBSD/)
- ftp://ftp15.us.FreeBSD.org/pub/FreeBSD/ (ftp)

A.3. Anonim CVS

A.3.1. Bevezetés

Az anonim CVS (vagy más néven *anoncvs*) a FreeBSD-hez mellékelt CVS-es segédprogramok által nyújtott olyan lehetõség, amivel távoli CVS repositorykkal tudunk szinkronizálni. Több más dolog mellett lehetõvé teszi a FreeBSD felhasználói számára, hogy kiemelt jogosultságok nélkül képesek legyenek olvasással kapcsolatos CVS mûveleteket végrehajtani a FreeBSD Projekt hivatalos anoncvs szerverein. A használatához egyszerûen csak a kiválasztott anoncvs szervert kell beállítani a `CVSROOT` környezeti változó értékének, ahol aztán a `cvs login` parancsnak a szerver által ismert "anoncvs" jelszót kell megadni. Ezután a cvs(1) paranccsal a többi CVS szerverhez hasonlóan lehetõségünk nyílik hozzáférni.

A `cvs login` parancs a bejelentkezésekhez szükséges jelszavakat a `HOME` könyvtárunkban levõ .cvspass állományban tárolja. Ha ez az állomány nem létezik, akkor a `cvs login` elsõ használatakor hibát kapunk. Ilyenkor csak hozzunk létre egy üres .cvspass állományt, majd próbálkozzunk újra.

Habár azt mondhatnánk, hogy a CVSup és az *anoncvs* lényegében egyazon feladatot oldják meg, mind a két esetben léteznek olyan kompromisszumok, amelyek befolyásolhatják a felhasználó választását a két szinkronizációs módszer között. Dióhéjban ezt úgy tudnánk összefoglalni, hogy a CVSup a hálózati erõforrásokat hatékonyabban kihasználja és kettejük közül ez a fejlettebb, azonban ennek meg kell fizetnünk az árát. A CVSup használatához elõször ugyanis telepítenünk kell és be kell állítanunk egy speciális klienst, illetve az adatokat a CVSup által *gyûjteményeknek* (collection) nevezett, viszonylag nagy méretû egyeségekben érhetjük el.

Ezzel szemben az anoncvs használata során a megfelelõ CVS modul nevének felhasználásával tetszõlegesen megvizsgálhatunk önálló állományokat vagy akár programokat (mint az `ls` vagy a `grep`). Természetesen az anoncvs segítségével csupán az olvasást igénylõ CVS mûveleteket

végezhetjük el, ezért ha a FreeBSD Projekt keretein belül fejleszteni is szeretnénk, akkor inkább érdemes a CVSup alkalmazást választani.

A.3.2. Az anonim CVS használata

A cvs(1) parancsot nagyon könnyû beállítani az anonim CVS repositoryk használatához, hiszen mindössze annyit kell tennünk, hogy a `CVSROOT` környezeti változó értékének megadjuk a FreeBSD Projekt valamelyik *anoncvs* szerverét. Ezen sorok írásának pillanatában a következõ szerverek érhetõek el:

- *Franciaország*: :pserver:anoncvs@anoncvs.fr.FreeBSD.org:/home/ncvs (pserver módban a jelszó "anoncvs", az SSH pedig nincs jelszó)
- *Tajvan*: :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs (pserver módban a `cvs login` használatával tetszõleges jelszó megadható, az SSH esetén pedig nincs jelszó)

  ```
  SSH2 HostKey: 1024 02:ed:1b:17:d6:97:2b:58:5e:5c:e2:da:3b:89:88:26
  /etc/ssh/ssh_host_rsa_key.pub
  SSH2 HostKey: 1024 e8:3b:29:7b:ca:9f:ac:e9:45:cb:c8:17:ae:9b:eb:55
  /etc/ssh/ssh_host_dsa_key.pub
  ```

- *Egyesült Államok*: anoncvs@anoncvs1.FreeBSD.org:/home/ncvs (csak SSH v2 - nincs jelszó)

  ```
  SSH2 HostKey: 2048 53:1f:15:a3:72:5c:43:f6:44:0e:6a:e9:bb:f8:01:62
  /etc/ssh/ssh_host_dsa_key.pub
  ```

Mivel a CVS használatával "kikérhetjük" (check out) tulajdonképpen a FreeBSD forrásainak akármelyik eddigi (vagy majd ezután keletkezõ) változatát, érdemes megismerkednünk a cvs(1) által alkalmazott revízió (revision) (az `-r` opcióval állítható) fogalmával és a FreeBSD Projekt repositoryjain belül engedélyezett értékeivel.

Címkéket (tag) két esetben használhatunk: a revíziók és az ágak esetén. A revíziós címkék mindig egy adott revízióra hivatkoznak, ami állandóan ugyanazt jelenti. Ezzel szemben az ágak címkéi a fejlesztés adott irányú menetének az adott pillanatban legfrissebb revízióját hivatkozzák. Mivel az ágak címkéi nem egy adott revízióra vonatkoznak, ezért elmondhatjuk róluk, hogy naponta változik a jelentésük.

Az CVS címkék tartalmazza a felhasználók számára fontos revíziós címkéket. Ezek azonban nem igazak a Portgyûjteményre, mivel a Portgyûjteménynek nincs egyszerre több fejlesztési iránya.

Egy ág címkéjének megadásával általában az adott irányhoz tartozó állományok legfrissebb változatát kapjuk meg. Ha viszont az állományok egy korábbi változatára lenne szükségünk, akkor a `-D dátum` opció megadásával meg tudjuk adni annak idõpontját. Errõl részletesebben a cvs(1) man oldalán olvashatunk.

A.3.3. Példák

Habár a továbbhaladáshoz mindenképpen javasoljuk a cvs(1) man oldalának részletes

áttanulmányozását, mutatunk néhány gyors példát az anonim CVS használatának tömör illusztrálására:

Példa 42. Valami (az ls(1)) kikérése a -CURRENT ágból

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
"Jelszóként" ezután bármit megadhatunk.
% cvs co ls
```

Példa 43. Az src/ fa kikérése SSH-n keresztül

```
% cvs -d anoncvs@anoncvs1.FreeBSD.org:/home/ncvs co src
The authenticity of host 'anoncvs1.freebsd.org (216.87.78.137)' can't be
established.
DSA key fingerprint is 53:1f:15:a3:72:5c:43:f6:44:0e:6a:e9:bb:f8:01:62.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'anoncvs1.freebsd.org' (DSA) to the list of known
hosts.
```

Példa 44. Az ls(1) 6-STABLE ágban szereplõ változatának kikérése

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
Amikor kéri,  "jelszóként" bármit megadhatunk.
% cvs co -rRELENG_6 ls
```

Példa 45. Az ls(1) változásainak (Unified Diff formátumú) listázása

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
Itt "jelszóként" bármit megadhatunk.
% cvs rdiff -u -rRELENG_5_3_0_RELEASE -rRELENG_5_4_0_RELEASE ls
```

Példa 46. A használható modulok nevének kiderítése

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
Ezután  "jelszóként" bármit megadhatunk.
% cvs co modules
% more modules/modules
```

A.3.4. Egyéb helyek

A következõ helyeken találhatunk még hasznos információkat a CVS használatáról:

- A CVS bemutatása (forrás: Kaliforna Állami Mûszaki Egyetem).
- A CVS honlapja, a CVS fejlesztésével és alkalmazásával foglalkozó közösség oldala.
- A CVSweb a FreeBSD Projekt által használt CVS rendszerének webes felülete.

A.4. A CTM használata

A CTM használatáva a távoli könyvtárakat tudunk egy központi változattal szinkronban tartani. Eredetileg a FreeBSD forrásaihoz fejlesztették ki, de idõvel mások más célokra is alkalmasnak találhatják majd. Az eltérések (delták) feldolgozásával kapcsolatban kevéske dokumentáció áll rendelkezésre, ezért a ctm-users-desc levelezési listát érdemes felkeresni, ha többet szeretnénk megtudni a CTM egyéb célú alkalmazásairól.

A.4.1. Miért használnánk a CTM-et?

A CTM segítségével a FreeBSD forrásainak helyi másolatát hozhatjuk létre. A források több különbözõ "kivitelben" is hozzáférhetõek. A CTM minden esetben képes eleget tenni az igényeinknek, akár az egész CVS fát, akár annak egy részét kívánjuk csak figyelemmel követni. Ha netalán FreeBSD fejlesztõk lennénk, és híján vagyunk vagy éppen gyenge TCP/IP kapcsolattal rendelkezünk, esetleg egyszerûen csak automatikusan értesülni szeretnénk a változásokról, a CTM-et nekünk találták ki. A leggyorsabban fejlõdõ ágakból is naponta legfeljebb három deltát fogunk kapni, azonban érdemes megfontolni a változások automatikus elküldését levélben. A szükséges frissítések méretét mindig igyekszünk minimalizálni. Ez egyébként általában alig 5 KB, de néha (tízbõl egyszer) elõfordul, hogy 10 és 50 KB között van, és idõnként 100 KB vagy afeletti mennyiségû frissítés is érkezhet.

Amikor a fejlesztõk által használt forrásokat töltjük le, magunknak kell gondoskodnunk a menet közben felmerülõ különbözõ problémák megoldásáról. Ez kiváltképp igaz abban az esetben, amikor az aktuális, vagy hivatalos nevén "CURRENT" ágat követjük. Mielõtt azonban egy ilyenbe belevágnánk, érdemes fellapozni a FreeBSD legfrissebb változatának használatáról szóló fejezetet.

A.4.2. Mire van szükségünk a CTM használatához?

A mûködéshez két komponens szükségeltetik: a CTM kliensprogramja és hozzá a kezdeti delták (amivel majd letöltjük a "CURRENT" forrásait).

A CTM program már a 2.0 kiadástól kezdve a FreeBSD része, és a források között a /usr/src/usr.sbin/ctm könyvtárban találjuk meg (amennyiben felraktuk).

A CTM mûködéséhez kellõ "deltákat" két módon, FTP-n vagy e-mailen keresztül szerezhetjük be. Ha el tudunk érni interneten levõ FTP oldalakat, akkor az alábbi FTP helyeken találunk a CTM-hez használható adatokat:

ftp://ftp.FreeBSD.org/pub/FreeBSD/CTM/

valamint lásd a tükrözéseket.

FTP-n keresztül lépjünk be a könyvtárba, töltsük le a README nevû állományt és kövessük a benne szereplõ utasításokat.

Ha viszont e-mailen keresztül akarjuk megszerezni a deltákat:

Iratkozzunk fel a CTM terjesztési listáinak egyikére. A ctm-cvs-cur-desc lista az egész CVS-fát, míg a ctm-src-cur-desc a fõ fejlesztési ágat teszi elérhetõvé. A CTM 4-STABLE src branch distribution levelezési lista a 4.X kiadásaihoz ágakat tartalmazza, és így tovább. (Ha nem tudjuk, hogyan kell feliratkozni egy levelezési listára, akkor kattintsunk a lista nevére vagy kövessük a https://lists.freebsd.org linket, majd kattintsunk arra a listára, ahova fel akarunk iratkozni. Ezen az oldalon az összes, a feliratkozáshoz nélkülözhetetlen információnak szerepelnie kell.)

Miután elkezdenek megérkezni a CTM-frissítéseket tartalmazó levelek, a tartalmukat a `ctm_rmail` programmal tudjuk kicsomagolni és felhasználni. Az /etc/aliases állományba akár közvetlenül is beírhatjuk a `ctm_rmail` programot, és ezzel a önállósítani tudjuk a levélben érkezõ frissítések feldolgozását. A `ctm_rmail` man oldalán olvashatjuk ennek részleteit.

Nem számít, milyen módon jutunk hozzá a CTM által használt deltákhoz, minden esetben fel kell iratkoznunk a ctm-announce-desc levelezési listára. Az elkövetkezendõkben ez lesz az egyetlen hely, ahová a CTM rendszer mûködtetésével kapcsolatos bejelentések beküldésre kerülnek. A feliratkozáshoz kattinsunk a fenti lista nevére és kövessük a mellette szereplõ utasításokat.

A.4.3. A CTM elsõ használata

Mielõtt nekilátnánk a CTM-hez tartozó delták használatának, elõször el kell jutnunk egy kiindulási ponthoz, ahonnan majd létre tudjuk hozni a rákövetkezõ deltákat.

Ehhez elsõként vegyük számba, pontosan mink is van. Általában mindenki egy "üres" könyvtárral kezd. Ilyenkor egy kezdeti "Empty" (mint "üres") elnevezésû deltával tudjuk megkezdeni az CTM által ismert fa szinkronizálását. Erre a célra lesznek majd szintén alkalmasak a "megkezdett" delták is, amelyek valamikor a CD-re fognak felkerülni.

Mivel a fák maguk több tíz megabyte-nyi méretûek, ezért érdemes inkább valami kéznél levõ eszközzel megkezdeni a folyamatot. Ha van -RELEASE verziójú CD-nk, akkor másoljuk le róla és bontsuk ki a kiinduláskénti használt forrásokat. Ezzel jelentõs mennyiségû adat átvitelét takaríthatjuk meg.

A "kezdõ" deltákat könnyen megismerjük a szám után `X` karakterrel leválasztott nevükrõl (például src-cur.3210XEmpty.gz). Az `X` után szereplõ megnevezés a kezdeti "kiindulás" (seed) fokának felel meg. Az Empty egy üres könyvtárra utal. A szabályok szerint az `Empty` állapotból 100 deltánként jön létre újabb (kiindulásra alkalmas) alapváltozat. Ezek azonban nagyon nagyok is lehetnek. A 70 vagy 80 megabyte-os `gzip`pel csomagolt adatok gyakoriak az XEmpty delták esetén.

Miután kiválasztottuk a számunkra megfelelõ alapváltozatot, szükségünk lesz a tõle nagyobb sorszámú összes deltára is.

A.4.4. A CTM használata a hétköznapokban

A delták felhasználásához egyszerûen csak ennyit kell tennünk:

```
# cd /ahol/tárolni/akarjuk/az/adatokat
# ctm -v -v /ahol/tároljuk/a/deltákat/src-xxx.*
```

A CTM képes értelmezni a `gzip` által csomagolt adatokat, ezért nincs szükség a delták elõzetes kitömörítésére, amivel tárhelyet tudunk spórolni.

Hacsak nem tekinti tökéletesen biztonságosnak az egész folyamatot, akkor a CTM nem fog módosítani a fán. A deltákat a CTM `-c` kapcsolójával is ellenõrizhetjük, aminek során egyáltalán nem fog módosulni a forrásfa. Ekkor egyszerûen csak ellenõrzi a delták sértetlenségét és megnézi, hogy minden rendben zajlana-e az alkalmazásuk során.

A CTM-nek vannak még további kapcsolói is, melyekrõl bõvebben a man oldalakból és a forráskódokból tájékozódhatunk.

Most már minden megvan, ami kellhet. Amikor kapunk egy újabb deltát, a forrásaink frissítéséhez csak futtassuk át a CTM-en.

Ne töröljük le azokat a deltákat, melyeket nehezen tudtunk letölteni. Helyette érdemes inkább megtartani ezeket arra az esetre, ha valami rossz történne. Még ha csak floppylemezek is állnak rendelkezésünkre, mindenképpen másoljuk le ezeket az `fdwrite` paranccsal.

A.4.5. A saját változtatásaink megtartása

Fejlesztõként biztosan szeretnénk kísérletezni és állományokat megváltoztatni a forrásfában. A CTM a helyben elkövetett változtatásokat csak korlátozottan támogatja: az ize nevû állomány meglétének vizsgálata elõtt az ize.ctm állományt fogja keresni. Ha létezik, akkor a CTM az ize helyett ezen fog dolgozni.

Ezzel a viselkedéssel nyerjük a saját változtatásaink megtartásának egyszerû módját: csak másoljuk le .ctm kiterjesztéssel a módosítani tervezett állományokat. Ezután már szabadon módosíthatjuk a forrásokat, miközben a CTM a .ctm kiterjesztésû állományokat folyamatosan szinkronban tartja.

A.4.6. A CTM egyéb érdekes beállításai

Derítsük ki pontosan miket is fog érinteni a frissítés

A CTM által a forrásokon elvégzendõ változtatások listáját az `-l` kapcsolóval kérdezhetjük le.

Ez akkor esik kézre, ha szeretnénk feljegyezni a bekövetkezõ változásokat, vagy bármilyen módon elõ- vagy utófeldolgozni a módosított állományokat, esetleg szimplán elõvigyázatosak akarunk lenni.

Biztonsági másolat készítése a frissítés elõtt

Néha egyszerûen csak szeretnénk az összes érintett állományról biztonsági másolatot készíteni a

CTM által elvégzett frissítés elõtt.

A `-B mentés-állomány` beállítás megadásával az adott CTM delta által módosítandó összes állomány tárolásra kerül a *mentés-állomány* nevû állományba.

A frissíthetõ állományok korlátozása

Egyes esetekben érdekünkben állhat leszûkíteni a CTM által eszközölt frissítések hatáskörét, vagy egyszerûen csak néhány állomány szinkronizálására van szükségünk.

A CTM számára feldolgozható állományok listáját reguláris kifejezés formájában az `-e` és `-x` opciók mentén határozhatjuk meg.

Például ha a lib/libc/Makefile állomány az összegyûjtött CTM delták szerinti legfrissebb verziójához kívánunk hozzájutni, akkor futtassuk az alábbi parancsot:

```
# cd /akarhova/ahova/ki/akarjuk/bontani/
# ctm -e '^lib/libc/Makefile' ~ctm/src-xxx.*
```

A CTM deltákban megadott minden egyes állomány esetén az `-e` az `-x` opciók a parancssorban történt megadásuk sorrendjében kerülnek feldolgozásra. Egy állományt kizárólag csak akkor dolgoz fel a CTM, ha az az `-e` és `-x` opciók kiértékelése után is indokolt.

A.4.7. További tervek a CTM-mel kapcsolatban

Rengeteg van:

- Valamiféle hitelesítés bevezetése a CTM rendszerbe, amivel észlelhetõek a meghamisított CTM-frissítések.
- A CTM beállításainak letisztázása, mivel eléggé megtévesztõek és nehézkesen használhatóak.

A.4.8. Egyebek

Léteznek delták a `portok` gyûjteményéhez is, azonban még nem mutatkozott túlzottan nagy érdeklõdés irántuk.

A.4.9. CTM tükrözések

A CTM/FreeBSD anonim FTP-n keresztül elérhetõ az alábbi tükröoldalak valamelyikérõl. Amennyiben ezen a módon kívánjuk letölteni a CTM rendszerhez tartozó állományokat, elõször próbálkozzunk a hozzánk legközelebb levõ szerverrel.

Ha bármilyen gond merülne fel, értesítsük a ctm-users-desc levelezési listát.

Kalifornia, Bay Area (hivatalos forrás)

- ftp://ftp.FreeBSD.org/pub/FreeBSD/development/CTM/

Dél-Afrika (a korábbi delták biztonsági másolatai)

- ftp://ftp.za.FreeBSD.org/pub/FreeBSD/CTM/

Tajvan/R.O.C.

- ftp://ctm.tw.FreeBSD.org/pub/FreeBSD/development/CTM/
- ftp://ctm2.tw.FreeBSD.org/pub/FreeBSD/development/CTM/
- ftp://ctm3.tw.FreeBSD.org/pub/FreeBSD/development/CTM/

Ha nem találtunk volna hozzánk közel esõ tükrözést, vagy ha talált tükör nem elég friss, akkor próbálkozzunk egy olyan keresõmotor használatával, mint például az alltheweb.

A.5. A CVSup használata

A.5.1. Bevezetés

A CVSup távoli szervereken található központi repositorykban levõ forrásfák terjesztésére és a rajtuk keresztüli frissítésre alkalmas programcsomag. A FreeBSD forrásait egy CVS repositoryban tartják karban Kaliforniában egy fejlesztéseket tároló központi számítógépen. A CVSup segítségével a FreeBSD felhasználói könnyen szinkronban tudják vele tartani a saját forrásaikat.

A CVSup az ún. *lehúzással* frissít. Ilyenkor a kliensek csak akkor kérnek a szervertõl frissítéseket, amikor szükségük van rá, miközben a szerver passzívan várja a frissítési kérelmeket. Ennek megfelelõen tehát minden esetben a kliens kezdeményezi a frissítést, a szerver pedig önmagától sosem küld ilyeneket kéretlenül. A felhasználóknak így vagy maguknak kell meghívniuk a CVSup kliensét, vagy a frissítések rendszeres automatikus letöltéséhez be kell állítaniuk a `cron` rendszerprogramot.

A CVSup kifejezés ebben az írásmódban az egész programcsomagra utal. Fõ alkotórészei a a felhasználó gépén futó `cvsup` nevû kliens, és a FreeBSD tükröoldalain futó `cvsupd` nevû szerver.

A FreeBSD dokumentációjának és levelezési listáinak fürkészése során rengeteg hivatkozást találhatunk egy sup nevû alkalmazásra. A sup a CVSup elõdje volt, és hasonló célokat szolgált. A CVSup használat tekintetében nagyon hasonlít a sup-hoz, és ami azt illeti, a a `sup` konfigurációs állományaival visszafele kompatibilis formátumot használ. Mivel a CVSup sokkal gyorsabb és rugalmasabb, a supot már nem használja a FreeBSD Projekt.

A csup a CVSup C nyelven újraírt változata. Legnagyobb elõnye, hogy gyorsabb és nincs szüksége a Modula-3 nyelv futtató környezetére, ezért azt nem kell a használatához telepíteni. Ráadásul, ha a FreeBSD 6.2 vagy annál késõbbi változatát használjuk, akkor minden további nélkül a rendelkezésünkre áll, hiszen az alaprendszer része. A FreeBSD korábbi verzióinak alaprendszerei ugyan nem tartalmazzák a csup(1) parancsot, viszont a net/csup port vagy csomag segítségével pillanatok alatt telepíteni tudjuk. Amennyiben a csup mellett tennénk le a voksunkat, a szakasz fennmaradó részében egyszerûen hagyjuk ki a CVSup telepítésérõl szóló lépéseket és a CVSup hivatkozásait helyettesítsük a csup programmal.

A.5.2. Telepítés

A CVSup telepítésének legegyszerûbb módja a FreeBSD csomaggyûjteményében található

elõrefordított net/cvsup csomag használata. Ha viszont inkább forrásból akarjuk telepíteni a CVSupot, akkor helyette használjuk a net/cvsup portot. De legyünk elõvigyázatosak: a net/cvsup portnak szüksége van a Modula-3 rendszerre, aminek letöltése és lefordítása pedig meglehetõsen sok idõt és tárhelyet igényel.

Ha olyan gépen akarjuk használni a CVSupot, ahol nincs XFree86™, Xorg vagy bármilyen más ilyen szerver, akkor használjuk a net/cvsup-without-gui portot, ami nem tartalmazza a hozzá tartozó grafikus felületet.

Ha a FreeBSD 6.1 vagy korábbi változatain szeretnénk telepíteni a csupot, használjuk a FreeBSD csomaggyûjteményében megtalálható net/csup csomagot. Ha viszont forrásból kívánjuk telepíteni a csup programot, akkor helyette használjuk a net/csup portot.

A.5.3. A CVSup beállítása

A CVSup mûködését a supfile elnevezésû állomány vezérli. A /usr/shared/examples/cvsup/ könyvárban találhatunk néhány példát a supfile állományokra.

A supfile állományban szereplõ információk a CVSup használatával kapcsolatban a következõ kérdéseket válaszolják meg:

- Milyen állományokat akarunk letölteni?
- Milyen verzióikra van szükségünk?
- Honnan akarjuk ezeket beszerezni?
- Hova akarjuk rakni a számítógépünkön?
- Hova akarjuk rakni az állapotot tároló állományokat?

Az imént feltett kérdésekre a következõ szakaszokban összeállítandó supfile segítségével fogunk válaszolni. Ehhez elõször bemutatjuk a supfile formátumú állományok általános szerkezetét.

A supfile állományok szöveget tartalmaznak. A megjegyzések `#` karakterrel kezdõdnek és a sor végéig tartanak. A kizárólag csak megjegyzéseket tartalmazó vagy üres sorok nem kerülnek feldolgozásra.

Az összes többi fennmaradó sorban pedig azokat az állományokat írjuk le, amelyeket a felhasználó le akar tölteni. Az ilyen fajtájú sorok egy "gyûjtemény" (collection) nevével kezdõdnek, ami állományok egy szerver által meghatározott logikai csoportjára utal. A gyûjtemény neve ennek megfelelõen elárulja a szervernek, hogy pontosan milyen állományokra van szükségünk. Ezután következik whitespace-szel elválasztva nulla vagy több mezõ, amelyek a korábban feltett kérdéseinket válaszolják meg rendre. Ezeknek a mezõknek két típusa létezik: a beállításokat és a konkrét értéket tároló mezõk. A beállításokat tároló mezõk különbözõ kulcsszavakat tartalmaznak, például a `delete` (törlés) vagy `compress` (tömörítés). Az értéket tároló mezõk is egy kulcsszóval kezdõdnek, azonban utána közvetlenül egy `=` (egyenlõségjel) jön, amelyet egy második szó követ szorosan. Így például a `release=cvs` pontosan egy ilyen értékmezõ lesz.

Egy supfile általában egynél több gyûjtemény letöltését írja le. Ezért az ilyen állományok felépítésének egyik módja, ha az egyes gyûjteményhez explicite megadjuk a hozzá tartozó mezõket. Azonban így a supfile állományok gyorsan megnövekednek és kényelmetlenné válnak, mivel a

legtöbb gyûjtemény esetén szinte ugyanazokat a mezõket kellene megadnunk. A CVSup az ilyen típusú bonyodalmak elkerülésére egy alapértelmezési megoldást javasol. A `*default` nevû álgyûjteménnyel kezdõdõ sorok segítségével meg tudunk adni olyan beállításokat és értékeket, amelyek az utána következõ gyûjtemények számára alapértelmezésnek fognak számítani a supfile állományban. Az itt megadott alapértelmezések természetesen az egyes gyûjteményekben tetszõleges módon felülbírálhatóak, a mezõk magán a gyûjteményen belüli megadásával. Az állományban az alapértelmezések is megváltoztathatóak vagy bõvíthetõek további `*default` sorok hozzáadásával.

Mindezek tudatában most már megkezdhetjük a FreeBSD-CURRENT ág tartalmának letöltésére és frissen tartására alkalmas supfile állomány összeállítását.

- Milyen állományokat akarunk letölteni?

  A CVSupon keresztül elérhetõ állományok "gyûjteményeknek" hívott nevesített csoportokra bontva érhetõek el. A hivatkozható gyûjtemények leírását a következõ szakaszban találjuk. Ebben a példában most szeretnénk letölteni az egész FreeBSD rendszer forrását. Ezt a `src-all` nevû gyûjteményre hivatkozva érhetjük el. A supfile állományunk létrehozásának elsõ lépéseként soronként egyet megadva felsoroljuk a letölteni kívánt gyûjteményeket (jelen esetünkben csak egyetlen egyet):

  ```
  src-all
  ```

- Milyen verzióikra van szükségünk?

  A CVSup használatával tulajdonképpen a források összes valaha létezett verziójához hozzá tudunk férni. Ez annak köszönhetõ, hogy a cvsupd szerver közvetlenül a CVS repositoryból dolgozik, ami pedig az összes verziót tartalmazza. A `tag=` és `date=` értékmezõk segítségével adhatjuk meg az igényelt verziókat.

  Legyünk óvatosak azonban a `tag=` mezõk helyes megadásával. Egyes címkék ugyanis csak bizonyos állománygyûjtemények esetén élnek. Ha hibás vagy elírt címkét adunk meg, akkor a CVSup törölni fog olyan állományokat, amelyeket valószínûleg nem kellene. A `ports-*` gyûjtemények esetében pedig kifejezetten *csak* a `tag=.` mezõk használhatóak!

  A `tag=` mezõk a tárházban található szimbolikus címkéket nevezik meg. A címkéknek két típusa van: a revíziókhoz és az ágakhoz tartozó címkék. A revíziós címkék mindig egy adott revíziót hivatkoznak, jelentésük állandó. Ezzel szemben az ágak címkéi egy adott fejlesztési ág adott idõpontjában elérhetõ revíziót címkézi. Mivel az ágak címkéi nem egy konkrét revízióra vonatkoznak, ezért akár olyanra is utalhatnak, ami pillanatnyilag még nem is létezik.

  Az CVS címkékben megtalálhatjuk a fontosabb ágak címkéit. A CVSup konfigurációs állományában a címkéket a `tag=` elõtaggal kell bevezetni (így tehát a `RELENG_4` címke hivatkozása `tag=RELENG_4` lesz). Ne felejtsük el, hogy a Portgyûjtemény esetében csak `tag=.` mezõ megadásának van értelme.

  Igyekezzünk pontosan lemásolni a címkék neveit, mivel a CVSup nem képes

> megkülönböztetni az érvényes és az érvénytelen címkéket. Ha véletlen elírjuk a címkét, akkor a CVSup úgy fog viselkedni, mintha olyan érvényes címkére hivatkozhatunk volna, amihez nem tartoznak állományok. Ennek következtében pedig egyszerûen letörli a már meglevõ forrásainkat.

Egy ág címkéjének megadása során általában az adott fejlesztési vonal legfrissebb verzióját kapjuk meg. Ha viszont az adott ág valamelyik korábbi változatára lenne szükségünk, akkor a `date=` értékmezõ felhasználásával meg tudjuk adni a hozzá tartozó dátumot. Ennek mûködésérõl a cvsup(1) man oldala részletesebben értekezik.

A példában mi most a FreeBSD-CURRENT verziót akarjuk letölteni. Ezért a következõ sort tesszük a supfile állományunk elejére:

```
*default tag=.
```

Ha nem adunk meg sem `tag=`, sem pedig `date=` mezõket, akkor egy fontos eset következik be. Ilyenkor ugyanis egy konkrét verzió helyett közvetlenül a szerver CVS repositoryjából kapjuk meg az állományokat, az összes kiegészítõ információjukkal együtt. A fejlesztõk általában ezt a típusú megoldást kedvelik, mivel így a saját rendszerükön is könnyen karban tudnak tartani egy példányt, amiben tudnak keresni a revíziók között és ki tudják kérni akár az állományok korábbi változatait is. Természetesen ennek függvényében jóval több tárhelyre van szükségük.

- Honnan akarjuk ezeket beszerezni?

  A `host=` mezõ beállításával közöljük a `cvsup` klienssel, honnan töltse le a frissítéseket. A CVSup tükrözések közül bármelyik megfelel erre a célra, habár leginkább azt érdemes választani, ami a kibertérben a hozzánk legközelebb esik. A példában most egy kitalált FreeBSD terjesztési oldalt választunk, a `cvsup99.FreeBSD.org`-ot:

  ```
  *default host=cvsup99.FreeBSD.org
  ```

  A CVSup futtatása elõtt tehát ne felejtsük el megváltoztatni ezt a létezõ számítógép hálózati nevére. A `cvsup` futtatásakor a `-h hálózati név` opció megadásával lehetõségünk ennek felülbírálására.

- Hova akarjuk rakni a számítógépünkön?

  A `prefix=` mezõ adja meg a `cvsup` számára, hogy hova tegye a kapott állományokat. A példában a forrásokat közvetlenül a forrásokat tároló központi könyvtárba, a /usr/src könyvtárba tettük. Mivel a src könyvtár neve már hallgatólagosan benne foglaltatik a letöltésre kiválasztott gyûjtemény nevében, ezért itt csak ennyit kell megadnunk:

  ```
  *default prefix=/usr
  ```

- Hova akarjuk rakni az állapotot tároló állományokat?

A CVSup kliens egy "bázisnak" (base) nevezett könyvtárban folyamatosan fenntart bizonyos állományokban állapotokat (status file). Ezek a már letöltött állományok nyilvántartásával segítik a CVSup hatékony munkavégzését. Mi most a szabványos bázist, a /var/db könyvtárat fogjuk használni:

```
*default base=/var/db
```

Amennyiben még nem létezne a bázisként használni kívánt könyvtár, ideje létrehoznunk. A `cvsup` ugyanis egy nem létezõ könyvtár esetén nem lesz hajlandó mûködni.

- További beállítások a supfile állományban:

  Általában még egy sor szokott szerepelni a supfile állományokban:

  ```
  *default release=cvs delete use-rel-suffix compress
  ```

  A `release=cvs` mezõ jelzi, hogy a szervernek a FreeBSD fõ CVS repositoryból kell kikeresnie az információkat. Tulajdonképpen majdnem mindig erről van szó, és az itt megadható többi lehetõség ismertetése most egyébként is meghaladná a szakasz határait.

  A `delete` hatására a CVSup képes lesz állományokat törölni. Mindig érdemes megadnunk, hiszen a CVSup csak így tudja teljes mértékben frissentartani a forrásokat. A CVSup természetesen csak azokat az állományokat igyekszik letörölni, amelyek miatt valóban felelõs. A kóbor állományokat nem fogja bántani.

  A `use-rel-suffix` hatása egy igazi... Rejtély. Ha tényleg érdekel minket a mûködése, lapozzuk fel bátran a cvsup(1) man oldalát. Nyugodtan adjuk meg és különösebben ne törõdjünk vele.

  A `compress` beállítás segítségével a kommunikációs csatornán vándorló adatokat tudjuk gzip-szerû módon tömöríteni. Ha a hálózati kapcsolatunk sebessége meghaladja a 1,5 Mbitet másodpercenként (T1), akkor ezt már nem érdemes használni, viszont minden más esetben lényeges gyorsulást hozhat.

- Összegezzük az eddigieket:

  Íme a példaként összerakott supfile állományunk teljes tartalma:

  ```
  *default tag=.
  *default host=cvsup99.FreeBSD.org
  *default prefix=/usr
  *default base=/var/db
  *default release=cvs delete use-rel-suffix compress

  src-all
  ```

A refuse állomány

Ahogy arról már korábban szó esett, a CVSup *lehúzással* frissít. Ez alapvetõen annyit jelent, hogy feltárcsázunk egy CVSup szervert, aki a következõt mondja nekünk: "A következõket tudod tõlem letölteni...", amire a kliensünk ezt válaszolja: "Rendben, akkor nekem kell ez, ez, ez meg ez." Alapértelmezés szerint a CVSup kliense azokat az állományokat fogja letölteni, amelyeket a konfigurációs állományban szereplõ gyûjtemények és címkék által megneveztünk. Ez azonban nem mindig felel meg az igényeinknek, különösen akkor, amikor a doc, ports vagy www fákat akarjuk letölteni - az emberek többsége ugyanis nem beszél négy vagy öt nyelven, ezért nincs is szükségük a nyelvfüggõ állományok letöltésére. A Portgyûjtemény letöltése során a *ports-all* helyett egyszerûen egyenként is felsorolhatjuk a számunkra érdekes kategóriákat (például *ports-astrology*, *ports-biology* stb). Azonban mivel a doc és a www fákhoz nincsenek nyelvfüggõ gyûjtemények, ezért elõ kell halásznunk a CVSup egyik remek funkcióját, a refuse állományt.

A refuse állománnyal lényegében arra utasítjuk a CVSup alkalmazást, hogy a gyûjteményekbõl ne töltse le az összes állományt. Úgy is fogalmazhatnánk, hogy javaslatára a kliens *visszautasít* (refuse) bizonyos szervertõl érkezõ állományokat. Ezeket a visszautasításokat tároló refuse állományt a bázis/sup/ könyvtárban találhatjuk meg (illetve ha még nincsenek, akkor ide kell rakunk ezeket). Itt a *bázis* a supfile állományban megadott `base=` mezõre utal, ami a példánkban a /var/db könyvtár volt. Ennek megfelelõen tehát a refuse állomány a /var/db/sup/refuse lesz.

A refuse állomány felépítése igen egyszerû: a letölteni nem kívánt állományok és könyvtárak neveit tartalmazza. Például ha az angolul mellett esetleg még beszélünk egy kevés németet is, de nincs szükségünk az angol dokumentáció német fordítására sem, akkor a következõket írjuk a refuse állományba:

```
doc/bn_*
doc/da_*
doc/de_*
doc/el_*
doc/es_*
doc/fr_*
doc/hu_*
doc/it_*
doc/ja_*
doc/mn_*
doc/nl_*
doc/no_*
doc/pl_*
doc/pt_*
doc/ru_*
doc/sr_*
doc/tr_*
doc/zh_*
```

és így tovább a többi nyelvre is (melyeket a FreeBSD CVS repository böngészésével deríthetjük ki).

Ezzel az alkalmas funkcióval a lassú vagy drága internetes kapcsolattal rendelkezõ felhasználók nagyon jól tudnak gazdálkodni, mivel így nem kell letölteniük az egyáltalán nem használt

állományokat. A refuse állományokról és a CVSup más hasonlóan elegáns funkcióiról a saját man oldaláról tudhatunk meg többet.

A.5.4. A CVSup futtatása

Most már készen állunk egy próba frissítés elvégzésére. A parancssorban nem sok mindent kell beírnunk ehhez:

```
# cvsup supfile
```

ahol a supfile a frissen létrehozott supfile állományunk neve lesz. Feltételezve, hogy a parancsot X11 alatt adtunk ki, az `cvsup` erre feldob egy grafikus ablakot néhány gombbal. Nyomjuk meg a **[go]** feliratú gombot és dõljünk hátra.

Mivel a példában a /usr/src könyvtárunk frissítését állítottuk be, az állományok aktualizálásához szükséges jogosultságok biztosításához a `cvsup` programot `root` felhasználóként kell elindítanunk. Teljesen érthetõ, ha egy kicsit izgatottak vagyunk ezekben a pillanatokban, hiszen az elõbb hoztunk létre egy általunk eddig ismeretlen programhoz egy konfigurációs állományt. Ezért megemlítenénk, hogy ilyenkor elõször mindig próbáljuk ki a konfigurációkat, mielõtt azok bármilyen módosítást végeznének a fontos állományainkon. Ehhez hozzunk létre valahol egy üres könyvtárat, majd adjuk meg a parancssorban ennek a nevét:

```
# mkdir /var/tmp/proba
# cvsup supfile /var/tmp/proba
```

Az így megadott könyvtárba kerülnek a frissítés eredményeképpen keletkezõ állományok. A CVSup elõször megvizsgálja a /usr/src könyvtárban található állományokat, viszont egyiküket sem módosítja vagy törli. A frissítések ehelyett a /var/tmp/proba/usr/src könyvtárba fognak kerülni. A CVSup emellett még a báziskönyvtárában tárolt állapotokat sem fogja megváltoztatni. A módosított állományok új változatai a megadott könyvtárba jönnek létre. Mivel a /usr/src könyvtárt ehhez csak olvasni fogjuk, a próba lefuttatásához még `root` felhasználónak sem kell lennünk.

Ha nem használunk X11-et vagy egyszerûen csak nincs szükségünk a grafikus felületre, a parancssorban pár további opció megadásával így is kiadhatjuk a `cvsup` parancsot:

```
# cvsup -g -L 2 supfile
```

A `-g` hatására a CVSup nem hozza be a grafikus felületét. Ha nem talál X11-et, akkor ez természetesen automatikus, de ellenkezõ esetben ezt is meg kell adnunk.

Az `-L 2` megadásával a CVSup az összes elvégzendõ frissítésrõl részletes értesítést ad. A részletességnek három foka van, `-L 0`-tól indulva egészen `-L 2`-ig. Itt az alapértelmezett érték a 0, amivel a hibaüzenetek kivételével egyetlen üzenetet sem kapunk.

Rengeteg egyéb beállítás adható még meg, ezeket a `cvsup -H` kiadásával kérdezhetjük le. A beállítások pontosabb leírását a man oldalon találjuk meg.

Miután elégedetten tapasztaltuk, hogy a frissítés remekül mûködik, a cron(8) segítségével próbáljuk meg az egész folyamatot önmûködövé tenni a CVSup szabályos idõközönkénti futtatásával. Ekkor viszont magától értetõdik, hogy a CVSup számára ne engedjük használni a grafikus felületet.

A.5.5. A CVSup állománygyûjteményei

A CVSup révén elérhetõ állománygyûjtemények egy hierarchikus rendszert alkotnak. Van néhány nagyobb állománygyûjtemény, amelyek kisebb al-állománygyûjteményekre bonthatóak. A nagyobb gyûjtemények letöltése ezért a kisebb algyûjtemények letöltésével egyenlõ. A gyûjtemények közt fennálló hierarchikus rendszer a lentebb szereplõ lista behúzásaiban érhetõ tetten.

A leggyakrabban használt gyûjtemények a `src-all` és a `ports-all` neveket viselik. A többi gyûjteményt általában csak kevesen és csak speciális célokra használják, ezért egyes tükrözéseken nem feltétlenül találjuk meg mindegyiküket.

`cvs-all release=cvs`

A FreeBSD fõ CVS repositoryja, beleértve a titkosításhoz tartozó kódokat is.

`distrib release=cvs`

A FreeBSD terjesztéséhez és tükrözéséhez kapcsolódó állományok.

`doc-all release=cvs`

A FreeBSD kézikönyvének és a többi dokumentáció forrásai. Nem tartalmazza a FreeBSD honlapjának forrásait.

`ports-all release=cvs`

A FreeBSD portgyûjteménye.

Ha nem akarjuk a `ports-all` egészét (vagyis a teljes portfát) frissíteni, csak a lentebb szereplõ egyes algyûjteményeket letölteni, akkor *soha* ne feledkezzünk meg a `ports-base` megadásáról! Amikor valami változik a portok mûködésében, akkor a `ports-base` által képviselt algyûjteményben szereplõ állományokat igen gyorsan elkezdik használni a "valódi" portok. Ezért ha csak a "valódi" portokat frissítjük, amelyek viszont igényt tartanak néhány újabb funkcióra is, akkor könnyen fordítási hibára vagy különbözõ rejtélyes hibaüzenetekbe futhatunk. Emiatt *legeslegelõször* mindig tegyünk róla, hogy a `ports-base` algyûjteményünk a lehetõ legfrissebb legyen.

Ha a ports/INDEX állomány egy saját példányát kívánjuk létrehozni, akkor ahhoz a `ports-all` gyûjteményt (tehát a teljes portfát) le *kell* kérnünk. A ports/INDEX állományt a portfa egy része alapján nem készíthetjük el. Errõl bõvebben lásd a GYIK-ot.

`ports-accessibility release=cvs`

A fogyatékos felhasználókat segítõ szoftverek.

`ports-arabic release=cvs`

Arab nyelvi támogatás.

`ports-archivers release=cvs`

Archiváló eszközök.

`ports-astro release=cvs`

Csillagászathoz tartozó portok.

`ports-audio release=cvs`

Hangtámogatás.

`ports-base release=cvs`

A Portgyûjtemény saját infrastruktúrája - az Mk/, Tools/ és /usr/ports különféle alkönyvtáraiban elhelyezkedõ állományok.

Ne hagyjuk figyelmen kívül a fenti fontos figyelmeztetést sem: ezt az algyûjteményt *mindig* a FreeBSD Portgyûjteményével együtt frissítsük!

`ports-benchmarks release=cvs`

Teljesítménytesztek.

`ports-biology release=cvs`

Biológia.

`ports-cad release=cvs`

Számítógépes tervezõeszközök (CAD).

`ports-chinese release=cvs`

Kínai nyelvi támogatás.

`ports-comms release=cvs`

Kommunikációs szoftverek.

`ports-converters release=cvs`

Karakterkódolások közti átalakítók.

`ports-databases release=cvs`

Adatbázisok.

`ports-deskutils release=cvs`

A számítógép feltalálása elõtt is már létezõ eszközök.

`ports-devel release=cvs`

Fejlesztõeszközök.

`ports-dns release=cvs`

Névfeloldással kapcsolatos szoftverek.

`ports-editors release=cvs`

Szövegszerkesztõk.

`ports-emulators release=cvs`
: Más operációs rendszerek emulátorai.

`ports-finance release=cvs`
: Pénzügyi, gazdasági és hasonló alkalmazások.

`ports-ftp release=cvs`
: FTP kliensek és szerverek.

`ports-games release=cvs`
: Játékok.

`ports-german release=cvs`
: Német nyelvi támogatás.

`ports-graphics release=cvs`
: Grafikus segédeszközök.

`ports-hebrew release=cvs`
: Héber nyelvi támogatás.

`ports-hungarian release=cvs`
: Magyar nyelvi támogatás.

`ports-irc release=cvs`
: IRC-vel kapcsolatos programok.

`ports-japanese release=cvs`
: Japán nyelvi támogatás.

`ports-java release=cvs`
: Java™ segédeszközök.

`ports-korean release=cvs`
: Koreai nyelvi támogatás.

`ports-lang release=cvs`
: Programozási nyelvek.

`ports-mail release=cvs`
: Levelezõ programok.

`ports-math release=cvs`
: Numerikus számításokkal foglalkozó programok.

`ports-mbone release=cvs`
: MBone alkalmazások.

`ports-misc release=cvs`
: Egyéb segédprogramok.

`ports-multimedia release=cvs`
: Multimediás szoftverek.

`ports-net release=cvs`
: Hálózati szoftverek.

`ports-net-im release=cvs`
: Üzenetküldõ (Instant Messaging, IM) szoftverek.

`ports-net-mgmt release=cvs`
: Hálózati karbantartó szoftverek.

`ports-net-p2p release=cvs`
: Egyenrangú (Peer to Peer, P2P) hálózatok.

`ports-news release=cvs`
: USENET hírszoftverek.

`ports-palm release=cvs`
: A Palm™ sorozat szoftveres támogatása.

`ports-polish release=cvs`
: Lengyel nyelvi támogatás.

`ports-ports-mgmt release=cvs`
: A portok és csomagok karbantartását végzõ segédeszközök.

`ports-portuguese release=cvs`
: Portugál nyelvi támogatás.

`ports-print release=cvs`
: Nyomdai programok.

`ports-russian release=cvs`
: Orosz nyelvi támogatás.

`ports-science release=cvs`
: Tudományos programok.

`ports-security release=cvs`
: Biztonsági segédprogramok.

`ports-shells release=cvs`
: Parancsértelmezõk.

`ports-sysutils release=cvs`
: Rendszerprogramok.

`ports-textproc release=cvs`
: Szövegfeldolgozást segítõ eszközök (kivéve az asztali kiadványszerkesztést).

`ports-ukrainian release=cvs`
: Ukrán nyelvi támogatás.

`ports-vietnamese release=cvs`
: Vietnámi nyelvi támogatás.

`ports-www release=cvs`
: A világhálóhoz tartozó szoftverek.

`ports-x11 release=cvs`
: Az X Window System mûködését segítõ portok.

`ports-x11-clocks release=cvs`
: X11 órák.

`ports-x11-drivers release=cvs`
: X11 meghajtók.

`ports-x11-fm release=cvs`
: X11 állománykezelõk.

`ports-x11-fonts release=cvs`
: X11 betûtípusok és a hozzájuk tartozó segédprogramok.

`ports-x11-toolkits release=cvs`
: X11 eszközrendszerek.

`ports-x11-servers release=cvs`
: X11 szerverek.

`ports-x11-themes release=cvs`
: X11 témák.

`ports-x11-wm release=cvs`
: X11 ablakkezelõk.

`projects-all release=cvs`
: A FreeBSD projektek forrásainak repositoryja.

`src-all release=cvs`
: A FreeBSD fontosabb forrásai, a titkosításhoz tartozó kódokkal együtt.

`src-base release=cvs`

A /usr/src könyvtárban levõ egyéb állományok.

`src-bin release=cvs`

Az egyfelhasználós módban használható segédeszközök (/usr/src/bin).

`src-cddl release=cvs`

A CDDL licenc szerint terjesztett segédprogramok és függvénykönyvtárak (/usr/src/cddl).

`src-contrib release=cvs`

A FreeBSD Projekten kívül fejlesztett segédprogramok és függvénykönyvtárak, viszonylag kevés módosítással (/usr/src/contrib).

`src-crypto release=cvs`

A FreeBSD Projekten kívül fejlesztett, titkosítással kapcsolatos segédprogramok és függvénykönyvtárak, viszonylag kevés módosítással (/usr/src/crypto).

`src-eBones release=cvs`

Kerberos és DES (/usr/src/eBones). A FreeBSD jelenlegi változatai nem használják.

`src-etc release=cvs`

A rendszer beállításait tartalmazó állományok (/usr/src/etc).

`src-games release=cvs`

Játékok (/usr/src/games).

`src-gnu release=cvs`

A GPL licenc szerint terjesztett segédprogramok (/usr/src/gnu).

`src-include release=cvs`

(C nyelvi) Header állományok (/usr/src/include).

`src-kerberos5 release=cvs`

A Kerberos5 biztonsági csomag (/usr/src/kerberos5).

`src-kerberosIV release=cvs`

A KerberosIV biztonsági csomag (/usr/src/kerberosIV).

`src-lib release=cvs`

Függvénykönyvtárak (/usr/src/lib).

`src-libexec release=cvs`

Más programok által futtatott rendszerprogramok (/usr/src/libexec).

`src-release release=cvs`

A FreeBSD kiadások elkészítéséhez szükséges állományok (/usr/src/release).

`src-rescue release=cvs`

Statikusan linkelt programok vészhelyzet esetére, lásd rescue(8) (/usr/src/rescue).

`src-sbin release=cvs`

Egyfelhasználós módban használható rendszereszközök (/usr/src/sbin).

`src-secure release=cvs`

Titkosítással foglalkozó függvénykönyvtárak és parancsok (/usr/src/secure).

`src-share release=cvs`

Több rendszer között megosztható állományok (/usr/src/share).

`src-sys release=cvs`

A rendszermag (/usr/src/sys).

`src-sys-crypto release=cvs`

A rendszermagban levõ titkosítással foglalkozó kód (/usr/src/sys/crypto).

`src-tools release=cvs`

A FreeBSD karbantartására való különbözõ segédprogramok (/usr/src/tools).

`src-usrbin release=cvs`

Felhasználói segédprogramok (/usr/src/usr.bin).

`src-usrsbin release=cvs`

Rendszerszintû segédprogramok (/usr/src/usr.sbin).

`www release=cvs`

A FreeBSD Projekt honlapjának forráskódja.

`distrib release=self`

A CVSup szerver saját konfigurációs állományai. A CVSup tükrözései használják.

`gnats release=current`

A GNATS hibanyilvántartó adatbázis.

`mail-archive release=current`

A FreeBSD levelezési listáinak archívuma.

`www release=current`

A FreeBSD Projekt honlapjának generált állományai (de nem a forrásai). A WWW tükrözések használják.

A.5.6. Bõvebb információk

A CVSup részletesebb bemutatását és a hozzá tartozó GYIK-ot A CVSup honlapján találjuk meg.

A CVSup FreeBSD-re vonatkozó tárgyalása a FreeBSD technical discussions levelezési listan

történik. Itt és az FreeBSD announcements levelezési listan jelentik be a szoftver újabb változatait.

A CVSup alkalmazással kapcsolatos kérdéseket és hibajelentéseket illetõen a CVSup GYIK-ot érdemes megnéznünk.

A.5.7. CVSup oldalak

A FreeBSD CVSup szerverei az alábbi oldalakon érhetõek el:

A.6. CVS címkék

Meg kell adnunk egy revízió címkéjét, amikor a cvs vagy CVSup használatával letöltjük vagy frissítjük a forrásokat. A revíziós címkék a FreeBSD egyik fejlesztési irányát vagy egy adott idõpontbeli állapotát hivatkozzák. Az elõbbi egy "ág címkéje", míg az utóbbi pedig egy "kiadás címkéje".

A.6.1. Az ágak címkéi

A `HEAD` kivételével (amely mindig egy érvényes címke) az összes címke csak a src/ fára vonatkozik. A ports/, doc/ és www/ fák nem tartalmaznak ágakat.

HEAD

A fõ fejlesztési ág, avagy a FreeBSD-CURRENT szimbolikus neve. Ha nem adunk meg revíziót, ez lesz az alapértelmezés.

A CVSup számára ezt `.` címke jelzi (itt most nem mondatvégi pontot jelöli, hanem a `.` karaktert).

A CVS számára ez lesz az alapértelmezett érték, ha nem adunk meg konkrét revíziós címkét. Többnyire *nem* túlzottan jó ötlet egy STABLE változatot használó gépen a CURRENT verziójú források kikérése, kivéve hacsak nem ez a szándékunk.

RELENG_8

A FreeBSD-8.X fejlesztési ága, más néven a FreeBSD 8-STABLE

RELENG_8_1

A FreeBSD-8.1 kiadás ága, ahová csak a biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_8_0

A FreeBSD-8.0 kiadás ága, ahová csak a biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_7

A FreeBSD-7.X fejlesztési ága, más néven a FreeBSD 7-STABLE

RELENG_7_3

A FreeBSD-7.3 kiadás ága, ahová csak a biztonsági frissítések és a kritikus hibajavítások

kerülnek.

RELENG_7_2

A FreeBSD-7.2 kiadás ága, ahová csak a biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_7_1

A FreeBSD-7.1 kiadás ága, ahová csak a biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_7_0

A FreeBSD-7.0 kiadás ága, ahová csak a biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_6

A FreeBSD-6.X fejlesztési ága, más néven a FreeBSD 6-STABLE

RELENG_6_4

A FreeBSD-6.4 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_6_3

A FreeBSD-6.3 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_6_2

A FreeBSD-6.2 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_6_1

A FreeBSD-6.1 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_6_0

A FreeBSD-6.0 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_5

A FreeBSD-5.X fejlesztési ág, más néven a FreeBSD 5-STABLE.

RELENG_5_5

A FreeBSD-5.5 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_5_4

A FreeBSD-5.4 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_5_3

A FreeBSD-5.3 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_5_2

A FreeBSD-5.2 és FreeBSD-5.2.1 kiadások ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_5_1

A FreeBSD-5.1 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_5_0

A FreeBSD-5.0 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_4

A FreeBSD-4.X fejlesztési ága, más néven a FreeBSD 4-STABLE.

RELENG_4_11

A FreeBSD-4.11 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_4_10

A FreeBSD-4.10 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_4_9

A FreeBSD-4.9 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_4_8

A FreeBSD-4.8 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_4_7

A FreeBSD-4.7 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_4_6

A FreeBSD-4.6 és FreeBSD-4.6.2 kiadások ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_4_5

A FreeBSD-4.5 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_4_4

A FreeBSD-4.4 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_4_3

A FreeBSD-4.3 kiadás ága, ahová csak biztonsági frissítések és a kritikus hibajavítások kerülnek.

RELENG_3

A FreeBSD-3.X fejlesztési ága, más néven a 3.X-STABLE.

RELENG_2_2

A FreeBSD-2.2.X fejlesztési ága, más néven a 2.2-STABLE. Ez az ág manapság már elavult.

A.6.2. A kiadások címkéi

Ezek a címkék a FreeBSD egyes kiadásainak dátumára hivatkoznak. Egy kiadás elõkészítésének és terjesztésének folyamatáról részleteiben a kiadásokat összefoglaló lapról és a kiadások építésérõl szóló cikkbõl tájékozódhatunk. Az src fában `RELENG_` kezdetû címkéket találunk. A ports és doc

fákban a címkék nevei a `RELEASE` elõtaggal kezdõdnek. Végezetül a www fában nincsenek kiadásokhoz tartozó címkék.

RELENG_8_1_0_RELEASE
: FreeBSD 8.1

RELENG_8_0_0_RELEASE
: FreeBSD 8.0

RELENG_7_3_0_RELEASE
: FreeBSD 7.3

RELENG_7_2_0_RELEASE
: FreeBSD 7.2

RELENG_7_1_0_RELEASE
: FreeBSD 7.1

RELENG_7_0_0_RELEASE
: FreeBSD 7.0

RELENG_6_4_0_RELEASE
: FreeBSD 6.4

RELENG_6_3_0_RELEASE
: FreeBSD 6.3

RELENG_6_2_0_RELEASE
: FreeBSD 6.2

RELENG_6_1_0_RELEASE
: FreeBSD 6.1

RELENG_6_0_0_RELEASE
: FreeBSD 6.0

RELENG_5_5_0_RELEASE
: FreeBSD 5.5

RELENG_5_4_0_RELEASE
: FreeBSD 5.4

RELENG_4_11_0_RELEASE
: FreeBSD 4.11

RELENG_5_3_0_RELEASE
: FreeBSD 5.3

RELENG_4_10_0_RELEASE
FreeBSD 4.10

RELENG_5_2_1_RELEASE
FreeBSD 5.2.1

RELENG_5_2_0_RELEASE
FreeBSD 5.2

RELENG_4_9_0_RELEASE
FreeBSD 4.9

RELENG_5_1_0_RELEASE
FreeBSD 5.1

RELENG_4_8_0_RELEASE
FreeBSD 4.8

RELENG_5_0_0_RELEASE
FreeBSD 5.0

RELENG_4_7_0_RELEASE
FreeBSD 4.7

RELENG_4_6_2_RELEASE
FreeBSD 4.6.2

RELENG_4_6_1_RELEASE
FreeBSD 4.6.1

RELENG_4_6_0_RELEASE
FreeBSD 4.6

RELENG_4_5_0_RELEASE
FreeBSD 4.5

RELENG_4_4_0_RELEASE
FreeBSD 4.4

RELENG_4_3_0_RELEASE
FreeBSD 4.3

RELENG_4_2_0_RELEASE
FreeBSD 4.2

RELENG_4_1_1_RELEASE
FreeBSD 4.1.1

RELENG_4_1_0_RELEASE
FreeBSD 4.1

RELENG_4_0_0_RELEASE
FreeBSD 4.0

RELENG_3_5_0_RELEASE
FreeBSD 3.5

RELENG_3_4_0_RELEASE
FreeBSD 3.4

RELENG_3_3_0_RELEASE
FreeBSD 3.3

RELENG_3_2_0_RELEASE
FreeBSD 3.2

RELENG_3_1_0_RELEASE
FreeBSD 3.1

RELENG_3_0_0_RELEASE
FreeBSD 3.0

RELENG_2_2_8_RELEASE
FreeBSD 2.2.8

RELENG_2_2_7_RELEASE
FreeBSD 2.2.7

RELENG_2_2_6_RELEASE
FreeBSD 2.2.6

RELENG_2_2_5_RELEASE
FreeBSD 2.2.5

RELENG_2_2_2_RELEASE
FreeBSD 2.2.2

RELENG_2_2_1_RELEASE
FreeBSD 2.2.1

RELENG_2_2_0_RELEASE
FreeBSD 2.2.0

A.7. AFS oldalak

A FreeBSD a következõ szerverein érhetõ el AFS:

Svédország

Az állományok a következõ helyen érhetõek el: /afs/stacken.kth.se/ftp/pub/FreeBSD/

```
stacken.kth.se          # Stacken Computer Club, KTH, Svédország
130.237.234.43          #hot.stacken.kth.se
130.237.237.230         #fishburger.stacken.kth.se
130.237.234.3           #milko.stacken.kth.se
```

Karbantartó: ftp@stacken.kth.se

A.8. Rsync oldalak

A most következõ oldalakon a FreeBSD-t érhetjük el az rsync protokollal. Az rsync segédprogram mûködésében leginkább a rcp(1) parancshoz hasonlít, de sokkal több beállítással rendelkezik, és az rsync távoli frissítéseket kezelõ protokollja segítségével csak az állományok csoportjai között levõ eltéréseket küldi át, amivel a hálózaton keresztüli szinkronizáció rendkívül felgyorsítható. Ez olyankor jelent számunkra a legtöbbet, ha a FreeBSD FTP szerverének vagy CVS repositoryjának egyik tükrözését tartjuk karban. Az rsync több operációs rendszerre is elérhetõ, és FreeBSD-n a net/rsync port vagy csomag tartalmazza.

Cseh Köztársaság

rsync://ftp.cz.FreeBSD.org/

Elérhetõ gyûjtemények:

- ftp: a FreeBSD FTP szerverének részleges tükrözése.
- FreeBSD: a FreeBSD FTP szerverének teljes tükrözése.

Hollandia

rsync://ftp.nl.FreeBSD.org/

Elérhetõ gyûjtemények:

- FreeBSD: a FreeBSD FTP szerverének teljes tükrözése.

Oroszország

rsync://ftp.mtu.ru

Elérhetõ gyûjtemények:

- FreeBSD: A FreeBSD FTP szerver teljes tartalma.
- FreeBSD-gnats: A GNATS hibanyilvántartó adatbázis.
- FreeBSD-Archive: A FreeBSD archívumait tároló FTP szerver tükrözése.

Tajvan

rsync://ftp.tw.FreeBSD.org/

rsync://ftp2.tw.FreeBSD.org/

rsync://ftp6.tw.FreeBSD.org/

Elérhetõ gyûjtemények:

- FreeBSD: a FreeBSD FTP szerverének teljes tükrözése.

Egyesült Királyság

rsync://rsync.mirrorservice.org/

Elérhetõ gyûjtemények:

- sites/ftp.freebsd.org: a FreeBSD FTP szerverének teljes tükrözése.

Amerikai Egyesült Államok

rsync://ftp-master.FreeBSD.org/

Ezt a szervert csak az elsõdleges FreeBSD tükrözéseknek szabad használniuk.

Elérhetõ gyûjtemények:

- FreeBSD: a FreeBSD FTP szerverének központi archívuma.
- acl: a FreeBSD központi ACL listája.

  rsync://ftp13.FreeBSD.org/

  Elérhetõ gyûjtemények:

- FreeBSD: a FreeBSD FTP szerver teljes tükrözése.

Függelék B: Irodalomjegyzék

Míg a man oldalak a FreeBSD operációs rendszer egyes önálló részeit tárgyalják, ismert a tény, hogy arról egyáltalán nem szólnak, miképpen illeszkednek egymáshoz ezek az alkotóelemek, és ezáltal hogyan mûködik maga az operációs rendszer. Erre a célra egyedül csak egy jó UNIX®-os rendszeradminisztrációs szakkönyv és egy jó felhasználói kézikönyv alkalmas.

B.1. A FreeBSD-rõl szóló könyvek és folyóiratok

Idegennyelvû könyvek és folyóiratok:

- Using FreeBSD (kínai). Drmaster, 1997. ISBN 9-578-39435-7.
- FreeBSD Unleashed (kínai fordítás). China Machine Press. ISBN 7-111-10201-0.
- FreeBSD From Scratch (1. kiadás, kínai). China Machine Press. ISBN 7-111-07482-3.
- FreeBSD From Scratch (2. kiadás, kínai). China Machine Press. ISBN 7-111-10286-X.
- FreeBSD Handbook (2. kiadás, kínai). Posts & Telecom Press. ISBN 7-115-10541-3.
- FreeBSD 3.x Internet (kínai). Tsinghua University Press. ISBN 7-900625-66-6.
- FreeBSD & Windows (kínai). China Railway Publishing House. ISBN 7-113-03845-X
- FreeBSD Internet Services HOWTO (kínai). China Railway Publishing House. ISBN 7-113-03423-3
- FreeBSD for PC 98’ers (japán). SHUWA System Co, LTD. ISBN 4-87966-468-5 C3055 P2900E.
- FreeBSD (japán). CUTT. ISBN 4-906391-22-2 C3055 P2400E.
- Complete Introduction to FreeBSD (japán). Shoeisha Co., Ltd. ISBN 4-88135-473-6 P3600E.
- Personal UNIX® Starter Kit FreeBSD (japán). ASCII. ISBN 4-7561-1733-3 P3000E.
- FreeBSD Handbook (japán fordítás). ASCII. ISBN 4-7561-1580-2 P3800E.
- FreeBSD mit Methode (német). Computer und Literatur Verlag/Vertrieb Hanser, 1998. ISBN 3-932311-31-0.
- FreeBSD 4 - Installieren, Konfigurieren, Administrieren (német). Computer und Literatur Verlag, 2001. ISBN 3-932311-88-4.
- FreeBSD 5 - Installieren, Konfigurieren, Administrieren (német). Computer und Literatur Verlag, 2003. ISBN 3-936546-06-1.
- FreeBSD de Luxe (német). Verlag Modere Industrie, 2003. ISBN 3-8266-1343-0.
- FreeBSD Install and Utilization Manual (japán). Mainichi Communications Inc., 1998. ISBN 4-8399-0112-0.
- Onno W Purbo, Dodi Maryanto, Syahrial Hubbany, Widjil Widodo *Building Internet Server with FreeBSD* (indonéz nyelven). Elex Media Komputindo.
- Absolute BSD: The Ultimate Guide to FreeBSD (kínai fordítás). GrandTech Press, 2003. ISBN 986-7944-92-5.
- The FreeBSD 6.0 Book (kínai). Drmaster, 2006. ISBN 9-575-27878-X.

Angol nyelvû könyvek és folyóiratok:

- Absolute BSD, 2nd Edition: The Complete Guide to FreeBSD. No Starch Press, 2007. ISBN: 978-1-59327-151-0
- The Complete FreeBSD. O'Reilly, 2003. ISBN: 0596005164
- The FreeBSD Corporate Networker's Guide. Addison-Wesley, 2000. ISBN: 0201704811
- FreeBSD: An Open-Source Operating System for Your Personal Computer. The Bit Tree Press, 2001. ISBN: 0971204500
- Teach Yourself FreeBSD in 24 Hours. Sams, 2002. ISBN: 0672324245
- FreeBSD 6 Unleashed. Sams, 2006. ISBN: 0672328755
- FreeBSD: The Complete Reference. McGrawHill, 2003. ISBN: 0072224096
- BSD Magazine, megjelenik a Software Press Sp., z o.o. SK gondozásában. ISSN 1898-9144

B.2. Felhasználói kézikönyvek

- Computer Systems Research Group, UC Berkeley. *4.4BSD User's Reference Manual*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-075-9
- Computer Systems Research Group, UC Berkeley. *4.4BSD User's Supplementary Documents*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-076-7
- *UNIX® in a Nutshell*. O'Reilly & Associates, Inc., 1990. ISBN 093717520X
- Mui, Linda. *What You Need To Know When You Can't Find Your UNIX® System Administrator*. O'Reilly & Associates, Inc., 1995. ISBN 1-56592-104-6
- Ohio Állami Egyetemnek van egy Alapozó UNIX® kurzusa, amely az Interneten keresztül is elérhetõ HTML és PostScript formátumokban.

  Ennek a dokumentumnak egy olasz fordítása is elérhetõ az Olasz FreeBSD Dokumentációs Projekt keretében.

- Jpman Project, Japanese FreeBSD User's Group. FreeBSD User's Reference Manual (japán fordítás). Mainichi Communications Inc., 1998. ISBN4-8399-0088-4 P3800E.
- Az Edinburghi Egyetemen készítettek az újoncok számára egy Internetes kézikönyvet a UNIX® környezetekhez.

B.3. Rendszeradminisztrátori kézikönyvek

- Albitz, Paul and Liu, Cricket. *DNS and BIND* (4. kiadás). O'Reilly & Associates, Inc., 2001. ISBN 1-59600-158-4
- Computer Systems Research Group, UC Berkeley. *4.4BSD System Manager's Manual*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-080-5
- Costales, Brian és mások. *Sendmail* (2. kiadás). O'Reilly & Associates, Inc., 1997. ISBN 1-56592-222-0
- Frisch, Æleen. *Essential System Administration* (2. kiadás). O'Reilly & Associates, Inc., 1995. ISBN 1-56592-127-5

- Hunt, Craig. *TCP/IP Network Administration* (2. kiadás). O'Reilly & Associates, Inc., 1997. ISBN 1-56592-322-7
- Nemeth, Evi. *UNIX® System Administration Handbook* (3. kiadás). Prentice Hall, 2000. ISBN 0-13-020601-6
- Stern, Hal. *Managing NFS and NIS*. O'Reilly & Associates, Inc., 1991. ISBN 0-937175-75-7
- Jpman Project, Japan FreeBSD Users Group. FreeBSD System Administrator's Manual (japán fordítás). Mainichi Communications Inc., 1998. ISBN4-8399-0109-0 P3300E.
- Dreyfus, Emmanuel. Cahiers de l'Admin: BSD (2. kiadás, franciául). Eyrolles, 2004. ISBN 2-212-11463-X

B.4. Programozói kézikönyvek

- Asente, Paul, Converse, Diana, and Swick, Ralph. *X Window System Toolkit*. Digital Press, 1998. ISBN 1-55558-178-1
- Computer Systems Research Group, UC Berkeley. *4.4BSD Programmer's Reference Manual*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-078-3
- Computer Systems Research Group, UC Berkeley. *4.4BSD Programmer's Supplementary Documents*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-079-1
- Harbison, Samuel P. and Steele, Guy L. Jr. *C: A Reference Manual* (4. kiadás). Prentice Hall, 1995. ISBN 0-13-326224-3
- Kernighan, Brian and Dennis M. Ritchie. *The C Programming Language* (2. kiadás). PTR Prentice Hall, 1988. ISBN 0-13-110362-8
- Lehey, Greg. *Porting UNIX® Software*. O'Reilly & Associates, Inc., 1995. ISBN 1-56592-126-7
- Plauger, P. J. *The Standard C Library*. Prentice Hall, 1992. ISBN 0-13-131509-9
- Spinellis, Diomidis. Code Reading: The Open Source Perspective. Addison-Wesley, 2003. ISBN 0-201-79940-5
- Spinellis, Diomidis. Code Quality: The Open Source Perspective. Addison-Wesley, 2006. ISBN 0-321-16607-8
- Stevens, W. Richard and Stephen A. Rago. *Advanced Programming in the UNIX® Environment* (2. kiadás). Reading, Mass. : Addison-Wesley, 2005. ISBN 0-201-43307-9
- Stevens, W. Richard. *UNIX® Network Programming* (2. kiadás), PTR Prentice Hall, 1998. ISBN 0-13-490012-X
- Wells, Bill. "Writing Serial Drivers for UNIX®". *Dr. Dobb's Journal*. 19(15), 1994. december, 68-71. és 97-99. oldal.

B.5. Az operációs rendszerek belsõ mûködéséröl

- Andleigh, Prabhat K. *UNIX® System Architecture*. Prentice-Hall, Inc., 1990. ISBN 0-13-949843-5
- Jolitz, William. "Porting UNIX® to the 386". *Dr. Dobb's Journal*. 1991. január - 1992. július.
- Leffler, Samuel J., Marshall Kirk McKusick, Michael J Karels és John Quarterman. *The Design and*

Implementation of the 4.3BSD UNIX® Operating System. Reading, Mass. : Addison-Wesley, 1989. ISBN 0-201-06196-1

- Leffler, Samuel J., Marshall Kirk McKusick. *The Design and Implementation of the 4.3BSD UNIX® Operating System: Answer Book*. Reading, Mass. : Addison-Wesley, 1991. ISBN 0-201-54629-9
- McKusick, Marshall Kirk, Keith Bostic, Michael J Karels és John Quarterman. *The Design and Implementation of the 4.4BSD Operating System*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-54979-4

  (A könyv 2. fejezete elérhetõ online a FreeBSD Dokumentációs Projekt részeként, valamint itt a 9. fejezet.)

- Marshall Kirk McKusick, George V. Neville-Neil. *The Design and Implementation of the FreeBSD Operating System*. Boston, Mass. : Addison-Wesley, 2004. ISBN 0-201-70245-2
- Stevens, W. Richard. *TCP/IP Illustrated, Vol 1: The Protocols*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-63346-9
- Schimmel, Curt. *UNIX® Systems for Modern Architectures*. Reading, Mass. : Addison-Wesley, 1994. ISBN 0-201-63338-8
- Stevens, W. Richard. *TCP/IP Illustrated, Vol 3: TCP for Transactions, HTTP, NNTP and the UNIX® Domain Protocols*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-63495-3
- Vahalia, Uresh. *UNIX® Internals - The New Frontiers*. Prentice Hall, 1996. ISBN 0-13-101908-2
- Wright, Gary R. és W. Richard Stevens. *TCP/IP Illustrated, Vol 2: The Implementation*. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-63354-X

B.6. Biztonságról szóló írások

- Cheswick, William R. és Steven M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-63357-4
- Garfinkel, Simson és Gene Spafford. *Practical UNIX® & Internet Security* (2. kiadás). O'Reilly & Associates, Inc., 1996. ISBN 1-56592-148-8
- Garfinkel, Simson. *PGP Pretty Good Privacy*. O'Reilly & Associates, Inc., 1995. ISBN 1-56592-098-8

B.7. Hardverrel foglalkozó írások

- Anderson, Don és Tom Shanley. *Pentium Processor System Architecture* (2. kiadás). Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40992-5
- Ferraro, Richard F. *Programmer's Guide to the EGA, VGA, and Super VGA Cards* (3. kiadás). Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-62490-7
- Az Intel® által gyártott processzorokról és chipsetekrõl, valamint az általuk kialakított szabványokról a saját fejlesztõi oldalukon, általában PDF állományok formájában kaphatunk információkat.
- Shanley, Tom. *80486 System Architecture* (3. kiadás). Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40994-1

- Shanley, Tom. *ISA System Architecture* (3. kiadás). Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40996-8
- Shanley, Tom. *PCI System Architecture* (4. kiadás). Reading, Mass. : Addison-Wesley, 1999. ISBN 0-201-30974-2
- Van Gilluwe, Frank. *The Undocumented PC* (2. kiadás). Reading, Mass: Addison-Wesley Pub. Co., 1996. ISBN 0-201-47950-8
- Messmer, Hans-Peter. *The Indispensable PC Hardware Book* (4. kiadás). Reading, Mass: Addison-Wesley Pub. Co., 2002. ISBN 0-201-59616-4

B.8. UNIX® történelem

- Lion, John. *Lion's Commentary on UNIX*® (6. kiadás, forráskóddal). ITP Media Group, 1996. ISBN 1573980137
- Raymond, Eric S. *The New Hacker's Dictionary* (3. kiadás). MIT Press, 1996. ISBN 0-262-68092-0. Vagy Zsargon fájlként is ismert.
- Salus, Peter H. *A quarter century of UNIX*®. Addison-Wesley Publishing Company, Inc., 1994. ISBN 0-201-54777-5
- Simon Garfinkel, Daniel Weise, Steven Strassmann. *The UNIX®-HATERS Handbook*. IDG Books Worldwide, Inc., 1994. ISBN 1-56884-203-1. Elfogyott, de még elérhetõ ezen a linken.
- Don Libes, Sandy Ressler. *Life with UNIX*® - különkiadás. Prentice-Hall, Inc., 1989. ISBN 0-13-536657-7
- *The BSD family tree*. http://www.FreeBSD.org/cgi/cvsweb.cgi/src/shared/misc/bsd-family-tree vagy egy telepített FreeBSD rendszeren a /usr/shared/misc/bsd-family-tree állomány.
- *Networked Computer Science Technical Reports Library*. http://www.ncstrl.org/
- *Old BSD releases from the Computer Systems Research group (CSRG)*. http://www.mckusick.com/csrg/Ez a 4 CD-s készlet tartalmazza az összes BSD verziót a 1BSD-tõl kezdve a 4.4BSD és 4.4BSD-Lite2-ig (de nem a 2.11BSD-t sajnos nem). Az utolsó lemezen megtalálhatóak a végleges források, illetve az SCCS állományok.

B.9. Magazinok és folyóiratok

- *The C/C++ Users Journal*. R&D Publications Inc. ISSN 1075-2838
- *Sys Admin - The Journal for UNIX® System Administrators*. Miller Freeman, Inc. ISSN 1061-2688
- *freeX - Das Magazin für Linux® - BSD - UNIX*® (német). Computer- und Literaturverlag GmbH. ISSN 1436-7033

Függelék C: Források az interneten

A FreeBSD gyors ütemû fejlõdése a nyomtatott médiát alkalmatlanná teszi a legfrissebb fejlesztések nyomonkövetésére. Ezzel szemben az elektronikus erõforrások a biztos, ha gyakran nem is csak az egyetlen, módjai a legújabb elõrelépések figyelemmel követésének. Mivel a FreeBSD-t többségében önkéntesek fejlesztik, az õt körülvevõ felhasználói közösség önmaga is egyfajta "szakmai segélynyújtó egyletként" funkcionál, amelyet leghatékonyabban elektronikus levélben, webes fórumokon vagy USENET hírcsoportokon keresztül érhetünk el.

A továbbiakban a FreeBSD felhasználók közösségének különbözõ fajtájú elérhetõségeit vázoljuk fel nagyvonalakban. Ha úgy érezzük, hogy ebbõl a felsorolásban kimaradt volna valami, akkor ne habozzunk róla értesítést küldeni a FreeBSD Dokumentációs Projekt levelezési lista címére (angolul), hogy felvehessük a többi közé.

C.1. Levelezési listák

A FreeBSD köré csoportosulókat levelezési listákon keresztül tudjuk közvetlenül elérni, ezen a módon tehetünk fel kérdéseket, vethetünk fel témákat. Ezek között több különbözõ területtel foglalkozó listát találhatunk. Ezért célszerû mindig a hozzászólásainkat a témánkhoz legközelebb álló listára küldeni, mert enélkül szinte biztos, hogy nem kapunk pontos vagy gyors választ.

A különbözõ listák témájának rövid leírása a dokumentum alján olvasható. *Szeretnénk mindenkit megkérni, hogy mielõtt feliratkozik vagy levelet küld valamelyik listára, figyelmesen olvassa el ezeket.* Az egyes listák tagjai már így is naponta többszáz FreeBSD-vel kapcsolatos üzenetet kapnak, miközben a listák tematikájának és szabályainak lefektetésével igyekszünk a jel-zaj arányt minél kedvezõbb szinten tartani. Ezek nélkül a levelezési listák a Projekt számára haszontalan kommunikációs eszközökké válnának.

A FreeBSD test levelezési lista címet használjuk, ha ki akarjuk próbálni, hogy tudunk-e levelet küldeni a FreeBSD listáira. A többi listára viszont lehetõleg ne küldjünk teszt jellegû üzeneteket.

Ha nem tudjuk eldönteni, hogy pontosan melyik listát is kellene megcímeznünk kérdésünkkel, olvassuk el a Hogyan kapjunk értékelhetõ választ a FreeBSD-questions levelezési listáról címû leírást (angolul).

Mielõtt akármelyik listára is levelet küldenénk, olvassuk el a Levelezési listák Gyakran Ismételt Kérdéseit (angolul), amivel elkerülhetjük a gyakran feltett kérdések és témák ismételt felhozását.

A levelezési listák tartalma folyamatosan archiválódik, és ezekben az archívumokban a FreeBSD honlapján tudunk keresni. Az itt elérhetõ, kulcsszavak alapján történõ keresés remek módját nyújtja a gyakran felmerülõ kérdések egyszerû és gyors megválaszolásának, ezért ilyen esetekben elõször mindig ezt javasolt használni. Ez egyben mellesleg azt is jelenti, hogy a FreeBSD levelezési listáira küldött üzenetek fennmaradnak az örökkévalóságig. Ha a beküldendõ üzenet bizalmas információkat tartalmaz, érdemes megfontolni egy eldobható anonim e-mail cím használatát és kizárólag csak a publikus részet beküldeni.

C.1.1. A listák összefoglalása

Általános listák: A következõ általános célú listákhoz szabadon (és nyugodtan) csatlakozhatunk:

Lista	Tartalom
freebsd-advocacy	A FreeBSD igéjének terjesztése
FreeBSD announcements levelezési lista	Fontosabb események és elõrelépések a projektek életében
freebsd-arch	Architekturális és tervezési kérdések tárgyalása
freebsd-bugbusters	A FreeBSD hibabejelentéseit tároló adatbázis és a kapcsolódó eszközök karbantartására vonatkozó megbeszélések
freebsd-bugs	Hibajelentések
freebsd-chat	A FreeBSD közösség nem szakmai jellegû dolgai
FreeBSD-CURRENT levelezési lista	A FreeBSD-CURRENT használatának tárgyalása
freebsd-isp	A FreeBSD-t alkalmazó internet-szolgáltatók fóruma
freebsd-jobs	FreeBSD-s munkalehetõségek
freebsd-policy	A FreeBSD fejlõdését irányító csoport (Core Team) döntéseirõl tájékoztató lista. A forgalma kicsi, csak olvasható.
freebsd-questions	A felhasználók kérdései és szakmai segítségnyújtás
FreeBSD security notifications levelezési lista	Biztonsági figyelmeztetések
FreeBSD-STABLE; levelezési lista	A FreeBSD-STABLE használatát illetõ kérdések
FreeBSD test levelezési lista	Ide lehet küldeni a próbaüzeneteket

Szakmai listák: A következõ listák szakmai jellegû témákat képviselnek. Mielõtt bármelyikükre levelet küldenénk vagy feliratkoznánk, figyelmesen olvassuk el a tartalmukat és céljaikat bemutató rövid leírásukat.

Lista	Tartalom
FreeBSD ACPI levelezési lista	Az ACPI és energiagazdálkodás támogatás fejlesztése
freebsd-afs	Az AFS portolása FreeBSD-re
freebsd-aic7xxx	Az Adaptec® AIC 7xxx sorozat meghajtóinak fejlesztése
alpha	A FreeBSD Alpha portja
freebsd-amd64	A FreeBSD AMD64 portja
freebsd-apache	Az Apache és hozzá tartozó portok tárgyalása

Lista	Tartalom
freebsd-arm	A FreeBSD ARM® portja
freebsd-atm	FreeBSD használata ATM hálózatokkal
freebsd-audit	A forráskód ellenõrzésérõl szóló projekt
freebsd-binup	A bináris frissítésekkel foglalkozó rendszer tervezése és fejlesztése
freebsd-bluetooth	A Bluetooth® technológia használata a FreeBSD-ben
freebsd-cluster	A FreeBSD klaszteres környezetben
freebsd-cvsweb	A CVSweb karbantartása
freebsd-database	Adatbázisok használata és fejlesztése FreeBSD alatt
freebsd-doc	FreeBSD-rõl szóló leírások készítése
freebsd-drivers	Eszközmeghajtók írása FreeBSD-re
freebsd-eclipse	Az Eclipse integrált fejlesztõi környezet, eszközeinek, gazdag kliens alkalmazásinak és portjainak FreeBSD alatti használata
freebsd-embedded	A FreeBSD használata beágyazott alkalmazásokban
freebsd-eol	Olyan FreeBSD-s szoftverek független továbbfejlesztése, amelyeket hivatalosan már nem támogatnak
freebsd-emulation	Linux/MS-DOS®/Windows® és hasonló rendszerek emulációja
freebsd-firewire	A FreeBSD és a FireWire® (iLink, IEEE 1394) kapcsolatának technikai kérdései
freebsd-fs	Állományrendszerek
freebsd-gecko	A Gecko Rendering Engine alkalmazásával kapcsolatos problémák
freebsd-geom	A GEOM-hoz tartozó témák és implementációk
freebsd-gnome	A GNOME és GNOME-alkalmazások portolása
freebsd-hackers	Általános szakmai témák
freebsd-hardware	A FreeBSD futtatására szolgáló hardverekkel foglalkozó témák
freebsd-i18n	A FreeBSD honosítása
freebsd-ia32	A FreeBSD használata az IA-32 (Intel® x86) platformon

Lista	Tartalom
freebsd-ia64	A FreeBSD portolása az Intel® következõ IA64 rendszereire
freebsd-ipfw	Az IP tûzfal kódjának újratervezését érintõ szakmai megbeszélések
freebsd-isdn	ISDN fejlesztõk levelei
freebsd-jail	A jail(8) segédprogram
freebsd-java	Java™ fejlesztõk kérdései és a JDK™-k átültetése FreeBSD-re
freebsd-kde	A KDE és KDE-alkalmazások portolása
freebsd-lfs	Az LFS portolása FreeBSD-re
freebsd-libh	A második generációs telepítõ- és csomagrendszer
freebsd-mips	A FreeBSD portolása MIPS®-re
freebsd-mobile	A mobil számítógépekkel kapcsolatos megbeszélések
freebsd-mono	Mono és C# alkalmazások FreeBSD alatt
freebsd-mozilla	A Mozilla átültetése FreeBSD-re
FreeBSD multimedia levelezési lista	Multimédia alkalmazások
freebsd-new-bus	A buszarchitektúrával kapcsolatos szakmai megbeszélések
freebsd-net	A TCP/IP forráskódjával és hálózatkezeléssel kapcsolatos kérdések
freebsd-openoffice	A OpenOffice.org és StarOffice™ alkalmazások portolása FreeBSD-re
freebsd-performance	Nagy terhelésû és teljesítményû rendszerek teljesítményhangolási kérdései
freebsd-perl	A rengeteg Perl alapú port karbantársa
freebsd-pf	A csomagszûrõ mûködésével kapcsolatos kérdések és megbeszélések
freebsd-platforms	Portolás nem Intel® architektúrájú platformokra
freebsd-ports	A Portgyûjtemény mûködése
freebsd-ports-bugs	A portokhoz tartozó hibák és hibajelentések megbeszélése
freebsd-ppc	A FreeBSD portolása PowerPC®-re
freebsd-proliant	HP ProLiant szerverek és a FreeBSD kapcsolata

Lista	Tartalom
freebsd-python	A Python FreeBSD-n futó változatának problémái
qa	A minõségbiztosítás megbeszélése, különösen a kiadások közeledtével
freebsd-rc	Az rc.d rendszer és annak fejlõdése
freebsd-realtime	A FreeBSD valósidejû kiterjesztéseinek fejlesztése
freebsd-ruby	A Ruby használata FreeBSD rendszereken
freebsd-scsi	A SCSI alrendszer
FreeBSD security levelezési lista	A FreeBSD mûködését fenyegetõ biztonsági problémák
freebsd-small	A FreeBSD használata beágyazott alkalmazásokban (elavult; helyette a freebsd-embedded címét használjuk)
smp	Az [A]Szimmetrikus többszálú feldolgozáshoz ([A]Symmetric MultiProcessing) tartozó tervezési megbeszélések
freebsd-sparc64	A FreeBSD portolása SPARC® alapú rendszerekre
freebsd-standards	A FreeBSD megfelelése a C99 és POSIX® szabványoknak
freebsd-sun4v	A FreeBSD portolása UltraSPARC® T1 alapú rendszerekre
freebsd-sysinstall	A sysinstall(8) fejlesztése
freebsd-threads	A FreeBSD szálkezelése
freebsd-testing	A FreeBSD teljesítmény- és megbízhatósági tesztjei
freebsd-tilera	A FreeBSD portolása a Tilera processzorcsalád tagjaira
freebsd-tokenring	A Token Ring támogatása a FreeBSD-ben
freebsd-toolchain	A FreeBSD alapvetõ segédprogramjainak karbantartása
freebsd-usb	USB támogatás a FreeBSD-ben
freebsd-virtualization	A FreeBSD részérõl támogatott különbözõ virtualizációs technológiák tárgyalása
freebsd-vuxml	A VuXML infrastruktúra tárgyalása
freebsd-x11	Az X11 karbantartása és támogata FreeBSD alatt

Lista	Tartalom
freebsd-xen	A Xen™ FreeBSD portjának (implementációk, használat) tárgyalása

Korlátozott listák: (Limited lists) A következõ listák sokkal jobban specializálódótt (és igényesebb) közösségnek szólnak, nem a nagyközönségnek. Ezért mielõtt egy ilyen listára feliratkoznánk, érdemes némi tapasztalatot gyûjtenünk a szakmai témájú listákon, így megismerjük az itt alkalmazott kommunikációs szabályokat.

Lista	Tartalom
freebsd-hubs	A tükrözések üzemeltetõi számára (infrastrukturális támogatás)
freebsd-user-groups	A felhasználói csoportok összefogása
freebsd-vendors	A forgalmazók koordinálása a kiadások elõtt
freebsd-wip-status	A FreeBSD-vel kapcsolatos folyamatban levõ fejlesztések helyzetjelentései
freebsd-www	A www.FreeBSD.org karbantartói számára

Kivonatolt listák: (Digest lists) Az eddig említett listák elérhetõek kivonatolt formában is. Miután feliratkoztunk egy listára, a hozzáférésünk beállításainál kiválaszthatjuk, hogy kivonatolt formátumban kívánjuk-e kapni a leveleket.

CVS és SVN listák: (CVS SVN lists) A következõ listák a forrásfa különbözõ részeinek változtatásáról és a hozzájuk tartozó üzenetekrõl adnak értesítést. Ezek a listák *csak olvasásra* vannak, nem szabad rájuk levelet küldeni.

Lista	Forráskód területe	A terület leírása (minek a forrása)
cvs-all	/usr/(CVSROOT|doc|ports)	A fában végzett akármelyik módosítás (az összes CVS lista együtt)
cvs-doc	/usr/(doc|www)	A doc és www ágak változásai
cvs-ports	/usr/ports	A portfa változásai
cvs-projects	/usr/projects	A projektek változásai
cvs-src	/usr/src	A rendszer forrásának változásai (az svn és cvs közti importer mûködése alapján generálódik)
A teljes src fa SVN commit üzenetei (kivéve "user" és "projects")	/usr/src	A Subversion repositoryk változásai (kivéve a user és a projects)

Lista	Forráskód területe	A terület leírása (minek a forrása)
Az src fa head/-current ágának SVN commit üzenetei	/usr/src	A Subversion repository "fõágának" (a FreeBSD-CURRENT forrásainak) változásai
svn-src-projects	/usr/projects	A projects változásai a forrásokat tároló Subversion repositoryn belül
svn-src-release	/usr/src	A releases változásai a forrásokat tároló Subversion repositoryn belül
svn-src-releng	/usr/src	A releng ágak (biztonsági frissítések és kiadások) változásai a forrásokat tároló Subversion repositoryn belül
svn-src-stable	/usr/src	A stabil verziókhoz tartozó ágak változásai a forrásokat tároló Subversion repositoryn belüle
svn-src-stable-6	/usr/src	A stable/6 ág változásai a forrásokat tároló Subversion repositoryn belül
svn-src-stable-7	/usr/src	A stable/7 ág változásai a forrásokat tároló Subversion repositoryn belül
svn-src-stable-8	/usr/src	A stable/8 ág változásai a forrásokat tároló Subversion repositoryn belül
svn-src-stable-other	/usr/src	A Subversion repositoryban található korábbi stable ágak változásai
svn-src-svnadmin	/usr/src	A forrásokat tároló Subversion repositoryhoz tartozó szkriptek és egy konfigurációs állományok változásai
svn-src-user	/usr/src	A user változásai a forrásokat tároló Subversion repositoryn belül
svn-src-vendor	/usr/src	A vendor változásai a forrásokat tároló Subversion repositoryn belül

C.1.2. Hogyan iratkozzunk fel

Ha fel akarunk iratkozni valamelyik listára, kattintsunk a nevére, vagy menjünk a https://lists.freebsd.org címre és a válasszuk ki onnan a keresett listát. A lista oldalán megtalálunk minden feliratkozással kapcsolatos utasítást.

Ténylegesen úgy tudunk üzenni egy listára, ha levelet küldünk az listanév@FreeBSD.org címre, amely ezután a lista tagjai között kézbesítésre kerül a világban.

A listáról úgy tudunk leiratkozni, ha a róla kapott valamelyik levél alján található URL-re kattintunk. Másik megoldás, ha magunk küldünk egy levelet a listanév-unsubscribe@FreeBSD.org címre.

Még egyszer szeretnénk kérni, hogy a szakmai témájú levelezési listákon folyó társalgásokat igyekezzünk az adott témán belül tartani. Ha csupán a fontosabb bejelentésekre vagyunk kíváncsiak, akkor a kisforgalmú FreeBSD announcements levelezési lista használatát válasszuk.

C.1.3. A listák tematikája

Minden FreeBSD-s levelezési lista rendelkezik bizonyos alapszabályokkal, amelyek minden tagnak el kell fogadnia. Az ismeretett irányelvek elleni vétkezés a FreeBSD postamesterének postmaster@FreeBSD.org két (2, azaz kettõ) írásos figyelmeztetését vonja maga után, amelyek figyelmen kívül hagyásával, tehát a harmadik szabálysértés alkalmával, a küldõ eltávolításra kerül a FreeBSD összes levelezési listájáról és a továbbiakban szûrni fogják a leveleit. Sajnáljuk, hogy ilyen szabályokat és szankciókat kellett bevezetnünk, de napjaink internetes technológiái igen elvadultak és ahogy az látható is, sokan egyszerûen nem fogják fel, mennyire sérülékenyek egyes részei.

Közlekedési szabályok:

- Minden beküldött levél témájának meg kell felelnie az adott lista tartalmának, tehát például a szakmai kérdésekkel foglalkozó listákon csak szakmai témájú leveleknek szabad megjelenniük. Az oda nem illõ cseverészés és értelmetlen vitázás csak a lista értékét csökkenti, ezért ezt senkitõl sem tûrjük. A kötetlenebb, konkrét téma nélküli megbeszéléseket inkább a FreeBSD chat levelezési lista címén folytassuk.
- 2 listánál többre ne küldjük be ugyanazt a levelet, és 2 listára is csak akkor küldjük, ha az egyértelmûen és nyilvánvalóan indokolt. A legtöbb listánál így is rengeteg az átfedés, kivéve a legtitkosabb kombinációkat (például "-stable és -scsi"), ezért nem túl sok értelme van egyszerre egynél több listát is értesíteni. Ha olyan üzenetet kapunk, amelynek a `Cc` (másolat) mezõjében több lista címe is szerepel, akkor továbbküldés vagy válaszadás során töröljük ezeket. *Az általunk küldött levelekért továbbra is mi magunk vagyunk a felelõsek, függetlenül attól, hogy ki volt a levél eredeti feladója.*
- Tilos (vita közben) személyeskedni vagy káromkodni, beleértve a felhasználókat és a fejlesztõket is. A netikett megszegését, például a privát levelezés elõzetes engedély nélküli továbbküldését vagy egyes részleteinek közlését, elítéljük, de nyíltan nem tiltjuk. Nagyon ritka esetekben *azonban* elõfordulhat, hogy a sértõ tartalom önmagában ellenkezik a lista elveivel és figyelmeztetést (esetleg kitiltást) von maga után.
- A FreeBSD-hez nem kötõdõ termékek vagy szolgáltatások reklámozása szigorúan tilos, és ha

bebizonyosodik, hogy a küldõ szándékosan küldte szét, akkor azonnali kitiltásban részesül.

Az egyes listák tematikája:

FreeBSD ACPI levelezési lista

Az ACPI és energiagazdálkodás támogatásának fejlesztése

freebsd-afs

Andrew File System

Ez a lista a CMU/Transarc AFS portolásáról szól

FreeBSD announcements levelezési lista

Fontosabb események / nagyobb lépések

Olyan emberek számára ajánlott ez a levelezési lista, akik csak a FreeBSD jelentõsebb eseményei bejelentései iránt érdeklõdnek. Ide értendõk a különbözõ idõközi és egyéb kiadások, a FreeBSD újításainak bejelentései. Idõnként önkéntesek toborzására stb. is használják. A forgalma nagyon kicsi, tartalma szigorúan ellenõrzött.

freebsd-arch

Architekturális és tervezési kérdések

Ez a lista a FreeBSD architektúráját érintõ megbeszélések színtere. Az itt megjelenõ üzenetek szigorúan szakmai jellegûek. Néhány idevágó téma:

- Hogyan alakítsuk úgy át a fordítási rendszert, hogy egyszerre több különbözõ paraméterû fordítás is képes legyen futni.
- Mit kellene javítani a VFS-en a Heidemann-rétegek mûködéséhez.
- Hogyan tudnánk úgy átalakítani az eszközmeghajtók felületét, hogy ugyanazok a meghajtók minden gond nélkül képesek legyenek több buszon és architektúrán is mûködni.
- Hogyan írjunk meghajtót hálózati eszközökhöz.

freebsd-audit

A forráskód vizsgálatát végzõ projekt

Ez a levelezési lista a FreeBSD forráskódjának vizsgálatával foglalkozik. Habár eredetileg csak a biztonságot érintõ változtatások ellenõrzésére jött létre, napjainkra már a forráskód mindenféle változását felülvizsgálja.

Erre a listára rengeteg javítás érkezik, amelyek valószínûleg egy átlag FreeBSD felhasználó számára nem túlzottan érdekesek. A kód változásától független biztonsági kérdések megvitatása a freebsd-security listán történik. Viszont az összes fejlesztõnek javasoljuk, hogy küldjék be felülvizsgálatra a javításaikat, különösen abban az esetben, amikor a forráskód olyan részéhez nyúlnak, ahol az adott hiba javítása a rendszer egészének mûködésére kihatással lehet.

freebsd-binup

A FreeBSD bináris frissítésével foglalkozó projekt

Ez a lista ad otthont a binup vagy más néven a bináris frissítési rendszer (binary update system) körül felmerülõ problémák tárgyalásának. Tervezési kérdések, implementációs részletek, javítások, hiba- és állapotjelentések, funkciók igénylése, a kód változásainak naplózása és minden, ami a binuppal kapcsolatos.

freebsd-bluetooth

Bluetooth® a FreeBSD-ben

Ez a Bluetooth®-os FreeBSD felhasználók gyülekezõhelye. Tervezési és implementációs kérdések, javítások, hiba- és állapotjelentések, funkciók igénylése, minden, ami Bluetooth®.

freebsd-bugbusters

A hibajelentések kezelésének összefogása

A lista célja a Bugmeister és az õ Bugbustereinek, valamint a hibajelentések adatbázisai iránti kifejezetten érdeklõdõ személyek együttmûködésének és kapcsolattartásának elõsegítése. Ez a lista nem az egyes hibákról, javításokról vagy azok jelentésérõl szól.

freebsd-bugs

Hibajelentések

Ezen a levelezési listán lehet a FreeBSD hibáit bejelenteni. Ha lehet, akkor a hibákat a send-pr(1) paranccsal vagy a webes felületen keresztül küldjük be.

freebsd-chat

A FreeBSD közösség nem szakmai jellegû dolgai

Erre a listára kerül minden olyan nem szakmai jellegû, társadalmi érintkezéssel kapcsolatos információ, ami a többi listáról kimaradt: Jordan mennyire hasonlít a rajzfilmeken látható vadászgörényre, kis- vagy nagybetûvel írjuk-e, ki iszik sok kávét, hol fõzik a legjobb söröket, ki fõz sört az alagsorában és így tovább. Elvétve felbukkannak olyan fontosabb események is (bulik, lakodalmak, gyermekáldás, új munkahely stb), amelyek ugyan szakmai témájúak, de a folyományaik már inkább a -chat listára tartoznak.

Core Team

A FreeBSD irányítását végzõ csapat

Ezt a belsõ levelezési listát a Core Team tagjai használják. Akkor érdemes ide levelet küldeni, ha FreeBSD-vel kapcsolatos fontos ügyekben lenne szükségünk döntésre vagy véleményre.

FreeBSD-CURRENT levelezési lista

A FreeBSD-CURRENT használatával kapcsolatos megbeszélések

A FreeBSD-CURRENT felhasználóinak levelezési listája. Itt értesülhetünk a -CURRENT felhasználókat érintõ friss újdonságairól, és azokról az utasításokról, amelyek követésével mûködéképesen tarthatjuk a -CURRENT rendszerünket. Aki a "-CURRENT" verziót használja, mindenképpen iratkozzon fel erre a listára. Ez is egy szakmai jellegû lista, ahová csak szigorúan ilyen témákat várnak.

freebsd-cvsweb

A FreeBSD CVSweb projekt

A FreeBSD CVSweb szolgáltatásának használatáról, fejlesztéséről és karbantartásáról szóló megbeszélések.

freebsd-doc

A dokumentációs projekt

Ez a levelezési lista a FreeBSD-rõl szóló különbözõ dokumentumok készítésével kapcsolatos problémák és projektek tárgyalásait öleli fel. A levelezési lista tagjait együttesen a "FreeBSD Dokumentációs Projekt"-nek nevezik. Ez egy nyílt lista, csatlakozzunk hozzá bátran!

freebsd-drivers

Eszközmeghajtók írása FreeBSD-re

A FreeBSD-hez készülõ eszközmeghajtókról szóló szakmai fórum. Elsõsorban itt tehetik fel a meghajtók készítõi a FreeBSD rendszermagjában megtalalálható API-kra vonatkozó kérdéseiket.

freebsd-eclipse

Az Eclipse integrált fejlesztõi környezetének, segéprogramjainak, kliensalkalmazásainak és portjainak FreeBSD felhasználók számára meghirdetett fóruma.

A lista azzal a szándékkal jött létre, hogy kölcsönös támogatást nyújtson az Eclipse fejlesztõi környezet, a hozzá tartozó segédeszközök, kliensalkalmazások FreeBSD változatának megválasztásában, telepítésében és használatában. Emellett az Eclipse környezet és pluginjainak FreeBSD-re történõ portolásáról is szó esik.

Valamint igyekszik minél többet profitálni az Eclipse és a FreeBSD köré csoportosuló közösségek kölcsönös információcseréjébõl.

Habár a lista elsõdlegesen az Eclipse felhasználóinek igényeire koncentrál, azok számára is táptalajt ad, akik az Eclipse keretrendszer segítségével FreeBSD specifikus alkalmazásokat szeretnének kifejleszteni.

freebsd-embedded

A FreeBSD használata beágyazott alkalmazásokban

Ez a lista a FreeBSD beágyazott rendszerekben történõ használatát igyekszik megvitatni. Ez egy szakmai jellegû lista, ezért ide szigorúan csak ilyen témájú leveleket várunk. A listán tárgyalt beágyazott rendszereknek tekintünk minden olyan számítási eszközt, amely az általános számítási környezetekkel szemben egyetlen feladatot lát el. Nem feltétlenül csak ilyenek, de például a különféle telefonok, illetve hálózati eszközök, mint például útválasztók, switchek, PBX-ek, távoli mérõeszközök, PDA-k, eladási rendszerek és így tovább.

freebsd-emulation

A Linux/MS-DOS®/Windows® rendszerek emulációja

Ezen a listán arról értekezhetünk és olvashatunk, hogy FreeBSD alatt miként futtassunk más operációs rendszerekre írt programokat.

freebsd-eol

Összefogás a FreeBSD Projekt által tovább már támogatott, FreeBSD-hez tartozó szoftverekért

Ezen a listán kap vagy kaphat helyet a FreeBSD Projekt által hivatalosan tovább már nem fejlesztett szoftverek felhasználói összefogáson alapuló támogatása (például biztonsági figyelmeztetések vagy javítások formájában).

freebsd-firewire

FireWire® (iLink, IEEE 1394)

Ez a levelezési lista foglalkozik a FreeBSD FireWire® (azaz IEEE 1394, avagy iLink) alrendszerének implementációjával. Az itt felmerülõ témák többek közt a szabványok, buszos eszközök és a hozzájuk tartozó protokollok, vezérlõkártyák és chipkészletek, valamint a mûködtetésükre szánt programok felépítése és megvalósítása.

freebsd-fs

Állományrendszerek

A FreeBSD-ben megjelenõ állományrendszerek kivesézése. Mivel ez egy szakmai jellegû lista, ide határozottan csak ilyen jellegû leveleket várunk.

freebsd-gecko

Gecko Rendering Engine

Ezen a levelezési listán a Gecko FreeBSD rendszerekre portolt változatával kapcsolatos fórumot találjuk.

Az itt felmerülõ témák többségükben a Gecko alapú alkalmazásokról, telepítésükrõl, és a FreeBSD alatti fejlesztésükrõl, támogatásukról szólnak.

freebsd-geom

GEOM

A GEOM és a vele kapcsolatos implementáció megbeszélései. Szakmai jellegû lista, ezért erre tekintettel csak ilyen témájú leveleket postázzunk ide.

freebsd-gnome

GNOME

A GNOME asztalkörnyezet FreeBSD rendszereket érintõ használatáról szóló lista. Mûszaki jellegû, ezért szigorúan csak ilyen témákban társgalodjunk itt.

freebsd-ipfw

IP tûzfalak

A FreeBSD-ben levõ IP tûzfal újratervezésével foglalkozó elgondolások és szakmai témájú megbeszélések otthona. Ide szigorúan csak ilyen témájú leveleket küldjünk!

freebsd-ia64

A FreeBSD portolása I64-re

Ez a levelezési lista a FreeBSD az Intel® IA-64 platformjára készített portjával foglalkozó egyének kommunikációs eszköze, ahol az ezzel kapcsolatos problémák és azok különbözõ megoldásai kerülnek terítékre. A téma iránt érdeklõdõket is szívesen látjuk.

freebsd-isdn

ISDN kommunikáció

Ez a levelezési lista a FreeBSD ISDN támogatásáról szól.

freebsd-java

Java™ alapú fejlesztések

A levelezési listán a nagyobb Java™ alkalmazások FreeBSD alapú fejlesztését, valamint a JDK™-k portolásáról és karbantartását beszélik meg.

freebsd-jobs

Munkát keres/kínál

Erre a fórumra tudjuk beküldeni a kifejezetten FreeBSD-hez kapcsolódó munkaajánlatokat és önéletrajzokat, tehát ez a megfelelõ hely, ha FreeBSD-s munkát keresünk, vagy éppen FreeBSD szakértõket. Ez azonban *nem* egy általános célú állásbörze, mert arra megvannak a megfelelõ helyek.

Szeretnénk hozzátenni, hogy ez a lista, a többi `FreeBSD.org` levelezési listához hasonlóan, világméretekben mûködik. Ezért ne felejtsük sosem pontosan megjelölni a munkavégzés helyét, illetve hogy milyen kommunikációs és esetlegesen költözési lehetõségeket javaslunk.

A leveleket csak nyílt formátumban küldjük - elsõsorban szöveges formátumban, de az egyszerûbb PDF, HTML vagy még néhány más hozzájuk hasonló formátumot is alkalmazhatunk. Az olyan zárt formátumok, mint például a Microsoft® Word (.doc) azonban nem fognak továbbítódni.

freebsd-kde

KDE

A KDE és FreeBSD kapcsolatáról szóló lista. Szigorúan szakmai jellegû, ezért csak ilyen témájú levelek küldése elfogadott.

freebsd-hackers

Szakmai kérdések

Ez a FreeBSD szakmai jellegû kérdéseivel foglalkozó fórum. Ez az elsõ számú szakmai levelezési lista. A FreeBSD fejlesztésével aktívan foglalkozó egyének számára ajánljuk, hiszen itt vethetik fel problémáikat, itt kereshetnek rájuk megoldásokat. Az ilyen típusú megbeszéléseket figyelemmel követõ egyéneket is szívesen fogadjuk. Mivel ez egy erõsen szakmai jellegû lista, ezért csak ilyen témájú leveleket várunk ide.

freebsd-hardware

A FreeBSD és a hardverek kapcsolatáról általában

Ezen a listán kerül megvitatásra minden olyan hardver, amelyen a FreeBSD mûködik: milyen gondok adódhatnak, milyen hardvereket érdemes beszereznünk vagy elkerülnünk.

freebsd-hubs

Tükrözések

A FreeBSD tükrözéseit karbantartó egyének számára fontos bejelentések és megbeszélések.

freebsd-isp

Az internet-szolgáltatók fóruma

Ezen a levelezési listán a FreeBSD-t használó internet-szolgáltatók tehetik fel kérdéseiket. Szigorúan csak szakmai jellegû kérdések engedélyezettek.

freebsd-mono

Mono és C# alkalmazások FreeBSD alatt

Ezen a levelezési listán a Mono fejlesztõi keretrendszer FreeBSD alatt futó változatával kapcsolatos megbeszélések folynak. Ez egy szakmai jellegû lista. Itt a Mono vagy más C# alkalmazások FreeBSD változatának elkészítésén dolgozó egyének tudnak problémákat felvetni vagy megvitatni a különbözõ megoldásokat. Rajtuk kívül viszont szeretettel várunk minden érdeklõdõt a téma iránt.

freebsd-openoffice

OpenOffice.org

Az OpenOffice.org és StarOffice™ portolásával és karbantartásával kapcsolatos megbeszélések.

freebsd-performance

A FreeBSD hangolásának és gyorsításának tárgyalása

Ezen a levelezési listán van lehetõségük a hackereknek, rendszergazdáknak és/vagy az érintett feleknek a FreeBSD teljesítményével kapcsolatos témákban kifejteni a véleményüket. Leginkább nagy terhelés alatt levõ, vagy teljesítménybeli problémákkal küszködõ, esetleg még többet tudó FreeBSD rendszerek tárgyalása a cél. Lehetõleg az érintett gyártókkal és szállítókkal együttesen próbáljuk kidolgozni a FreeBSD teljesítményének növelésére tett kísérleteinket, ezért õket is szívesen látjuk ezen a listán. Ez a kifejezetten szakmai jellegû lista többségében a tapasztalt FreeBSD felhasználók, hackerek vagy rendszergazdák számára tárja fel a gyors, megbízható és skálázható FreeBSD rendszerek lehetõségeit. Ez alapvetõen nem egy kérdezgetõs lista, ahol a dokumentációk elolvasását tudjuk megspórolni, hanem egy olyan hely, ahol a teljesítményt érintõ megválaszolatlan kérdések és elõremutató fejlesztések nyernek teret.

freebsd-pf

A csomagszûrõ tûzfalrendszerrel kapcsolatos kérdések

A FreeBSD csomagszûrõjéhez (packet filter, pf) tartozó tûzfalrendszer megbeszéléseit összefoglaló lista. Szakmai jellegû fejtegetések és felhasználói kérdések egyaránt jöhetnek. Továbbá ezen a listán foglalkozunk az ALTQ rendszer mûködésével is.

freebsd-platforms

Portolás nem Intel® plaformokra

A FreeBSD különbözõ, nem az Intel® architektúrára építkezõ portjainak indítványozása és általános jellegû megvitatása. Ez egy kiemelten szakmai jellegû lista, ezért ide csak ilyen témájú leveleket várunk.

freebsd-policy

Az Core Team szabályozásai

Alacsony forgalmú, csak olvasható lista, ahol a FreeBSD fejlesztését irányító csoport különbözõ döntéseirõl olvashatunk.

freebsd-ports

A "portok" megbeszélése

A FreeBSD "portgyûjteményével" (/usr/ports), a portok infrastruktúrájával és a portok fejlesztésének irányításával kapcsolatos megbeszélések. Erõsen szakmai jellegû lista, ezért ide csak ilyen témában írjunk.

freebsd-ports-bugs

A "portok" hibáinak tárgyalása

A FreeBSD "portgyûjteményének" (/usr/ports), a bejelentett portok és azok módosításához kötõdõ hibajelentésekkel foglalkozó lista. Ez egy szakmai jellegû lista, ahol csak ilyen jellegû témákra számítunk.

freebsd-proliant

A FreeBSD és a HP ProLiant szerverek kapcsolatát érintõ szakmai megbeszélések

Ezen a levelezési listán a FreeBSD HP ProLiant szervereken történõ használatát célozzuk meg, beleértve a ProLianthoz tartozó eszközmeghajtókat, karbantartó és konfigurációs szoftvereket és BIOS-frissítéseket. Ennek megfelelõen tehát a hpasmd, hpasmcli és hpacucli modulok is elsõsorban itt kerülnek felboncolásra.

freebsd-python

A FreeBSD és a Python

A lista a FreeBSD Python támogatásának fejlesztésérõl folytatott szakmai megbeszéléseket foglalja össze. Elsõsorban a Python portolásával foglalkozó egyének, valamint a külsõ fejlesztõk által készített modulok és a Zope FreeBSD-s alkalmazásával foglalkozik. Az említett témák iránti érdeklõdõket is szeretettel várjuk.

freebsd-questions

Felhasználói kérdések

Ez a levelezési lista a FreeBSD-vel kapcsolatos kérdésekrõl szól. Lehetõleg ne küldjünk "hogyan" témájú kérdéseket erre a szakmai listára, hacsak nem kifejezetten szakmai jellegûnek szánjuk.

freebsd-ruby

A Ruby használata FreeBSD rendszereken

Ezen a listán a FreeBSD Ruby támogatásával foglalkozunk, témáját tekintve teljesen szakmai jellegû. Elsõsorban a Ruby portokon, külsõ Ruby könyvtárakon és rendszereken dolgozó fejlesztõk figyelmébe ajánljuk.

Mindenkit szeretettel várunk, aki ezekkel kapcsolatos szakmai tárgyú témákat szeretne megvitatni.

freebsd-scsi

A SCSI alrendszer

Ezt a levelezési listát a FreeBSD alatt a SCSI alrendszerrel foglalkozók számára tarjuk fenn. Mivel ez egy erõsen szakmai jellegû lista, ezért rajta csak szakmai témák megengedettek.

FreeBSD security levelezési lista

Biztonsági problémák

A FreeBSD biztonságát illetõ kérdések (DES, Kerberos, biztonsági rések és javításaik, stb.) Szakmai jellegû lista, ezért ide csak a témához szorosan kapcsolódó leveleket szabad beküldeni. Alapvetõen nem kérdezz-felelek típusú a lista mûködése, habár a GYIK-hoz minden hozzájárulást (kérdést ÉS választ EGYARÁNT) szívesen veszünk.

FreeBSD security notifications levelezési lista

Biztonsági figyelmeztetések

A FreeBSD-t érintõ biztonsági problémákról és javításaikról szóló értesítések. Megbeszélésekkel, vitákkal nem foglalkozik, mivel azok a FreeBSD-security listán folynak.

freebsd-small

A FreeBSD használata beágyazott alkalmazásokban

A szokatlanul kis méretû vagy beágyazott FreeBSD rendszerekhez kapcsolódó megbeszélések színhelye. Szakmai jellegû lista, ezért szigorúan csak a témához tartozó leveleket fogad.

Ezt a listát idõközben felváltotta a freebsd-embedded lista.

FreeBSD-STABLE; levelezési lista

A FreeBSD-STABLE használatáról szóló lista

Ez a FreeBSD-STABLE használóinak levelezési listája. Ide kerülnek beküldésre a -STABLE ágat futtató felhasználókat érintõ friss változások, valamint hozzájuk kötõdõen a -STABLE használatához szükséges elvégzendõ lépések. Aki a "STABLE" jelzésû változatot használja, mindenképpen iratkozzon fel rá. Szigorúan szakmai jellegû lista, ezért csak szakmai témájú leveleket vár.

freebsd-standards

C99 és POSIX megfelelés

Ez a fórum foglalkozik a FreeBSD és a C99, valamint a POSIX szabványok szerinti megfelelésével.

freebsd-toolchain

A FreeBSD alapvetõ segédprogramjainak karbantartása

Ezen a listán a FreeBSD egyes kiadásaihoz mellékelt alapvetõ segédprogramokkal kapcsolatos témákat találjuk meg. Ilyen többek közt rendszerben használt Clang és a GCC fordítók aktuálisan használt változatai, de emellett még szó eshet a rendszerhez kapcsolódó különféle assemblerek, linkerek és debuggerek állapotáról.

freebsd-usb

A FreeBSD USB támogatása

Ez a levelezési lista fogja összes a FreeBSD USB támogatásával foglalkozó szakmai témákat.

freebsd-user-groups

A felhasználói csoportokat irányító lista

Ez a levelezési lista az egyes területeken mûködõ felhasználói csoportok az irányítást végzõ központi csoport tagjai általi összehangolásához tartozó problémák megbeszélésére való. Ez a lista leginkább a gyûlések letisztázására és a több csoporton átívelõ nagyobb projektek szervezéséhez használatos.

freebsd-vendors

Gyártók

A FreeBSD projekt és a hozzá kötödõ hardver- és szoftvergyártók együttmûködését elõsegítõ lista.

freebsd-virtualization

A FreeBSD részérõl támogatott különbözõ virtualizációs technológiák

Ezen a levelezési listán elsõsorban a FreeBSD által támogatott virtualizációs megoldásokat vitatjuk meg. Ennek keretében egyrészt az ehhez kapcsolódó alapvetõ funkciók megvalósítása valamint további újítások kerülnek a középpontba, másrészt a felhasználók számára ezzel létrehoztunk egy fórumot a felmerülõ problémák megoldására és az alkalmazási lehetõségek megbeszelésére.

freebsd-wip-status

A FreeBSD-vel kapcsolatos folyamatban levõ fejlesztések helyzetjelentése

Ezen a levelezési listán kerülnek bejelentésre a FreeBSD továbbfejlesztéséhez fûzõdõ különbözõ munkák és azok haladásának menete. Az ide befutó üzeneteket moderálják. Javasoljuk, hogy elsõdlegesen az adott témához tartozó tematikus FreeBSD listára küldjük a bejelentésünket és csak egy másolatot erre a listára. Ennek köszönhetõen a munkánk az adott témaspecifikus listán rögtön meg is vitatható, mivel ezen a listán semmi ilyen nem engedélyezett.

A lista archívumába tekintve tájékozódhatunk arról, hogy pontosan milyen formai követelmények illene megfelelnie a beküldenõ üzenetünknek.

A listára beérkezõ üzenetekbõl egy szerkesztett válogatás jelenik meg néhány havonta a FreeBSD honlapján a Projekt helyzetjelentésének részeként . A korábban beküldött jelentések mellett itt még találhatunk további példákat.

freebsd-xen

A Xen™ FreeBSD portjának (implementáció és használat) megvitatása

A lista elsõsorban a Xen™ FreeBSD-re készült változatával foglalkozik. Elõreláthatólag elég kevesen fognak írni erre a listára ahhoz, hogy helyet kapjanak rajta az implementációt és a kialakítást érintõ szakmai jellegû megbeszélések és a telepítéssel kapcsolatos kérdések egyaránt.

C.1.4. A levelezési listák szûrése

A kéretlen reklámlevelek, vírusok és egyebek elleni védekezés céljából a FreeBSD levelezési listáinak forgalmát több módon is szûrik. Az ebben a szakaszban bemutatott szûrési megoldások nem fedik le a levelezési listák védelme érdekében alkalmazott összes lehetõséget.

A levelezési listákra csak bizonyos típusú csatolt állományokat küldhetünk be. Az alábbi listában nem található MIME típusú csatolt objektumokat még a listára érkezés elõtt törlik.

- application/octet-stream
- application/pdf
- application/pgp-signature
- application/x-pkcs7-signature
- message/rfc822
- multipart/alternative
- multipart/related
- multipart/signed
- text/html
- text/plain
- text/x-diff
- text/x-patch

Egyes levelezési listák ugyan megengedhetnek további csatolt MIME objektumokat is, habár a legtöbb lista esetében a fenti lista a mérvadó.

Ha egy levélben a szöveg HTML és nyers szöveg formátumban is szerepel, a HTML változat automatikusan eltávolításra kerül. Ha az e-mail csak HTML formában tartalmazza a szöveget, akkor automatikusan nyers szövegre alakítódik át.

C.2. Usenet hírcsoportok

A két FreeBSD-s hírcsoport mellett még akadnak olyan további csoportok is, ahol FreeBSD témájú kérdéseket vitathatunk meg vagy hasznos lehet számunkra. Az itt felsorolt hírcsoportok

kulcsszavakkal kereshetõ archívuma Warren Toomey tulajdona (wkt@cs.adfa.edu.au).

C.2.1. BSD-s hírcsoportok

- comp.unix.bsd.freebsd.announce
- comp.unix.bsd.freebsd.misc
- de.comp.os.unix.bsd (német)
- fr.comp.os.bsd (francia)
- it.comp.os.freebsd (olasz)
- tw.bbs.comp.386bsd (hagyományos kínai)

C.2.2. Egyéb érdekes UNIX®-os hírcsoportok

- comp.unix
- comp.unix.questions
- comp.unix.admin
- comp.unix.programmer
- comp.unix.shell
- comp.unix.user-friendly
- comp.security.unix
- comp.sources.unix
- comp.unix.advocacy
- comp.unix.misc
- comp.bugs.4bsd
- comp.bugs.4bsd.ucb-fixes
- comp.unix.bsd

C.2.3. X Window System

- comp.windows.x.i386unix
- comp.windows.x
- comp.windows.x.apps
- comp.windows.x.announce
- comp.windows.x.intrinsics
- comp.windows.x.motif
- comp.windows.x.pex
- comp.emulators.ms-windows.wine

C.3. Világhálós szolgáltatások

Central Servers, Armenia, Australia, {mirrors-austria}, Czech Republic, Denmark, Finland, France, Germany, Hong Kong, Ireland, Japan, Latvia, Lithuania, Netherlands, Norway, Russia, Slovenia, South Africa, Spain, Sweden, Switzerland, Taiwan, United Kingdom, United States of America.

(as of UTC)

Central Servers

- https://www.FreeBSD.org/

Armenia

- http://www1.am.FreeBSD.org/ (IPv6)

Australia

- http://www.au.FreeBSD.org/
- http://www2.au.FreeBSD.org/

{mirrors-austria}

- http://www1.am.FreeBSD.org/ (IPv6)

Czech Republic

- http://www.cz.FreeBSD.org/ (IPv6)

Denmark

- http://www.dk.FreeBSD.org/ (IPv6)

Finland

- http://www.fi.FreeBSD.org/

France

- http://www1.fr.FreeBSD.org/

Germany

- http://www.de.FreeBSD.org/

Hong Kong

- http://www.hk.FreeBSD.org/

Ireland

- http://www.ie.FreeBSD.org/

Japan

- http://www.jp.FreeBSD.org/www.FreeBSD.org/ (IPv6)

Latvia

- http://www.lv.FreeBSD.org/

Lithuania

- http://www.lt.FreeBSD.org/

Netherlands

- http://www.nl.FreeBSD.org/

Norway

- http://www.no.FreeBSD.org/

Russia

- http://www.ru.FreeBSD.org/ (IPv6)

Slovenia

- http://www.si.FreeBSD.org/

South Africa

- http://www.za.FreeBSD.org/

Spain

- http://www.es.FreeBSD.org/
- http://www2.es.FreeBSD.org/

Sweden

- http://www.se.FreeBSD.org/

Switzerland

- http://www.ch.FreeBSD.org/ (IPv6)
- http://www2.ch.FreeBSD.org/ (IPv6)

Taiwan

- http://www.tw.FreeBSD.org/
- http://www2.tw.FreeBSD.org/
- http://www4.tw.FreeBSD.org/

- http://www5.tw.FreeBSD.org/ (IPv6)

United Kingdom

- http://www1.uk.FreeBSD.org
- http://www3.uk.FreeBSD.org/

United States of America

- http://www5.us.FreeBSD.org/ (IPv6)

C.3.1. Fórumok, blogok és ismertségi hálózatok

- A FreeBSD fórumok a FreeBSD kapcsán felmerülõ kérdések és szakmai témák megvitatásához egy webes felületet kínálnak fel.
- A Planet FreeBSD honlapján fejlesztõk által vezetett tucatnyi webes naplót és hozzájuk tartozó RSS feedeket találhatunk. Sok fejlesztõ ezen a módon készít rövid feljegyzéseket a jelenlegi munkájáról, az új javításokról és más egyéb terveirõl.
- A Youtube-on keresztül elérhetõ BSDConferences csatornán a világ minden táján tartott különbözõ BSD témájú konferenciák videoanyagait találhatjuk meg. Segítségével megtekinthetjük a fontosabb fejlesztõk által a saját munkájukról tartott különbözõ elõadásokat.

C.3.2. Hivatalos tükrözések

C.4. E-mail címek

A következõ felhasználói csoportok nyújtanak FreeBSD-s e-mail címeket tagjaiknak. A rendszergazdák bármilyen visszaélés esetén fenntartják a visszavonás jogát.

Címtartomány	Lehetõségek	Felhasználói csoport	Rendszergazda
ukug.uk.FreeBSD.org	Csak továbbítás	ukfreebsd@uk.FreeBSD.org	Lee Johnston lee@uk.FreeBSD.org

Függelék D: PGP-kulcsok

The OpenPGP keys of the FreeBSD.org officers are shown here. These keys can be used to verify a signature or send encrypted email to one of the officers. A full list of FreeBSD OpenPGP keys is available in the PGP Keys article. The complete keyring can be downloaded at pgpkeys.txt.

D.1. Tisztségviselők

D.1.1. Security Officer Team <security-officer@FreeBSD.org>

```
pub   rsa4096/D9AD2A18057474CB 2022-12-11 [C] [expires: 2030-01-24]
      Key fingerprint = 0BE3 3275 D74C 953C 79F8  1107 D9AD 2A18 0574 74CB
uid                           FreeBSD Security Officer <security-officer@freebsd.org>
sub   rsa4096/6E58DE901F001AEF 2022-12-11 [S] [expires: 2028-01-24]
sub   rsa4096/46DB26D62F6039B7 2022-12-11 [E] [expires: 2028-01-24]
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGOVdeUBEADHF5VGg1iPbACB+7lomX6aDytUf0k2k2Yc/Kp6lfYv7JKU+1nr
TcNF7Gt1YkajPSeWRKNZw/X94g4w5TEOHbJ6QQWx9g+N7RjEq75actQ/r2N5zY4S
ujfFTepbvgR55mLTxlxGKFBmNrfNbpHRyh4GwFRgPlxf5Jy9SB+0m54yFS4QlSd0
pIzO0CLkjHUFy/8S93oSK2zUkgok5gLWruBXom+8VC3OtBElkWswPkE1pKZvMQCv
VyM+7BS+MCFXSdZczDZZoEzpQJGhUYFsdg0KqlLv6z1rP+HsgUYKTkRpcrumDQV0
MMuCE4ECU6nFDDTnbR8Wn3LF5oTt0GtwS0nWf+nZ1SFTDURcSPR4Lp/PKjuDAkOS
P8BaruCNx1ItHSwcnXw0gS4+h8FjtWNZpsawtzjjgApcl+m9KP6dkBcbN+i1DHm6
NG6YQVtVWyN8aOKmoC/FEm1CWh1bv+ri9XOkF2EqT/ktbjbT1hFoFGBkS9/35y1G
3KKyWtwKcyF4OXcArl6sQwGgiYnZEG3sUMaGrwQovRtMf7le3cAYsMkXyiAnEufa
deuabYLD8qp9L/eNo+9aZmhJqQg4EQb+ePH7bGPNDZ+M5oGUwReX857FoWaPhs4L
dAKQ1YwASxdKKh8wnaamjIeZSGP5TCjurH7pADAIaB3/D+ZNl2a7od+C1wARAQAB
tDdGcmVlQlNEIFNlY3VyaXR5IE9mZmljZXIgPHNlY3VyaXR5LW9mZmljZXJAZnJl
ZWJzZC5vcmc+iQJSBBMBCgA8AhsBBAsJCAcEFQoJCAUWAgMBAAIeBQIXgBYhBAvj
MnXXTJU8efgRB9mtKhgFdHTLBQJjlXeQBQkF3u+rAAoJENmtKhgFdHTLOVoQALS3
cj7rqYkHiV4zDYrgPEp9O1kAyGI8VdfGAMkDVTqr+wP4v/o7LIUrgwZl5qxesVFB
VknFr0Wp5g9h0iAjasoI5sDd6tH2SmumhBHXFVdftzDQhrugxH6fWRhHs0SaFYCk
Qt5nFbcpUfWgtQ35XTbsL8iENdYpjKXsSFQrJneGSwxIjWYTFn6ps/AI3gwR8+Bn
OffEFdYugJ049O6Vu6YBFJHrnMO7NbF4v95dVYuLtpMIaXWM+V9KITmhaBzFz5fM
Q7UOzcLlbxOYKNIWcp8QQk429mayKW5VUeUExUD1ZzBHn+P6ZG7QTMDu/RmBqiHo
ewCMVz4n9uXT5BiOngE4CvS0WQwHzK+k9MLpG2u/Bo9+LT0Ceh9Ou1rfU5+0tRwl
GyOFFjf3INS7I7gkcAwxQ7dzDItN/UQPZpg8y9mABU2x4enz0AvTnb61d/1dnTEr
tdNgU433he0ZnD1HurZCjBEWC656wv6iMdWcD8gjhMbmEpPmjvXcYlTO6zhEygSM
DiwdQCWK2W4++YJerA6ULBi3niNWBpofOFH8XylV56ruhjtHCo7+/3carcMoPOJv
lVZ1zCKxLro3TRBT15JTFBGqblRyTopFK3PuxW//GTnZOtpQEOV6yL4RAXcWeC1d
1hb5k/YxUmRF6XsDNEH4b08T8ZO8dV3dAV43Wh1oiQEzBBABCAAdFiEEuyjUCzYO
7pNq7RVv5fe8y6O93fgFAmObXVYACgkQ5fe8y6O93fiBlwf/W8y1XXJIx1ZA3n6u
f7aS70rbP9KFPr4U0dixwKE/gbtIQ9ckeNXrDDWz0v0NCz4qS+33IPiJg1WcY3vR
W90e7QgAueCo5TdZPImPbCs42vadpa5byMXS4Pw+xyT+d/yp2oLKYbj3En4bg1GM
w71DezIjvV+e01UR++u1t9yZ8LOWM5Kumz1zyQLZDZ8qIKt1bBfpa+E0cEqtNQWu
```

```
iGhQE3AHI8eWV+jBkg5y2zHRIevbWb1UPsj43lgkFtAGHk9rrM8Rmgr4AXr531iD
srBwauKZ/MElcF3MINuLH+gkPPaFHw/YIpLRLaZXZVsw3Xi1RNXI2n2ea29dvs/C
Lcf1vYkCMwQQAQgAHRYhBPwOh4rlr+eIAo1jVdOXkvSep+XCBQJjm14FAAoJENOX
kvSep+XC0DcP/1ZB7k9p1T+9QbbZZE1PJiHby3815ccH3XKexbNmmakHIn3L6Cet
F891Kqt9ssbhFRMNtyZ/k/8y8Hv5bKxVep5/HMyK+8aqfDFN0WMrqZh0/CiR6DJh
gnAmPNw/hAVHMHaYGII9kCrFfPFJ02FKoc81g9F08odb7TV+UlvRjkErhRxF+dGS
wQoO0RCbf0Z1cs7nd0Vb2z4IJh4XMxBjWc/uQ2Q9dH/0uRzwpAnR4YX+MG5YrX7Z
zBvDyR0r76iQwRSDKgioNgkr6R3rq1NZGdaj+8b0LzdOqtzKJ/eupDe3+H67e/EN
qymtreGjrubpiU9bKvYArisUqhE5KtguryvR6Qz9bj87nPg33DT3WWGVrwFRxBox
dbWzjQFv0wug8m4GAwVF7fPR5/eW7IHw8zvgn0vSPcZz7MZ4e6Y5jN4kA5/xWJYZ
Sps54qQWB+FA30unIXN68KqdIzONIbtaY3W4/JjJUCm4T+wEjKaH+wJX8w1DMjlg
mkTmGh/UrTyC1vXbPgk9Sy3cRTICR1T9z7W8UlmTtnKrUklrjlFR7SXzrEXzLGOX
Fm+NEHpHNXqzcm6c3QfzY/yQ9HSAQ/t7SUQ9caRePbDz3/msyPxtGFor9roQv6VN
wRXCyRgkH4Y5tPhJAQ8G/FxX+VXFb93QL0lfelb23/BBu6cUwW63SRn5iHUEExYI
AB0WIQQeB2Johg/5/ikUnJwDU9SVF1S13AUCZISO3wAKCRADU9SVF1S13NnqAP95
LA1Om9XSAKl76VtV+L3JPDdAwIdbNa0OsRT4Wm7U3wD/YoFrdHXVHHQFKwYeUUhj
XZcxnZLe9Ixo0/JP+RVFVw6JAlIEEwEKADwCGwEECwkIBwQVCgkIBRYCAwEAAh4F
AheAFiEEC+MydddMlTx5+BEH2a0qGAV0dMsFAmluhFoFCQ1lXf0ACgkQ2a0qGAV0
dMv8ixAAhYn3uxnjTCBZ5BJ6FMdwFXgGZMZVnL1qEoHA7HWB0ZoQlu29sFzBX97x
rkVZfWiU82V9/PYyHh4EY0hN9oOJMDDhSgi+DfdoEablIcIXR9uBBL/Snnak6uEF
gOD6VFH1T3o7iBmS/ixee2YFXq5QxAwEiANSAsxDm2rlyztPc5xM76fz5eEcgp1m
UDI7qFjoEkJVvRraJGVy3AUh4sr/94VAXPuCa7JnC6a8j9SD8XtzqIfcuqNBXXfA
m4ji4fN+KcDHtZPdkNLDBu1DEXQkOR/1S++NmXHYCEdQEuVOwLVsy6vQqgc+unju
Xh9sOrdiXUWylHh2hvgmwkjhMGOREiIRwNoA09m0OKZ2jo2dSlhNdVFRD5yhD92t
X9BarQbbjQtyCvoWnCJMFfw/vDnxfDdUdV3GxLhaOWvR8ItWQ7ts8tEZSjR/G76/
fHj1JRNuHRdhwype6ehieewJSMyzByhT0A3Njv3KLI4nwQj//MYzB5FJ9uOmKLcG
w7PB0yVjW4oRbG62+Qv6IQybCyrAL+12qgbUXbMsTqMwmsmdgtsqr/7OJNo3PBxX
2rUEio/mez8wuzNB0mJtlMynoMqXtbFX7AAXHk16fOGd6L6guW8+IhPedT3TvrjS
hqiQVvjOhM6rEYZ/YUfcIdkWJVpmE6+MRPVUufyMWeHaWqsKXoa5Ag0EY5V2yQEQ
AOqjzPpMUCGu8eElXnAd2PruC6hi+lc/yC90KqizxIuW6qLQBaAkTCWq7suYpDqo
ygn7YM3rL50S285WAECAXrcst/cVAqr0UH/e6p4iJCUIiXcfjd/wq20RnN/+VuvL
hjpCFLY5czfVS31D7Uh9MbC+zUTz8nVTiNCsAao0qSdfJDIzB4nSO+9xIsme/dLs
I5QlU5PdxOBV6HdEhCUXOoratJCbKAOlLxtPwyMKxmv4oZ7Mqlt10peKjhpBb97q
cIzJhHxujQZD0OmzIA6xoQ2eSCGdxCEDsZ09kr3Esw1AwKnQ51xmWpFWNFk6627M
1bo8+hzOz81CrTZhYrgE+1JXv6V1L2A9lMsimdE1BHNycDS+dBOpIB9qxXCwAab4
ykfvNxox/ZPDUrTy7v7mDI5uDNTNCYYsKCj1UidycOKzSziB9Oa2uvmMJ5XstgNB
f7Z8Cky1dtVd4ol6bU9L5nos9tbYeSXFi4bmcWB7AJiVCMq6N+LBbUKWGLglB4TU
1qhttpqv31X9V6ges5gARY/RuRTKsVyhwsn7SDcqmNKRy0im2AYakwEp7hT07ula
hOSLxjP+5hCf+nSJlwbxJ8ozwjjbzeN2yLlJSI00klkIFBNUDt3wzFRW/n6qlf+/
lepgzekfNrYMtfPB8AT07Z2A3U4xlgiV346dZymbY/EjABEBAAGJBHIEGAEKACYW
IQQL4zJ110yVPHn4EQfZrSoYBXR0ywUCY5V2yQIbAgUJAgIpAAJACRDZrSoYBXR0
y8F0IAQZAQoAHRYhBLYVJ36BCH33XIGDO25Y3pAfABrvBQJjlXbJAAoJEG5Y3pAf
ABrvuBsQAOlQFPXhx6whO4yw5ZizIS02YHhSVMVYKS2T9jPIKi1qxnEiEw9eKH0b
W0Oj0TEhZPyM2NJID7DRWK5r8+KsMu8jwm1fUmIrefAx6fCVfCWRECT1MlbL3jhh
6AcX/nK2e3Bn8vgExhzczO3JlvD6wPCc0FkpiY7yDB9ihu1+gbE5Hg6dvftttRXD
rbEdAifbNp9KYxDigxdlOb0S14hjCBysLWH5Su/khcIlkeuqZcI8TmDldnUb2OqT
CVpFhaNwsPSrHBzmb0s2sXo4FL03pLsOdwhi31W6kjk4KvW5FKrOpoEwUMKVNMf5
0DHdvonUoUHRSIc/cV5NqUWHwvc0T5031qk0CCRRa/+/iij/p2RG7c1mx7ZECj+j
ZfmvjSqT+WHJ1BFlNJMWyK4fdVRZWyCaoAecdbukwzDwUCUqHJFIWeFtbut7SOPx
cwg7sbnKNApAKdi491dvH75s/U/OwRYO/2P+ymHlqtyix2jq0ReSVYcQPXswQ8i2
ifX41F+xTSl4RWCBBeXB1Nxk3+HsV4Jnnp1zAJZ0KlKW/oJxbNFdI1TImkpr2p8i
```

```
oFf+aiePLvDkgeaG8vABgjoihPXWHVAMR8Z+GvBY/A6OdexpibkTvC/zDr0/Exs4
lsylZKDwvvFbctcpHVXBeCBQLX5vfLrsTkaCLWF/SV9OdMykvYKU7ZAP+gKEwhp+
HPFuOHZbOBhqFUdkfeCkdzX/QGdzTuz349roRhgz2vRfN7MtbtuzA6NWWhEWt5Dc
UgX/Y5I3Q4Z2bt3JiXQ6WJMgMMOXAr+XxtxyRRykc1HV3DQ/cq8OWYubNnIbgebP
NIFr2OIWKsR9yDaucZzpmLfzaMZUAu5hWmU9fIw5SIKGnQABBnNMhilfD+CkETp6
baTvjTK4rpaobjJdeCTrsWgfXRNC8x3hDvcrjPD70MylOGVQdx8GYChWJnCKXsLT
GX7KwdfxkjclTyzWvdcCemp0eLhamLGb9y1dtWdNIDcVCvZJy0lipHVUdFYYxb4i
LZJANL631tlPM6AA8sO1/L4mqEGnAIHVrUQd+2QkSiOl9mKlpgaR/fJz683BR5Qe
n9ywX0JPtBupqPW3t9VbO/uNxUqlHCeAhPi9NLOpujpYLfgW5QAfS3u0nkp5nrbk
CoQUua2q0Oj7J0mFmtWtcE1c9+THmFJVb8j2G9yQw3ADe3Qp9ALazP5nVDVri8NZ
BhHK1/KuBmRYZtcyfqXUnKoiiWAlm5rHaRiztW7e3wqm2oJu/RkEAagybutEuBWh
2Ej2+gDxjEKKtIKGu54lif4kqTwwjKTcN1ekGihwwgCMUkBSBeNXk1ClkzLFHwES
JCcFwdEgpVYQTKFsu0emYISyco3IpUajGzfUiQRyBBgBCgAmAhsCFiEEC+MydddM
lTx5+BEH2a0qGAV0dMsFAmWFy28FCQPyfaYCQMF0IAQZAQoAHRYhBLYVJ36BCH33
XIGDO25Y3pAfABrvBQJjlXbJAAoJEG5Y3pAfABrvuBsQAOlQFPXhx6whO4yw5Ziz
IS02YHhSVMVYKS2T9jPIKi1qxnEiEw9eKH0bW0Oj0TEhZPyM2NJID7DRWK5r8+Ks
Mu8jwm1fUmIrefAx6fCVfCWRECT1MlbL3jhh6AcX/nK2e3Bn8vgExhzczO3JlvD6
wPCc0FkpiY7yDB9ihu1+gbE5Hg6dvftttRXDrbEdAifbNp9KYxDigxdlOb0S14hj
CBysLWH5Su/khcIlkeuqZcI8TmDldnUb2OqTCVpFhaNwsPSrHBzmb0s2sXo4FL03
pLsOdwhi31W6kjk4KvW5FKrOpoEwUMKVNMf50DHdvonUoUHRSIc/cV5NqUWHwvc0
T5031qk0CCRRa/+/iij/p2RG7c1mx7ZECj+jZfmvjSqT+WHJ1BFlNJMWyK4fdVRZ
WyCaoAecdbukwzDwUCUqHJFIWeFtbut7SOPxcwg7sbnKNApAKdi491dvH75s/U/O
wRYO/2P+ymHlqtyix2jq0ReSVYcQPXswQ8i2ifX41F+xTSl4RWCBBeXB1Nxk3+Hs
V4Jnnp1zAJZ0KlKW/oJxbNFdI1TImkpr2p8ioFf+aiePLvDkgeaG8vABgjoihPXW
HVAMR8Z+GvBY/A6OdexpibkTvC/zDr0/Exs4lsylZKDwvvFbctcpHVXBeCBQLX5v
fLrsTkaCLWF/SV9OdMykvYKUCRDZrSoYBXR0yyOqEACitDvbkbfjaton6izr4T8Q
U2yvhJHkf4B6KeVDbKY1J4784OxXp2bJgPeF53SYBe8gm3YHjp8ULh4A/19U4hsw
yE8ymcm5nIs8OLyBdxkuBZJGEnzxH3woiyYqWH7991kzhEjUkuMgKLuTI1HiO0oL
MuPQNhUHOnWafSVPC0XO/tIL12OmoUuc7ligY9Z9AceFjTZOuHamixHAAc6hpxdI
W+yhC/qTpc2VK0niWeuQfq3453iRTf9MnR5Beztl3ZYRWcx7UiFuKGwZwBibNnNm
Us6GyQcJ5UTa1oeJcLqHi0Lf/r0jXo3wgJq7EZjjVyU+GI2ZVoDOaS6c4/OvLm62
XoeSlnn/dQxUcjUki+x8lb69IxSF1xAgsC/oNtFZYd5rHdlnqIBUYK0lLtSCXBkz
VeivSiQa0hL5on8LDu1nw2bXyW61yt/YxVb4FanMxAqdYVBhOfU0RaPNifH01rbb
4TwC9bTZN1LQ1KI/Swb/SruUE0RyT28fhYRtsReS2PnUODghJSFDJbwFbBZf6RKI
16q1xqKRRvxIWPm+lMOi1NLOKR9P+OKy9HmChMw0UJUcVl1cJ2xtRl3wi5t6AA6H
oNv/TrLeYVgMR9wYmKlpvjTQ5jTdrbHD1XP5jGsp8QsJMGja1m/7cryReCpcVxvI
meReaOdgz+zDmQqq3O5zuIkEcgQYAQoAJgIbAhYhBAvjMnXXTJU8efgRB9mtKhgF
dHTLBQJnhEEJBQkF0/JAAkDBdCAEGQEKAB0WIQS2FSd+gQh991yBgztuWN6QHwAa
7wUCY5V2yQAKCRBuWN6QHwAa77gbEADpUBT14cesITuMsOWYsyEtNmB4UlTFWCkt
k/YzyCotasZxIhMPXih9G1tDo9ExIWT8jNjSSA+w0Viua/PirDLvI8JtX1JiK3nw
MenwlXwlkRAk9TJWy944YegHF/5ytntwZ/L4BMYc3MztyZbw+sDwnNBZKYmO8gwf
YobtfoGxOR4Onb37bbUVw62xHQIn2zafSmMQ4oMXZTm9EteIYwgcrC1h+Urv5IXC
JZHrqmXCPE5g5XZ1G9jqkwlaRYWjcLD0qxwc5m9LNrF6OBS9N6S7DncIYt9VupI5
OCr1uRSqzqaBMFDClTTH+dAx3b6J1KFB0UiHP3FeTalFh8L3NE+dN9apNAgkUWv/
v4oo/6dkRu3NZse2RAo/o2X5r40qk/lhydQRZTSTFsiuH3VUWVsgmqAHnHW7pMMw
8FAlKhyRSFnhbW7re0jj8XMIO7G5yjQKQCnYuPdXbx++bP1PzsEWDv9j/sph5arc
osdo6tEXklWHED17MEPIton1+NRfsU0peEVggQXlwdTcZN/h7FeCZ56dcwCWdCpS
lv6CcWzRXSNUyJpKa9qfIqBX/monjy7w5IHmhvLwAYI6IoT11h1QDEfGfhrwWPwO
jnXsaYm5E7wv8w69PxMbOJbMpWSg8L7xW3LXKR1VwXggUC1+b3y67E5Ggi1hf0lf
TnTMpL2ClAkQ2a0qGAV0dMsNxRAAsuW1aLh+hgydW+iH6DmdQRMESsB1kE02kO14
62TAQaziIAvNoxw5h48xvyEnrDA8d+9IDMyxdrLmAbndUlSveMa9+EPiGHwr6VTy
```

```
FL8nA5F7DcFi4mjEyGKe18JcaAlYUtvHgWH6EjiX2iSXpsrJFEhtfFNolZ5sp9LF
I6hOBihSJxZK4sbMR7Q6IkDuAVpTFLiejBRlsXpFvTGL6040CtxbL5cqkVMYP38r
FMTuc3pGGJA4wb5EC1dGjUi6XjbYH7kuCAFyXqV9eQQP61x7K9W8qnXW+weCIMKf
SX7AcCtH1jXBAM6lqpPrh6amc+/rbg2eNA7DmgJnEY4apIcDB/b4khRMga2ozeGW
WyIvOaVvR2R7ALQ+Rgut85cM+4+Vl2PHmOzW/yYdHVb5REQItFR5COb/mGUqYhkC
tiV3nXo/K0uOQKu5SBbNzLuNvuwdn+Eimxjl18VnrGG7sjtUa0MLmtr62GiEVrhr
Dqa/biHp8LdWkAQjLZ4aTRh2XZiggaVFZHmkw3ILPyKKM21UXdM0YRk3TGVK8ODQ
y58ebPS4v9yYT9gUA9UDkDYeGcF2qjoDPVNvcG6H8jCSsPRl1KZwtqqITCOSAIAP
I4Nu97kO6nbOyQpYlwjd1MhvVXnP66mHSvmqaxbNGX1mF9B/yERkBkooNZrKuJSv
BTC2J1qJBHIEGAEKACYCGwIWIQQL4zJ110yVPHn4EQfZrSoYBXR0ywUCaW6EcgUJ
CaGkeQJAwXQgBBkBCgAdFiEEthUnfoEIffdcgYM7bljekB8AGu8FAmOVdskACgkQ
bljekB8AGu+4GxAA6VAU9eHHrCE7jLDlmLMhLTZgeFJUxVgpLZP2M8gqLWrGcSIT
D14ofRtbQ6PRMSFk/IzY0kgPsNFYrmvz4qwy7yPCbV9SYit58DHp8JV8JZEQJPUy
VsveOGHoBxf+crZ7cGfy+ATGHNzM7cmW8PrA8JzQWSmJjvIMH2KG7X6BsTkeDp29
+221FcOtsR0CJ9s2n0pjEOKDF2U5vRLXiGMIHKwtYflK7+SFwiWR66plwjxOYOV2
dRvY6pMJWkWFo3Cw9KscHOZvSzaxejgUvTekuw53CGLfVbqSOTgq9bkUqs6mgTBQ
wpU0x/nQMd2+idShQdFIhz9xXk2pRYfC9zRPnTfWqTQIJFFr/7+KKP+nZEbtzWbH
tkQKP6Nl+a+NKpP5YcnUEWU0kxbIrh91VFlbIJqgB5x1u6TDMPBQJSockUhZ4W1u
63tI4/FzCDuxuco0CkAp2Lj3V28fvmz9T87BFg7/Y/7KYeWq3KLHaOrRF5JVhxA9
ezBDyLaJ9fjUX7FNKXhFYIEF5cHU3GTf4exXgmeenXMAlnQqUpb+gnFs0V0jVMia
SmvanyKgV/5qJ48u8OSB5oby8AGCOiKE9dYdUAxHxn4a8Fj8Do517GmJuRO8L/MO
vT8TGziWzKVkoPC+8Vty1ykdVcF4IFAtfm98uuxORoItYX9JX050zKS9gpQJENmt
KhgFdHTLozcQAJCdYMPo1SPgxxRI40BCD4ekuA6ZMzdTuuso9XTTLf3moh+eu0Y+
Pv6ISOFI93g09V34H3XnuaOnA16DyApO6mCPZ58DunjRRYHv4wxbeff6vw9twPQb
bcdNgI2ZXSKjUef4FygkpUhy6075m6zoUfcsR/22Ggzd8Qw3chfoqiBZs7SlGwEH
sxkB+dJvgCMwYaLGqS9lNbHSEFnETb0ihFSHKDrK6pYrFnfsuArJN5owuTN+dzim
8g0agB3Pzs4A6+9Hpt9sN78VEY+ZzFhQD8knEgqSNbFIaNxsG5/L8/RcXuNyPvvz
5MIkhfacOX37YFfAhCORfN3PaVWYQ7oDnv8Nh69i9SkxXKUcwX0/iYaim8pwwM6a
HXhTlqBgKR5b5Hcx6Cwx1GTRUpdbJYThmz7LQgzjBIhmoGbxR3vyJjSyPGBZa3Ih
jAETYzMt7j0qGPiJUx7kcHeK7apAH1xB4DnfuPLSC1GjWptb31A22wjWy/75SJk8
0FoyYKPj5RqiQF+5CcSLRwTJEEi+9ggM8mUJiFNUOtfZgATgmCyEKG4fL1N4iyrj
fBha11/EO3LcH4t0rj5i8SkT+w8A0bgpOliLhnKpFQMy+cRewkzNIS2ynGlHqnKA
biuBEr1ji0iHNHczTMuqSYwraVHis+098D+1me5+9SaDp8yEIKgVuTUAuQINBGOV
dwcBEADKRVXM2aPWTzbIFu/dVFuaOsJ653dZyDvkuK3N+6tkxM/ChhwrsaOTcH32
drei59OASOeKW2qYNe2+2tIT5Gt+xxZC9vJ+FcjHcdArAJt0BH1LHSuscCldkrnG
HlXK1dbqXDeD3Vw+7LtSO/wOAllxSRtOO8ydGfPZSWYkBIgKu4IbxKIpwtLJvtGu
mqP6HKgoFgijNcgjzssyV5Xka79vUkZ2M4jEfqtlNMfcYPV5ehjMFJhcGyXPZzkV
byx+321Ov3bvBIKAGK4JRThWRv00qJUhlkUjImr89CUg/4vqxDeGiE+FnwXvi/Vs
tQJqzcYgDvsypsx0LtyUsQ53K/svoXHfltIKsNdBaCPY1i3C+5EfumtIjo7iUl0q
Katpdw2n4j8bQhZq+AhkIcwRzOnuP7/zTXdxdF6VWyekUKYe5ZH738/Ktpb1JR4X
sydKdcV5oX3/Qh43VTIZgJs+fopgQF5NpqLbTXym9KRRHp9cWNyv1DwOIPqD7Eo+
PeTLLyIPXiDAnAt/89EhFDFh/7Lcsicz6n6cnxsGXk3CI/b4NqPkZxr5nC1WAPxB
k6D5PdOZL/3Cu96MvG2cjgdMzIO0/T2loC1DtLKo6qNDMN5JxckoN6EV5XIpT0L3
z3V9Cb8C6XPQ1pa9ftRpAlydPJVroCXPegiULqSkW9SGwtKFFwARAQABiQI8BBgB
CgAmFiEEC+MydddMlTx5+BEH2a0qGAV0dMsFAmOVdwcCGwwFCQICKQAACgkQ2a0q
GAV0dMsMLQ/8CA5hHB3RZchUU6JlAY23itrs3GSIZJIC+fJXOt1D5XuYidhQb5mE
vtK0Eud7FRLHHJ+Sn5DsE68ipZb1m5dCM7oqn2sMQC3H1qX2SR80zEpwx34F7zjs
hTEK7qkQwBwK+zzQtAp209LBBdmRc2nfDPXy6rZF1MeJoBAjRieAlaoO+SmPLlM7
/wZkcBH+fOxU+f4gCHneTIxQdlRQ1/RgAXL002x8QmSMZ8Zd3LNeUsU9JQKH2drn
lOpFmFmUlxXmhDP+MpjycLNppldAJGyYIS1wp/TMd51+jU2WfR2q267vkc1t/x4N
w2XdCfb5KPD2nz9JdTVrYMl++Aqr04I808V056q0V83HMyu34XAtruItKaz7V13M
```

```
sou0sKaC4pGmQixRYVJjsfpkuU47YBchWQTcZzu4gMJM33E3AvbD6FmhZokH+quR
CGMxvlkKZ3I/NJsZdx1jj+qBY/4WSUWLHinjJLycy06PO6xXTOgrDKy8BE0envml
jNG7TxYpda11QAjTo7MtG8U1fDiRH8sZ2Jso75wbwdJHYmlRA3VYcYEqsywF+JoF
2uPf1ejP7szK/p9/wNSXZOFen1ZvEuXBSF8aFGrd3AeMRmk/Tlg96NTT6+nkvVaN
BLCZUZfNUhuvH0vkLZNn19u0+vGBuAGt8zH2cVLFbhS7U7kG0QvnyaSJAjwEGAEK
ACYCGwwWIQQL4zJ110yVPHn4EQfZrSoYBXR0ywUCZYXLqgUJA/J9aAAKCRDZrSoY
BXR0y6DtEACbExPX5JXm+2MzxLj/Rid48fDxnVpu5/vFDXa9uJEuoDl1VHz2a1mu
UqNtYXCMGW+D2Q6h0lUkJ0XIioUpxu4KZWDzRmePhVmy6/5gUWGq45FRS1Lgj3sU
iI8VdSkhiZ1rPWE1QCmQM++RqQjC+ow+m4gjkAkPOsKjnVr2SGUfGL7q68UeuAAQ
dK427GhvHDTlF9XMAH3S8LYX+uHUvtpt5ajRgx5D+XjS9ESMQuYF2BKdJb5pTB92
bjuVq3x+9z0w4PPW8V3roaxZkhMH+MVputN6IvP9qkL0ff+429KPre9TULlq1iBT
jWVloGQVBrKPrsk914GOGKihkd89kMyh2JyejO0r0mKaku0bXMXAlxdBxjXxEvHA
ILSlaRbW7UsF16fDvKC8o5lSDvzY1KWyzhnRSLGq7HsyxVfX9n6metyd+IclEoPC
62FVdXwOQyVKXm21+IS6XZNDDLYBuNwcCiBpSU6S+UZEP7kFN2FtpGoq8u6PtYpg
w9E0c0MbXY1TiOb55UzXHyWSCHy8eHawAQNdjr6V93Xz1etkfvAQJSRU+N9AhugC
EiSAaiURb7opJp2RlLxB5jlLPxS222l0z5CnO7UdliMJY0bzFfkJPOHFnF+mHITL
JTLogzNZrZUGYoN+FhxkhPswlp1wLJSB5UsKMe6n3TtCs9l53EmyAIkCPAQYAQoA
JgIbDBYhBAvjMnXXTJU8efgRB9mtKhgFdHTLBQJnhEEuBQkF0/ICAAoJENmtKhgF
dHTLJtUP/1dPcTrrUm1lOOCkbq3ZxirJOegxZal7YxmCagz6zg6V9zPCA39jLkyD
dsnzzL/fOGwGzgPUS+xGhlZUFFvUFj1ETLGShBUJY4X8USTvSaBg2CO7kQ50hIul
o1b4FJOToiIdChCOIxHGiws73iBHLMK1Q0gJ/u5xr3sD/Ea8e6jFDbywJbcYjMnx
l53mr0bN7H7rM3W/S002PJQlNMa7JWl6673wpGkW7JBYt0t9SPtWQf3sLgIR6n3N
d3kQYz8EDmpyRXqQMz+BheNX1UDfRrbLAEwg/nVejhYWI31iWziHD79u+8ArxiBb
nDPV0Y25AiGBUBOVvtTv2nLdzm49bAfdAvnFToAp4N+2Knzm2CxdfUROPmuzVAIf
xsR4/vE8VMr762llsnzakzA4xzEoTQ4xtKksxV+738lKZnwH8/umne/GpLPSzH/v
OohpLwYVxbO3M1b+D1lmFdflkoTVNmPcjGh986VlGkRHQdOxymy72uGNYnKllcfs
+sbNz77Km03ov5qF4mZO6IaWwZtIbSZZIttgEWKxl0iusOUULQmaAVtAxI+STQY5
Cul2wFt1RbgANSecC0ts9HiEeWnT/v0sUFG06vmRYjhnQT9YP+/N3u8xHcgNGHwN
ilPTnS87qVIsf8fiWHMZq2V53UjDm3dCIljocF5YGq2ySTEXjjkYiQI8BBgBCgAm
AhsMFiEEC+MydddMlTx5+BEH2a0qGAV0dMsFAmluhHIFCQmhpDsACgkQ2a0qGAV0
dMs5ag//V7mahuI9VikX1lG599GSjm1cXeDrZmGtamz9hYE7EzGlyxf5WQn83Sxu
2kNe9UbaLcIpKzZ9CWcb7NfC6odUIzJsO21qSKNuoP7FV9472JTNFMhaj1+o1P8b
v1nGOU4H9IH/y0Wqkuteef3t4le0iCtePMU3vXgfUFAmm/VF/tgmJQb2hLkXFNCQ
QwK//jjm9QHajaU2wEpvskarsRGfGBintBbUqUUxyPgirqAC8dFAWrjefmGePWEW
JfRLql39II+xbjEzADBB8yqnxLuaUQfVF7t2oAOfp09dP++bCua7FO1WfuFoa1Yw
um84/XszGgkxDRFkBxabpMCurPWgyhkGto4TqKdcWTMl4S1rTxt4Bm/6fqMn+04z
/AiXgU/iIbEKhvQUxsTxyNBLPFIRD7VmrqrOr3wkHgGHJgP0656ABq5J+T4mtFhX
uL0U9nGSm5lV9Hl1ana+Kcs4DqzgQO3QJk/SoulBDZ6at0pK2Y016JrvY51fYpx6
a/IOaXE2JRi8itVlpciDTjSOJs4eNlUx0C3b/LdzU5LZqHtBea1xWxlKPn0VFjYr
YNML6o5B6AL05AcySZj8kNZTWE2ZfOR2usLdiB8b+u2F7WGXZBiQYSnr75WsN94S
nYwO46v5SlzGqTwEN/UBfTkKPklVNgeOMn7fgjziEtMmtBlb7ws=
=ijsc
-----END PGP PUBLIC KEY BLOCK-----
```

D.1.2. Core Team Titkár <core-secretary@FreeBSD.org>

```
pub   rsa4096/4D632518C3546B05 2024-02-17 [SC] [expires: 2028-02-08]
      Key fingerprint = 1A23 6A92 528D 00DD 7965  76FE 4D63 2518 C354 6B05
uid                           FreeBSD Core Team Secretary <core-
```

```
secretary@FreeBSD.org>
sub   rsa4096/CABFDE12CA516ED2 2024-02-17 [E] [expires: 2028-02-08]
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGXQ1o8BEAC+Rcg8cmVxuP17Vu+q5KgCx/XiulQuqKXAqqBlYCH2jqk6DINP
yFrREGBhzd/qNmlAYEahQ4Zgl0bUZNTrZVDyzicOvPP0jH+KSTQwRs7NOawEdlVO
cyHrwDCPEqf5ZzD4NhfTriEOw+j0pEH/onitUGvoQRtx15xWyaJQxDEBMTYMLewE
86D1bltwnTNczE3UZb7oQLJXkAX5hcLtou70XJGgZITvJkK+kp/xot2eFjnqRz/u
WeXnKhYAmC07EKwZ1uw047eHKwMMRBYqzApLwoQtfE430Kxf2q8de64x8zDbi6YM
1J4r8OAxOtHVyfJ0j7Q23DEZz0VVb4b1Tx5OG2Re/KSNvqI0awJO4TcRmOR88OyY
dzyXgnX6Sa7GVQY1FXvn7vtFuDAt7egZOzeomSHL9bdX07LTQ4UtM88EV9wm3q4q
smoatV9jsvPQ1zxCU3aQD/5eWTJH2/kz1LIuBL/Qi5XQpJn91lBtUWJrCgkHWPGu
f//rnnXmsG7DAChW+yZ7cFO8lfNa8sFhPqSxCYphWmJTrvadyQtDngB8JakWdnmK
pfGS6y5lel+181vw38ZZKt04AKM+nDY8051lBM7Q9Q6kTLI33UZeImndx5xYukVD
kV6aQ31HYfEark15c7iEz+OAcwFnM2ntXMt7kKGd40CqzusiPcQkPqPbAQARAQAB
tDhGcmVlQlNEIENvcmUgVGVhbSBTZWNyZXRhcnkgPGNvcmUtc2VjcmV0YXJ5QEZy
ZWVCU0Qub3JnPokCVwQTAQoAQQIbAwgLCQ0IDAcLAwUVCgkICwUWAwIBAAIeBQIX
gBYhBBojapJSjQDdeWV2/k1jJRjDVGsFBQJnp8PiBQkHeofTAAoJEE1jJRjDVGsF
GMIQALhj+mNpH8OmTFeihQ6t9P8un3llz6Wmqe/Q+ULWeqJvV/uC1J5T9fnoGhwF
MgECuguXYJtoYQ16KXnsSOs1tcqMOK9GtEtFJTGe2DtfLiBednwUvu9j3HmTlwLN
M+7rqkiC0HCg2qSjcMjxbVbA5BSgNkgfyTSO2YdjfaZ+ceiHwo/qa5xWE6i2dMR1
PLGMMHTeELdtdSH6VR9/3h0qt3qzwdMBRCVAQHbim7CqwjUH9jg+IOySXl81jNoB
xuVZR3pKshyY4Oxo1dK+W86Uiff/+c4jCAxokGWIR7C4MkZZWUQqV7920gkZFC/5
qzpT1A1/sFUg8HfFTr0vCoPSVFWn2+Tto3vr78DICVaAf02aYAFlyKK18BMCoHkS
hDDO+/JQZmvHOIgEYK+T2WN1c9gm9IDJzGZuH0X2C/vkREnJKkccJfB4pXuN10wt
fqyP9fn8h6+/t+sv3qNLm4d8fkmLXofulG3WB4i/F+Hip2rjPvvBCF7zl4xy6cJ2
xVY5HUOBTqmIwVhYwUpXaqNoWa/qJBLTuot3z7ciKmKX6Lq+Dze5dzhrPNl/CalC
HBf3miHRK3TZbYLooG3bcEWgxU2BnBi3vl5NpCoUOKkPYRlALXi2TyHmPxO7oT4e
mIzS6BgnXOqO+AXvbKKfayuSePdBqkNMK/SMC4Dylkf6Xj25iQIzBBABCgAdFiEE
EBpxaxYrAOVb7eoFrbv4YQo3ibcFAmbu7x4ACgkQrbv4YQo3ibcr5A//TIcbD/EW
YJz0zrUQdc9xG3UNfU6uHmQzAuUy5ginevyqv0TSso5qvSkOHvdxbi41rfMiB2RJ
V3q0n0PSvHFld89fOTZUMZXaPvozCiBYWScrt+KA/2pq3K8mUumHS+IFtpHLL2Tu
+gI8XCHUFxO9HUThM/rFlIyEdzoctgmqQ7IG4uZG+o2J3w09llhDAUe0vraJK1On
p9yFACnRrhqvl41JeUWcv9MH9JcwHUqtUo9WLTCIb+hkByTOyRfHBYpYw//bdXdf
6rkCwKVWpyMDbk61zq2VsS1kqbP9IH/A8CsBnA6mg+zPq2i7HIFw8Swj4OGJcIvG
a9ubUYJRjDhX/vBpNrtncANZ88FQmA+Maq0vu0LS5IIGyIKkvd1fKIsyvBDDa9kE
nfCW0XMkJA0Gf+kxdB+elXQHBwKO2sr5BiKnJT5lJq3Yu9fxxGBnf93yiN4E9bmF
gG7cZxpyb1Bp76TJhLcANyybOTjCtiNRrgqeaSx9/6hSPfPigGXIne0H2lmJO6oq
jUrsYmFiWvU9sc7AcPVw/eHG8FgW35TuwKX71z8w994iaahUPNcSVyXOUD+QNROv
HhGUXrc81t613rivh22N0NZpNubVatq43KV7+/bnPyWBIi1Awh3vIFsNNSQSrYxF
lUuQaAHQXTeZMZ/7npE4t86seMt0T7BGn765Ag0EZdDWjwEQAL3VwFifpnRCYzQ4
VZ1dAjp8w542iRprqeA+C3tvNbk2OvKpN3DIc49l2bgNZ/VI7/T58lEKrFsgLK4w
AWtv18OV7xuh0AuOblmq5MvcPrUelkPj5HA7M6Ng/rAubuHfwdP/IjIrzzY6+XDh
fP9N5KIv9VRJYT23BbvLPeit4J4tQEB/NpxL3L6zI75qq4R3T/EkHP6Y9Buup9Lr
isJlcxkSK+CyORCgTpEz6fWsXiTDgS7cTaQ969XCygBpj4wRQzwkBdokxo1wsiFt
4zLt07/PrPYjeHltTDkrF9XDNhLNJ5GliHnd01oHg1j5/n+9Osvh1maoFwuBxXTN
nTZ2P+7RKLBaVQVSkUa5KJbXoM3v7bMbXM5aLs2XjglrAzrZOV+Y7zOu5UYQdpXd
7x8XAEtEozRJzt7dosQIhKx9h4LlltFFulDUpf5VCvcUEoAzMbiX0aju9n5Rwsql
DHatU9Mm3W8OdFXj1foIXZD+VX2Jp9DgxPLoAJ0CWhPXJ8f+WFSJZCjWoPJEJKWY
```

```
0EOxiXyAuvAyniAZAC/eKZkfGckXmu7edRgYbRTTwPmZ/axa/k9aHsHLwmbxuZo/
xxQXzqU70EleHZ31A3mOqsC1epjN6dn4AFKgvVesP/K/fbvSsfsfiABS2A/68ne4
zJG73Tnk4L6J79vrXc2iMJbmdg3ZABEBAAGJAjwEGAEKACYCGwwWIQQaI2qSUo0A
3Xlldv5NYyUYw1RrBQUCZ6fEFQUJB3qIBgAKCRBNYyUYw1RrBQd9EACMVckxzy4w
aUGlERguJ+kslS8MkMjNqnfDPLRDVxbUxNvbbhw7/u9blM65DCGCeNLc2n4oiu5C
E3I095AKmvq/0dOa81mEEdZkC1CVc64bXWbEyz5AtSHUpgdxRso6C+YopndSiz1T
WcIagQRXfWaw3FBWPooA87gmibmSmCegCtqx+uyc5QxX2eFI8mRK7vlfnGpYKHs0
D1/yUSGQOwoNRJ5FYm+ynfDE3FzHEQL7lv1vpv1k3xNKfBziMMg4IMEBKNHV4VKN
qJpa8UCodeTGSWQdNnCeCWPsxz5oQjCcVH9Z3e8sMpWLHhRcBZzSwXUOws2GbRMH
xHwrfrpHJcrBhe64pgfGOvZLUJ9BDs+8egTHsqRFacipbtTR+hhVhuJEHDaQQWuM
8IRHj9HIuTAczET8JTDHTMIoo8DZdOtiW/YgqCDwYghkI77d12oNQqYoeJ2HiqbK
cGzwCpsR0A+p/iOAxJG13tsxqZV8TQ8iTokWG6ACtZ7sfeWEhxqMbUKUMgogZn0I
3n1kV+tUZC6BQrIYI7TiKg95wLZsIydeoIsQoNZwvyKAXfVmQ62YjIX8njZwN+07
8/ipUPJxCYa8zL8BZyDmoFJqa3y9z+11+vtiZ9t+aTwGvjpHDwyeCJco7go9cU3m
GRFZYciqIoG4n3tl8Pob15vFlVqk47rRqg==
=8TzT
-----END PGP PUBLIC KEY BLOCK-----
```

D.1.3. Ports Management Team Titkár <portmgr-secretary@FreeBSD.org>

```
pub   ed25519/E3C401F60D709D59 2023-03-06 [SC] [expires: 2027-03-05]
      Key fingerprint = BED4 A1D3 6555 B681 2E9F  ABDA E3C4 01F6 0D70 9D59
uid                           FreeBSD Ports Management Team Secretary <portmgr-
secretary@FreeBSD.org>
sub   cv25519/2C92B55E27A641C3 2023-03-06 [E] [expires: 2027-03-05]
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mDMEZAXJvxYJKwYBBAHaRw8BAQdASFAC20WL3R1T6uNyGMZbfJCxDkcP4C5vi3Op
tcZ2fbq0R0ZyZWVCU0QgUG9ydHMgTWFuYWdlbWVudCBUZWFtIFNlY3JldGFyeSA8
cG9ydG1nci1zZWNyZXRhcnlARnJlZUJTRC5vcmc+iJYEExYKAD4WIQS+1KHTZVW2
gS6fq9rjxAH2DXCdWQUCZAXJvwIbAwUJB4TOAAULCQgHAwUVCgkICwUWAwIBAAIe
BQIXgAAKCRDjxAH2DXCdWYN1AP43TjyfZtZ3DLYT++g0+SuPsoO/3yWVybA+UmFL
zb8MngEA+LLNUfvEwCuXS/soh+ww5bpfmi3UUmeGiQEAXug3iA+JATMEEAEKAB0W
IQT7N0XIbxXo7ayBMvzYKU7Du8TX1QUCZAXLkwAKCRDYKU7Du8TX1XHMB/9R1MX4
6zMgpKqPPt76GOI+eGEdBK6bY8aJZjQGdqTh9f6VtXVoTGIG7cvhc9X8tDBoB0PT
2KZWheF51AV1+NHU4HwLAQ1BMebrFvWSfkw4xg4fBGwDhz9/GN85No+Js772V5ey
8lRiL6meRVWxMlLyWcxGd8JjcC5yX/iAUQ3SBGCLqW7unWjjg7CTd+AMBwcqPGrv
ax8q6eFVguJcHJAjMnKf6HAy4cpK3s+uMoUBCGnszSN12B3ysKfyC4pNO/pix5tA
Q5v8aRqTeFPh5zmNhWo0KGPzplTPqRQSHDl7GDQC8Ru3MhzFkeWzHsexjZVwS6W2
DPcYpuuAsA0XOZIZiQIzBBABCgAdFiEEEBpxaxYrAOVb7eoFrbv4YQo3ibcFAmQF
0u0ACgkQrbv4YQo3ibccwg/9F2Xuic3nhKxRbB3mJeDo6SYQETa/Gh1qQ34+8zlt
8UMazOx67gnYQfy+pXjro6eQ2up0a4eUYezcNOudqAQD21nRz3HA6EQVNcE/TzEA
xl5CJntTaLOt7S+EDXFW5BuQIvhhoMGgm8+WNVgA0EJ7tfL0OcYBSvr19fqwChEn
9c14cSk6mgHSsleP5NvskYN053pxHwy0LTSb8YBBv52th37t/CRFC1363rS5q+D7
JixFopd1O5pKpA5ipvE4gGgRjPtwjx0SjjepwK/3fuhEJQQyKzTIKlMfu2Dj/iR2
Li1Sfccau5LQXOj9fUITU3u1YG7yrm8VGzT7ao4d+KRwgMLjd2pLqiGIbbJwGBiP
FRmtilWQoeIlmSlFX4obAA517DOK0pW1mH8+eEn4EJd3SekT3yzFyKTASv0J48Z8
```

```
3F928xg+eZvHxVC0t1J+J5IG0gt3EEncuWKIPQGR7PiQbti6R3FQVTz6WfMWOebP
Qi0E9F/Aqakr6Vj2sKGrDq+ebpaF5G8Yw1YrUl2IDiPzkCegp3ZbI0wh11Xvzhi8
LXPQGK4jBQas4G8cegfitzmtdGRHYrbMv0R9I4mvaL+WlOuD2AvyVG28lguqVhnN
AZP+ohdquYyX2CNCVvbKWAtXo6Ur0vWG8BL8m6defAtEkIwVBALaOHQOSI3aNUz4
lwy4OARkBcm/EgorBgEEAZdVAQUBAQdAsefmSfxEOdOr02+K/6noYCuJ1FeAWVz6
jFYQ+9w6jggDAQgHiH4EGBYKACYWIQS+1KHTZVW2gS6fq9rjxAH2DXCdWQUCZAXJ
vwIbDAUJB4TOAAAKCRDjxAH2DXCdWRl4AP9h5ot212BK29S6ZcMBhHvmtF5PG1oD
c7LnZycSRmbFiwEAndCMpAGOhDW8iVgDd0wLQq/ZMPe+xccfG1b3zFH2EgE=
=iiAT
-----END PGP PUBLIC KEY BLOCK-----
```

D.1.4. <doceng-secretary@FreeBSD.org>

```
pub   rsa2048/E1C03580AEB45E58 2019-10-31 [SC] [expires: 2022-10-30]
      Key fingerprint = F24D 7B32 B864 625E 5541  A0E4 E1C0 3580 AEB4 5E58
uid                           FreeBSD Doceng Team Secretary <doceng-
secretary@freebsd.org>
sub   rsa2048/9EA8D713509472FC 2019-10-31 [E] [expires: 2022-10-30]
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQENBF27FFcBCADeoSsIgyQUY8vREwkTikwFFlNg31MVy5s/Nq1cNK1PRfRMnprS
yfB62KqbYuz16bmQKaA9zHN4FGfiTvR6tl66LVHm1s/5HPiLv8sP14GsruLro9zN
v72dO7a9i68bMw+jarPOnu9dGiDFEI0dACOkdCGEYKEUapQeNpmWRrQ46BeXyFwF
JcNx76bJJUkwk6fWC0W63D762e6lCEX6ndoaPjjLBnFvtx13heNGUc8RukBwe2mA
U5pSGHj47J05bdWiRSwZaXa8PcW+20zTWaP755w7zWe4h60GANY7OsT9nuOqsioJ
QonxTrJuZweKRV8fNQ1EfDws3HZr7/7iXvO3ABEBAAG0PEZyZWVCU0QgRG9jZW5n
IFRlYW0gU2VjcmV0YXJ5IDxkb2Nlbmctc2VjcmV0YXJ5QGZyZWVic2Qub3JnPokB
VAQTAQoAPhYhBPJNezK4ZGJeVUGg5OHANYCutF5YBQJduxRXAhsDBQkFo5qABQsJ
CAcDBRUKCQgLBRYDAgEAAh4BAheAAAoJEOHANYCutF5YB2IIALw+EPYmOz9qlqIn
oTFmk/5MrcdzC5iLEfxubbF6TopDWsWPiOh5mAuvfEmROSGf6ctvdYe9UtQV3VNY
KeeyskeFrIBOFo2KG/dFqKPAWef6IfhbW3HWDWo5uOBg01jHzQ/pB1n6SMKiXfsM
idL9wN+UQKxF3Y7S/bVrZTV0isRUolO9+8kQeSYT/NMojVM0H2fWrTP/TaNEW4fY
JBDAl5hsktzdl8sdbNqdC0GiX3xb4GvgVzGGQELagsxjfuXk6PfOyn6Wx2d+yRcI
FrKojmhihBp5VGFQkntBIXQkaW0xhW+WBGxwXdaAl0drQlZ3W+edgdOl705x73kf
Uw3Fh2a5AQ0EXbsUVwEIANEPAsltM4vFj2pi5xEuHEcZIrIX/ZJhoaBtZkqvkB+H
4pu3/eQHK5hg0Dw12ugffPMz8mi57iGNI9TXd8ZYMJxAdvEZSDHCKZTX9G+FcxWa
/AzKNiG25uSISzz7rMB/lV1gofCdGtpHFRFTiNxFcoacugTdlYDiscgJZMJSg/hC
GXBdEKXR5WRAgAGandcL8llCToOt1lZEOkd5vJM861w6evgDhAZ2HGhRuG8/NDxG
r4UtlnYGUCFof/Q4oPNbDJzmZXF+8OQyTNcEpVD3leEOWG1Uv5XWS2XKVHcHZZ++
ISo/B5Q6Oi3SJFCVV9f+g09YF+PgfP/mVMBgif2fT20AEQEAAYkBPAQYAQoAJhYh
BPJNezK4ZGJeVUGg5OHANYCutF5YBQJduxRXAhsMBQkFo5qAAAoJEOHANYCutF5Y
kecIAMTh2VHQqjXHTszQMsy3NjiTVVITI3z+pzY0u2EYmLytXQ2pZMzLHMcklmub
5po0X4EvL6bZiJcLMI2mSrOs0Gp8P3hyMI40IkqoLMp7VA2LFlPgIJ7K5W4oVwf8
khY6lw7qg2l69APm/MM3xAyiL4p6MU8tpvWg5AncZ6lxyy27rxVflzEtCrKQuG/a
oVaOlMjH3uxvOK6IIxlhvWD0nKs/e2h2HIAZ+ILE6ytS5ZEg2GXuigoQZdEnv71L
xyvE9JANwGZLkDxnS5pgN2ikfkQYlFpJEkrNTQleCOHIIIp8vgJngEaP51xOIbQM
CiG/y3cmKQ/ZfH7BBvlZVtZKQsI=
```

```
=MQKT
-----END PGP PUBLIC KEY BLOCK-----
```