FreeBSD Porter’s Handbook

Índice

1. Introdução . . . 9
2. Criando um Novo Port . . . 10
3. Port Rápido . . . 11
   3.1. Escrevendo o Makefile . . . 11
   3.2. Escrevendo os Arquivos de Descrição . . . 12
   3.3. Criando o Arquivo Checksum . . . 14
   3.4. Testando o Port . . . 14
   3.5. Verificando o Port com `portlint` . . . 15
   3.6. Enviando o Novo Port . . . 15
4. Port Lento . . . 17
   4.1. Como as Coisas Funcionam . . . 17
   4.2. Obtendo os Fontes Originais . . . 18
   4.3. Modificando o Port . . . 19
   4.4. Patching . . . 19
   4.5. Configurando . . . 23
   4.6. Manipulando a Entrada do Usuário . . . 23
5. Configurando o Makefile . . . 24
   5.1. O Código Fonte Original . . . 24
   5.2. Categorização . . . 34
   5.3. Os Arquivos de Distribuição . . . 41
   5.4. `MAINTAINER` . . . 66
   5.5. `COMMENT` . . . 67
   5.6. Licenças . . . 68
   5.7. `PORTSCOUT` . . . 79
   5.8. Dependências . . . 79
   5.9. O `USE_*` . . . 83
   5.10. Ports Slaves e `MASTERDIR` . . . 86
   5.11. Páginas de Manual . . . 87
   5.12. Arquivos de Informação . . . 87
   5.13. Opções do Makefile . . . 87
   5.14. Especificando o Diretório de Trabalho . . . 107
   5.15. Manipulando Conflitos . . . 108
   5.16. Instalando Arquivos . . . 110
   5.17. Use `BINARY_ALIAS` para Renomear Comandos Em Vez de Aplicar Patch na Compilação . . . 114
6. Considerações Especiais . . . 116
   6.1. Staging . . . 116
   6.2. Bibliotecas Empacotadas (Bundled) . . . 117
   6.3. Bibliotecas Compartilhadas . . . 119

6.4. Ports com Restrições de Distribuição ou Preocupações Legais . . . 119
6.5. Mecanismos de Compilação . . . 121
6.6. Usando o GNU Autotools . . . 135
6.7. Usando o GNU `gettext` . . . 135
6.8. Usando Perl . . . 137
6.9. Usando o X11 . . . 139
6.10. Usando o GNOME . . . 142
6.11. Componentes GNOME . . . 144
6.12. Usando o Qt . . . 149
6.13. Usando o KDE . . . 155
6.14. Usando o LXQt . . . 163
6.15. Usando Java . . . 163
6.16. Aplicações Web, Apache e PHP . . . 167
6.17. Usando Python . . . 170
6.18. Usando Tcl/Tk . . . 173
6.19. Usando Ruby . . . 174
6.20. Usando SDL . . . 174
6.21. Usando wxWidgets . . . 175
6.22. Usando Lua . . . 180
6.23. Usando `iconv` . . . 184
6.24. Usando o Xfce . . . 185
6.25. Usando Bancos de Dados . . . 187
6.26. Iniciando e Parando Serviços (com scripts `rc`) . . . 188
6.27. Adicionando Usuários e Grupos . . . 191
6.28. Ports que Dependem dos Fontes do kernel . . . 191
6.29. Bibliotecas Go . . . 192
6.30. Bibliotecas Haskell . . . 192
6.31. Arquivos Shell Completion . . . 192
7. Flavors . . . 193
7.1. Uma Introdução aos Flavors . . . 193
7.2. Usando FLAVORS . . . 193
7.3. `USES=php` e Flavors . . . 196
7.4. `USES=python` e Flavors . . . 197
7.5. `USES=lua` e Flavors . . . 198
8. Prácticas Avançadas de pkg-plist . . . 199
8.1. Alterando o pkg-plist Baseado em Variáveis Make . . . 199
8.2. Diretórios Vazios . . . 200
8.3. Arquivos de Configuração . . . 201
8.4. Lista de Pacotes Estática versus Dinâmica . . . 202
8.5. Criação Automatizada da Lista de Pacotes . . . 202
8.6. Expandindo a Lista de Pacotes com Keywords . . . 204

9. pkg-* . . . 212
9.1. pkg-message . . . 212
9.2. pkg-install . . . 215
9.3. pkg-deinstall . . . 215
9.4. Mudando os nomes dos pkg-* . . . 215
9.5. Fazendo uso de `SUB_FILES` e `SUB_LIST` . . . 216
10. Testando o Port . . . 217
10.1. Executando `make describe` . . . 217
10.2. Portlint . . . 217
10.3. Ferramentas do Ports . . . 217
10.4. `PREFIX` e `DESTDIR` . . . 218
10.5. Poudriere . . . 219
11. Atualizando um Port . . . 229
11.1. Usando o Subversion para Criar Patches . . . 230
11.2. UPDATE e MOVED . . . 232
12. Segurança . . . 234
12.1. Por Que Segurança é Tão Importante . . . 234
12.2. Corrigindo Vulnerabilidades de Segurança . . . 234
12.3. Mantendo a Comunidade Informada . . . 235
13. O Que Fazer e Não Fazer . . . 240
13.1. Introdução . . . 240
13.2. `WRKDIR` . . . 240
13.3. `WRKDIRPREFIX` . . . 240
13.4. Diferenciando Sistemas Operacionais e Versões de OS . . . 240
13.5. Escrevendo Algo Depois do bsd.port.mk . . . 241
13.6. Uso de Declarações `exec` em Wrapper Scripts . . . 242
13.7. Faça as Coisas Racionalmente . . . 242
13.8. Respeite Ambos `CC` e `CXX` . . . 242
13.9. Respeite `CFLAGS` . . . 243
13.10. Logs de Compilação Detalhados . . . 243
13.11. Feedback . . . 244
13.12. README.html . . . 244
13.13. Marcando um Port não Instalável com a variável `BROKEN`, `FORBIDDEN` ou `IGNORE` . . . 244
13.14. Considerações Arquitetônicas . . . 246
13.15. Marcando um Port para Remoção com `DEPRECATED` ou `EXPIRATION_DATE` . . . 247
13.16. Evite o Uso do Construtor `.error` . . . 247
13.17. Uso de sysctl . . . 248
13.18. Atualizando Distfiles . . . 248
13.19. Uso de Padrões POSIX . . . 248
13.20. Miscelânea . . . 249
14. Um Exemplo de Makefile . . . 250

15. Ordem das Variáveis nos Makefiles de Port . . . 252
15.1. Bloco PORTNAME . . . 252
15.2. Bloco PATCHFILES . . . 253
15.3. Bloco MAINTAINER . . . 253
15.4. Bloco LICENSE . . . 253
15.5. Mensagens Genéricas BROKEN/IGNORE/DEPRECATED . . . 254
15.6. O Bloco de Dependências . . . 254
15.7. Flavors . . . 255
15.8. USES e USE_x . . . 255
15.9. Variáveis Padrão bsd.port.mk . . . 255
15.10. Opções e Assistentes . . . 255
15.11. O Restante das Variáveis . . . 256
15.12. Os Targets . . . 256
16. Mantendo-se Atualizado . . . 258
16.1. FreshPorts . . . 258
16.2. A interface Web para o Repositório do Código Fonte . . . 258
16.3. A Lista de Discussão de Ports do FreeBSD . . . 258
16.4. O Cluster de Compilação de Ports do FreeBSD . . . 259
16.5. Portscout: o Scanner de Distfile de Ports do FreeBSD . . . 259
16.6. O Sistema de Monitoramento de Ports do FreeBSD . . . 259
17. Usando Macros USES . . . 261
17.1. Uma introdução ao USES . . . 261
17.2. 7z . . . 261
17.3. ada . . . 262
17.4. autoreconf . . . 262
17.5. blaslapack . . . 262
17.6. bdb . . . 262
17.7. bison . . . 263
17.8. cabal . . . 263
17.9. cargo . . . 264
17.10. charsetfix . . . 264
17.11. cmake . . . 264
17.12. compiler . . . 265
17.13. cpe . . . 265
17.14. cran . . . 266
17.15. desktop-file-utils . . . 266
17.16. desthack . . . 266
17.17. display . . . 266
17.18. dos2unix . . . 267
17.19. drupal . . . 267
17.20. fakeroot . . . 267

17.21. fam . . . 267
17.22. firebird . . . 267
17.23. fonts . . . 267
17.24. fortran . . . 268
17.25. fuse . . . 268
17.26. gem . . . 268
17.27. gettext . . . 268
17.28. gettext-runtime . . . 268
17.29. gettext-tools . . . 268
17.30. ghostscript . . . 269
17.31. gl . . . 269
17.32. gmake . . . 269
17.33. gnome . . . 270
17.34. go . . . 272
17.35. gperf . . . 273
17.36. grantlee . . . 273
17.37. groff . . . 274
17.38. gssapi . . . 274
17.39. horde . . . 275
17.40. iconv . . . 275
17.41. imake . . . 275
17.42. kde . . . 275
17.43. kmod . . . 275
17.44. lha . . . 276
17.45. libarchive . . . 276
17.46. libedit . . . 276
17.47. libtool . . . 276
17.48. linux . . . 276
17.49. localbase . . . 279
17.50. lua . . . 279
17.51. lxqt . . . 279
17.52. makeinfo . . . 279
17.53. makeself . . . 280
17.54. mate . . . 280
17.55. meson . . . 281
17.56. metaport . . . 281
17.57. mysql . . . 281
17.58. mono . . . 281
17.59. motif . . . 281
17.60. ncurses . . . 282
17.61. ninja . . . 282

17.62. objc . . . 282
17.63. openal . . . 282
17.64. pathfix . . . 282
17.65. pear . . . 282
17.66. perl5 . . . 283
17.67. pgsql . . . 283
17.68. php . . . 283
17.69. pkgconfig . . . 285
17.70. pure . . . 286
17.71. pyqt . . . 286
17.72. python . . . 287
17.73. qmail . . . 287
17.74. qmake . . . 288
17.75. qt . . . 288
17.76. qt-dist . . . 288
17.77. readline . . . 289
17.78. samba . . . 289
17.79. scons . . . 289
17.80. shared-mime-info . . . 289
17.81. shebangfix . . . 289
17.82. sqlite . . . 292
17.83. ssl . . . 292
17.84. tar . . . 293
17.85. tcl . . . 293
17.86. terminfo . . . 293
17.87. tk . . . 293
17.88. uidfix . . . 293
17.89. uniquefiles . . . 294
17.90. varnish . . . 294
17.91. webplugin . . . 294
17.92. xfce . . . 295
17.93. xorg . . . 295
17.94. xorg-cat . . . 297
17.95. zip . . . 297
18. Valores __FreeBSD_version . . . 298
18.1. Versões do FreeBSD 13 . . . 298
18.2. Versões do FreeBSD 12 . . . 312
18.3. Versões do FreeBSD 11 . . . 327
18.4. Versões do FreeBSD 10 . . . 347
18.5. Versões do FreeBSD 9 . . . 361
18.6. Versões do FreeBSD 8 . . . 370

18.7. Versões do FreeBSD 7 386
18.8. Versões do FreeBSD 6 395
18.9. Versões do FreeBSD 5 402
18.10. Versões do FreeBSD 4 414
18.11. Versões do FreeBSD 3 418
18.12. Versões do FreeBSD 2.2 420
18.13. FreeBSD 2 Antes das Versões 2.2-RELEASE 420

Capítulo 1. Introdução

A Coleção de Ports do FreeBSD é a maneira como quase todo mundo instala aplicativos ("ports") no FreeBSD. Como tudo no FreeBSD, é principalmente um esforço voluntário. É importante ter isso em mente ao ler este documento.

No FreeBSD, qualquer um pode enviar um novo port ou ser voluntário para manter um port que esteja sem mantenedor. Nenhum privilégio de commit é necessário.

Capítulo 2. Criando um Novo Port

Interessado em fazer um novo port ou atualizar os ports existentes? Ótimo!

O que segue são algumas instruções para criar um novo port para o FreeBSD. Para atualizar um port existente, leia este documento e depois leia o Atualizando um Port.

Quando este documento não for suficientemente detalhado, consulte /usr/ports/Mk/bsd.port.mk, que é incluído por todos os Makefiles dos ports. Mesmo aqueles que não estão hackeando os Makefiles diariamente podem ganhar muito conhecimento com isso. Além disso, perguntas específicas podem ser enviadas à Lista de discussão do ports do FreeBSD.

> Apenas uma fração das variáveis (`VAR`) que podem ser sobrepostas são mencionados neste documento. A maioria (se não todas) estão documentadas no início do /usr/ports/Mk/bsd.port.mk; as outras provavelmente deveriam estar também. Observe que esse arquivo usa uma configuração de tabulação não padrão: O Emacs e o Vim irão reconhecer a configuração ao carregar o arquivo. Ambos vi(1) e ex(1) podem ser configurados para usar o valor correto digitando `:set tabstop=4` uma vez que o arquivo foi carregado.

Procurando algo fácil para começar? Dê uma olhada na lista de ports desejados e veja se você pode trabalhar em um (ou mais de um).

Capítulo 3. Port Rápido

Esta seção descreve como criar rapidamente um novo port. Para aplicativos em que esse método rápido não for adequado, o processo "Slow Porting" está descrito no Port Lento.

Primeiro, obtenha o tarball original e coloque-o em DISTDIR, que por padrão é o diretório /usr/ports/distfiles.

Estas etapas assumem que o software foi compilado de forma simples (out-of-the-box). Em outras palavras, não foi necessária absolutamente nenhuma mudança para o aplicativo funcionar em um sistema FreeBSD. Se alguma coisa teve que ser alterada, por favor consulte o Port Lento.

Recomenda-se definir a variável DEVELOPER do make(1) em /etc/make.conf antes de começar o trabalho com os ports.

```
# echo DEVELOPER=yes >> /etc/make.conf
```

Esta configuração habilita o "modo de desenvolvedor" que exibe avisos sobre a descontinuidade de comandos e ativa algumas verificações de qualidade adicionais nas execuções do comando make.

3.1. Escrevendo o Makefile

O Makefile mínimo seria algo assim:

```
# $FreeBSD: head/pt_BR.ISO8859-1/books/porters-handbook/book.xml 54410 2020-08-05
22:13:01Z dbaio $

PORTNAME=   oneko
DISTVERSION=    1.1b
CATEGORIES= games
MASTER_SITES=   ftp://ftp.cs.columbia.edu/archives/X11R5/contrib/

MAINTAINER= youremail@example.com
COMMENT=    Cat chasing a mouse all over the screen

.include <bsd.port.mk>
```

Em alguns casos, o Makefile de um port existente pode conter linhas adicionais no cabeçalho, como o nome do port e a data em que foi criado. Esta informação adicional foi declarada obsoleta e está sendo eliminada.

Tente entender o exemplo. Não se preocupe com o conteúdo da linha $FreeBSD: head/pt_BR.ISO8859-1/books/porters-handbook/book.xml 54410 2020-08-05 22:13:01Z dbaio $, ela

será preenchida automaticamente pelo Subversion quando o port for importado para nossa árvore de ports principais. Um exemplo mais detalhado é mostrado na seção exemplo de Makefile.

3.2. Escrevendo os Arquivos de Descrição

Existem dois arquivos de descrição que são necessários para qualquer port, independente deles estarem empacotados ou não. Eles são o pkg-descr e o pkg-plist. Seus prefixos pkg- distingue-os de outros arquivos.

3.2.1. pkg-descr

Esta é uma descrição mais longa do port. Um ou alguns parágrafos que explicam o que o port faz é suficiente.

> Isto *não é* um manual ou uma descrição detalhada sobre como usar ou compilar o port! *Por favor, tenha cuidado ao copiar do README ou manpage*. Muitas vezes, eles não são uma descrição concisa do port ou estão em um formato estranho. Por exemplo, as páginas de manual têm espaçamento justificado, o que parece particularmente ruim com fontes monoespaçadas.
>
> Por outro lado, o conteúdo de pkg-descr deve ser mais longo que a linha `COMMENT` do Makefile. Ele deve explicar com mais profundidade o que é o port.

Um pkg-descr bem escrito descreve o port completamente o suficiente para que os usuários não precisem consultar a documentação ou visitar o site para entender o que o software faz, como ele pode ser útil ou quais recursos particularmente legais ele possui. A menção de certos requisitos, como um kit de ferramentas gráfico, dependências pesadas, ambiente de runtime ou linguagens de implementação, ajuda os usuários a decidir se este port funcionará para eles.

Inclua uma URL para a página Web oficial. Prefixe *um* dos sites (escolha o mais comum) com `WWW:` (seguido por um único espaço) para que as ferramentas automatizadas funcionem corretamente. Se a URI é a raiz do site ou diretório, ele deve ser terminado com uma barra.

> Se a página web listada para um port não estiver disponível, tente pesquisar na Internet primeiro para ver se o site oficial foi movido, foi renomeado ou se está hospedado em outro lugar.

Este exemplo mostra como parece o pkg-descr:

```
This is a port of oneko, in which a cat chases a poor mouse all over
the screen.
 :
(etc.)

WWW: http://www.oneko.org/
```

3.2.2. pkg-plist

Este arquivo lista todos os arquivos instalados pelo port. Ele também é chamado de "packing list" (lista de empacotamento) porque o pacote é gerado empacotando os arquivos listados aqui. Os pathnames são relativos ao prefixo de instalação (geralmente /usr/local).

Aqui está um pequeno exemplo:

```
bin/oneko
man/man1/oneko.1.gz
lib/X11/app-defaults/Oneko
lib/X11/oneko/cat1.xpm
lib/X11/oneko/cat2.xpm
lib/X11/oneko/mouse.xpm
```

Consulte a manpage do pkg-create(8) para detalhes sobre a lista de empacotamento.

É recomendado manter todos os nomes de arquivos neste arquivo classificados em ordem alfabética. Isso tornará muito mais fácil verificar as alterações ao atualizar o port.

Criar uma lista de packing manualmente pode ser uma tarefa muito tediosa. Se o port instalar um grande número de arquivos, criar a lista de empacotamento automaticamente pode economizar tempo.

Há apenas um caso em que o pkg-plist pode ser omitido de um port. Se o port instalar apenas alguns arquivos, liste-os em `PLIST_FILES`, dentro do Makefile do port. Por exemplo, poderíamos passar sem o pkg-plist no port oneko acima, adicionando estas linhas para no Makefile:

```
PLIST_FILES=    bin/oneko \
        man/man1/oneko.1.gz \
        lib/X11/app-defaults/Oneko \
        lib/X11/oneko/cat1.xpm \
        lib/X11/oneko/cat2.xpm \
        lib/X11/oneko/mouse.xpm
```

Uso de `PLIST_FILES` não deve ser abusado. Ao procurar pela origem de um arquivo, as pessoas geralmente tentam usar o grep através do pkg-plist nos arquivos na árvore de ports. Listar os arquivos na variável `PLIST_FILES` dentro do Makefile torna esta busca mais difícil.

Se um port precisar criar um diretório vazio, ou criar diretórios fora do ${PREFIX} durante a instalação, consulte Limpando Diretórios Vazios para maiores informações.

Como `PLIST_FILES` é uma variavel do make(1), qualquer entrada com espaços deve

ser envolvida por aspas. Por exemplo, se estiver usando palavras-chave descritas em pkg-create(8) e na Expandindo a Lista de Pacotes com Keywords, a entrada deve ser citada.

```
PLIST_FILES=    "@sample ${ETCDIR}/oneko.conf.sample"
```

Mais tarde vamos ver como o pkg-plist e a `PLIST_FILES` podem ser utilizados para executar tarefas mais sofisticadas.

3.3. Criando o Arquivo Checksum

Apenas digite `make makesum`. O framework do ports irá gerar automaticamente o distinfo. Não tente gerar o arquivo manualmente.

3.4. Testando o Port

Certifique-se de que as regras do port façam exatamente o que é desejado, incluindo o empacotamento do port. Estes são os pontos importantes a serem verificados:

- pkg-plist não contém nada não instalado pelo port.
- pkg-plist contém tudo o que é instalado pelo port.
- O port pode ser instalado usando o target `install`. Isso verifica se o script de instalação está funcionando corretamente.
- O port pode ser desinstalado adequadamente usando o target `deinstall`. Isso verifica se o script de desinstalação funciona corretamente.
- O port só tem acesso aos recursos de rede durante a fase target `fetch`. Isto é importante para os construtores de pacotes, tais como o ports-mgmt/poudriere.
- Certifique-se de que o comando `make package` pode ser executado como um usuário normal (ou seja, não como `root`). Se isso falhar, talvez seja necessário corrigir o software. Veja a `fakeroot` e também a `uidfix`.

Procedure: Ordem Recomendada de Teste

1. `make stage`
2. `make stage-qa`
3. `make package`
4. `make install`
5. `make deinstall`
6. `make package` (como usuário)

Certifique-se de que nenhum aviso é exibido em nenhum dos estágios.

Testes automatizados completos podem ser feitos com o ports-mgmt/poudriere da coleção do Ports, veja a Poudriere para maiores informações. Ele mantém `jails` onde todas as etapas mostradas acima podem ser testadas sem afetar o estado do sistema host.

3.5. Verificando o Port com `portlint`

Por favor, use o `portlint` para ver se o port está de acordo com as nossas diretrizes. O programa ports-mgmt/portlint faz parte da coleção de ports. Em particular, ele verifica se o Makefile está correto e se o pacote está nomeado apropriadamente.

> Não siga cegamente a saída do `portlint`. Ela é uma ferramenta de lint estática e às vezes comete erros.

3.6. Enviando o Novo Port

Antes de enviar o novo port, leia a seção sobre o que fazer e o que não fazer.

Uma vez feliz com o port, a única coisa que resta é colocá-lo na árvore principal do FreeBSD e deixar todo mundo feliz também.

> Nós não precisamos do diretório work ou do pacote pkgname.tgz, então exclua-os agora.

Em seguida, crie um patch(1). Assumindo que o port é chamado `oneko` e está na categoria `games`.

Exemplo 1. Criando um .diff para um Novo Port

Adicione todos os arquivos com `svn add`. Utilize o `cd` e vá para a base da árvore de ports, para que os caminhos completos dos arquivos alterados sejam incluídos no diff, então gere o diff com `svn diff`. Por exemplo:

```
% svn add .
% cd ../..
% svn diff games/oneko > oneko.diff
```

> Para ser mais fácil para os committers aplicarem o patch em sua cópia de trabalho da árvore de ports, por favor, gere o .diff da base da sua árvore de ports.

Envie o oneko.diff com o formulário de submissão de bugs. Use product "Ports & Packages", component "Individual Port(s)" e siga as diretrizes mostradas lá. Adicione uma breve descrição do programa ao campo Description do PR (talvez uma versão curta do `COMMENT`), e lembre-se de adicionar o oneko.diff como um anexo.

> Dar uma boa descrição no resumo do relatório de problema facilita muito o

> trabalho dos commiters de ports. Preferimos algo como "New port: `*category/portname breve descrição do port*`" para novos ports. Usar este esquema torna mais fácil e rápido começar o trabalho para fazer o commit de um novo port.

Depois de enviar o port, por favor, seja paciente. O tempo necessário para incluir um novo port no FreeBSD pode variar de alguns dias até alguns meses. Um formulário simples de pesquisa no banco de dados do Relatório de Problemas está disponível em https://bugs.freebsd.org/bugzilla/query.cgi.

Para obter uma listagem dos PRs *abertos* para os ports, selecione *Open* e *Ports & Packages* no formulário de pesquisa, clique em **[Search]**.

Depois de analisar o novo port, nós responderemos se necessário, e iremos adicioná-lo a árvore. O nome do remetente também será adicionado à lista de Contribuidores Adicionais do FreeBSD e outros arquivos.

Também é possível enviar ports usando um arquivo shar(1). Usando o exemplo anterior com o port `oneko` acima.

Exemplo 2. Criando um .shar para um Novo Port

vá para o diretório acima, onde o diretório do port está localizado, e use `tar` para criar o arquivo shar:

```
% cd ..
% tar cf oneko.shar --format shar oneko
```

oneko.shar pode ser enviado da mesma maneira que oneko.diff acima.

Capítulo 4. Port Lento

Certo, então não foi tão simples e o port precisou de algumas modificações para poder funcionar. Nesta seção, vamos explicar passo a passo como modificá-lo para que funcione com o paradigma do ports.

4.1. Como as Coisas Funcionam

Primeiro, esta é a sequência de eventos que ocorre quando o usuário executa `make` no diretório do port. Ter o bsd.port.mk aberto em outra janela enquanto lê esta seção realmente irá ajudar a entender melhor.

Mas não se preocupe, não são muitas as pessoas que entendem exatamente como o bsd.port.mk funciona...*:-)*

1. O target `fetch` é executado. O target `fetch` é responsável por garantir que o tarball exista localmente em `DISTDIR`. Se o `fetch` não puder encontrar os arquivos necessários no `DISTDIR` ele procurará a URL na variável `MASTER_SITES`, definida no Makefile, assim como nos nossos mirrors FTP nos quais colocamos os distfiles como backup. Em seguida, ele tentará buscar o arquivo de distribuição nomeado com `FETCH`, assumindo que o site solicitante tem acesso direto à Internet. Se isso for bem sucedido, ele salvará o arquivo em `DISTDIR` para uso futuro e continuará.
2. O target `extract` é executado. Ele procura pelo arquivo de distribuição do port (normalmente um tarball compactado) em `DISTDIR` e irá descompactá-lo em um subdiretório temporário especificado por `WRKDIR` (padrão é work).
3. O target `patch` é executado. Primeiro, quaisquer patches definidos em `PATCHFILES` são aplicados. Segundo, se arquivos de patch nomeados patch-* forem encontrados em `PATCHDIR` (padrão para o subdiretório files), eles serão aplicados neste momento em ordem alfabética.
4. O target `configure` é executado. Ele pode fazer qualquer uma de muitas coisas diferentes.
   a. Se existir, scripts/configure é executado.
   b. E se `HAS_CONFIGURE` ou `GNU_CONFIGURE` está definido, WRKSRC/configure é executado.
5. O target `build` é executado. Ele é responsável por mudar para o diretório de trabalho privado do port (`WRKSRC`) e compila-lo.
6. O target `stage` é executado. Este coloca o conjunto final de arquivos construídos em um diretório temporário (`STAGEDIR`, Veja Staging). A hierarquia deste diretório espelha a do sistema no qual o pacote será instalado.
7. O target `package` é executado. Ele cria um pacote usando os arquivos do diretório temporário criado durante o target `stage` e o pkg-plist do port.
8. O target `install` é executado. Este instala o pacote criado durante o target `package` no host.

As ações acima são padrão. Além disso, defina os targets `pre-something` ou `post-something`, ou insira

scripts com esses nomes no subdiretório scripts, e eles serão executados antes ou depois das ações padrão serem executadas.

Por exemplo, se houver um target `post-extract` definido no Makefile e um arquivo pre-build no subdiretório scripts, o target `post-extract` será chamado após as ações de extração regulares e pre-build será executado antes que as regras de compilação padrão sejam feitas. Recomenda-se usar targets no Makefile se as ações forem simples, porque será mais fácil para alguém descobrir que tipo de ação não padrão o port necessita.

As ações padrão são feitas pelos targets `do-something` do bsd.port.mk. Por exemplo, os comandos para extrair um port estão no target `do-extract`. Se o target padrão não fizer o trabalho direito, redefina o target `do-something` no Makefile.

> O target "principal" (por exemplo, `extract`, `configure`, etc.) fazem nada mais do que certificar-se de que todos os estágios até aquele estão concluídos e chamar os targets ou scripts reais, e eles não pretendem ser alterados. Para consertar a extração, corrija `do-extract`, mas nunca mude a forma como `extract` opera! Além disso, o target `post-deinstall` é inválido e não é executado pela infraestrutura de ports.

Agora que o que acontece quando o usuário digita `make install` é melhor entendido, vamos seguir as etapas recomendadas para criar o port perfeito.

4.2. Obtendo os Fontes Originais

Obtenha os fontes originais (normalmente) como um tarball compactado (foo.tar.gz ou foo.tar.bz2) e copie-o para `DISTDIR`. Use fontes do *mainstream* sempre que possível.

Definir a variável `MASTER_SITES` para refletir onde o tarball original reside. Existem definições abreviadas para a maioria dos sites mainstream em bsd.sites.mk. Por favor, use esses sites - e as definições associadas—se for possível, para ajudar a evitar o problema de ter as mesmas informações repetidas várias vezes na base de origem. Como esses sites tendem a mudar com o tempo, isso se torna um pesadelo de manutenção para todos os envolvidos. Veja `MASTER_SITES` para detalhes.

Se não houver nenhum site FTP/HTTP bem conectado à rede ou se puder encontrar apenas sites com formatos irritantemente não-padrão, coloque uma cópia em um servidor FTP ou HTTP confiável (por exemplo, uma home page).

Se um lugar conveniente e confiável para colocar o distfile não puder ser encontrado, nós podemos "hospedar" em `ftp.FreeBSD.org`; no entanto, esta é a solução menos preferida. O distfile deve ser colocado em ~/public_distfiles/ da conta `freefall` de alguém. Peça para a pessoa que for fazer o commit do port para realizer isso. Essa pessoa também irá definir `MASTER_SITES` para `LOCAL/username` onde `username` é o seu login do cluster do FreeBSD.

Se o distfile do port mudar o tempo todo sem nenhum tipo de atualização de versão pelo autor, considere colocar o distfile em uma página pessoal e liste-a como o `MASTER_SITES` primário. Tente falar com o autor do port para parar de fazer isso; Isso realmente ajuda a estabelecer algum tipo de controle de código-fonte. Hospedar uma versão específica impedirá que os usuários obtenham

erros de `checksum mismatch`, e também irá reduzir a carga de trabalho dos mantenedores do nosso site FTP. Além disso, se houver apenas um site master para o port, recomenda-se armazenar um backup em uma home page e listá-lo como o `MASTER_SITES` secundário.

Se o port exigir patches adicionais disponíveis na Internet, baixe-os também e coloque-os em `DISTDIR`. Não se preocupe se eles vierem de um site diferente de onde vem o tarball do código fonte principal, temos uma maneira de lidar com essas situações (veja a descrição PATCHFILES abaixo).

4.3. Modificando o Port

Desempacote uma cópia do tarball em um diretório privado e faça as alterações necessárias para que o port compile corretamente sob a versão atual do FreeBSD. *Atenção dobrada* nessas etapas, pois elas serão necessárias para automatizar o processo em breve. Tudo, incluindo a exclusão, adição ou modificação de arquivos, devem ser realizados usando um script automatizado ou um arquivo patch quando o port estiver finalizado.

Se o port exigir interação/customização significativa do usuário para compilar ou instalar, dê uma olhada em um dos scripts Configure clássicos de Larry Wall e talvez faça algo semelhante. O objetivo da nova coleção de ports é fazer com que cada port seja "plug-and-play" o quanto possível para o usuário final, usando um mínimo de espaço em disco.

> A menos que explicitamente declarado, os arquivos de patch, scripts e outros arquivos criados e contribuídos para a coleção de ports do FreeBSD são assumidos como cobertos pelas condições de copyright padrão do BSD.

4.4. Patching

Na preparação do port, arquivos que forem adicionados ou alterados podem ser gravados com diff(1) para posterior inclusão em um patch(1). Fazer isso com um arquivo típico envolve salvar uma cópia do arquivo original antes de fazer qualquer alteração usando um sufixo .orig.

```
% cp file file.orig
```

Depois que todas as alterações forem realizadas, `cd` de volta ao diretório do port. Execute `make makepatch` para gerar arquivos de patch atualizados no diretório files.

> Usar `BINARY_ALIAS` para substituir comandos codificados durante a compilação e para evitar patching de arquivos de compilação. Veja Use `BINARY_ALIAS` para Renomear Comandos Em Vez de Aplicar Patch na Compilação para maiores informações.

4.4.1. Regras Gerais para Patching

Arquivos patch são armazenados em `PATCHDIR`, geralmente files/, de onde serão aplicados automaticamente. Todas os patches devem ser relativos ao `WRKSRC`. Tipicamente `WRKSRC` é um subdiretório de `WRKDIR`, o diretório onde o distfile é extraído. Execute `make -V WRKSRC` para ver o

caminho real. Os nomes dos patches devem seguir estas regras:

- Evite ter mais de um patch modificando o mesmo arquivo. Por exemplo, ter os dois patch-foobar.c e patch-foobar.c2 fazendo alterações em ${WRKSRC}/foobar.c torna-os frágeis e difíceis de serem depurados.
- Ao criar nomes para arquivos de patch, substitua cada underline (`_`) com dois underlines (`__`) e cada barra (`/`) com um underline (`_`). Por exemplo, para corrigir um arquivo chamado `src/freeglut_joystick.c` nomeie o patch correspondente `patch-src_freeglut_joystick.c`. Não nomeie patches como patch-aa ou patch-ab. Sempre use o caminho e o nome do arquivo nos nomes dos patches. O `make makepatch` gera automaticamente os nomes corretos.
- Um patch pode modificar vários arquivos se as alterações estiverem relacionadas e o patch tiver o nome apropriado. Por exemplo, patch-add-missing-stdlib.h.
- Use apenas caracteres `[-+._ a-zA-Z0-9]` para nomear patches. Em particular, não use `::` como um separador de path, use _ no lugar.

Minimize a quantidade de mudanças de espaço em branco não funcionais em patches. É comum no mundo Open Source para projetos compartilhar grandes quantidades de uma base de código, mas obedecer a regras de recuo e estilo diferentes. Ao usar uma funcionalidade funcional de um projeto para consertar áreas similares em outra, por favor, tenha cuidado: o patch resultante pode estar cheio de mudanças não-funcionais. Ele não só aumenta o tamanho do repositório do ports, mas torna difícil descobrir o que exatamente causou o problema e o que foi alterado em todos.

Se um arquivo precisar ser excluído, faça-o no target `post-extract` em vez de como parte do patch.

4.4.2. Geração Manual de Patches

> A criação manual de patches geralmente não é necessária. A geração automática de patches, conforme descrito anteriormente nesta seção, é o método preferido. No entanto, patches manuais podem ser necessários ocasionalmente.

Patches são salvos em arquivos nomeados como patch-* onde * indica o nome do caminho do arquivo que está sendo feito o patch, como patch-imakefile ou patch-src-config.h.

Depois que o arquivo foi modificado, diff(1) é usado para registrar as diferenças entre a versão original e a modificada. `-u` faz com que o diff(1) produza diffs "unificados", a forma preferida.

```
% diff -u file.orig file > patch-pathname-file
```

Ao gerar patches para novos arquivos adicionados, `-N` é usado para dizer ao diff(1) para tratar o arquivo original inexistente como se existisse, mas estava vazio:

```
% diff -u -N newfile.orig newfile > patch-pathname-newfile
```

Não adicione Strings RCS `$FreeBSD: head/pt_BR.ISO8859-1/books/porters-handbook/book.xml 54410 2020-08-05 22:13:01Z dbaio $` em patches. Quando os patches são adicionados ao repositório Subversion com `svn add`, a propriedade `fbsd:nokeywords` é definida para `yes` automaticamente para

que as keywords no patch não sejam modificadas no commit. A propriedade pode ser adicionada manualmente `svn propset fbsd:nokeywords yes files…`.

Usar a opção (`-r`) do diff(1) para gerar patches é razoável, mas por favor, analise os patches resultantes para se certificar de que não há nenhum lixo desnecessário neles. Em particular, diffs entre dois arquivos de backup, quando o port usa `Imake` ou GNU `configure`, etc., diffs de Makefiles são desnecessários e devem ser eliminados. Se for necessário editar o configure.in e executar o `autoconf` para regerar o `configure`, não gere diffs do `configure` (ele geralmente cresce para algumas milhares de linhas!). Em vez disso, defina `USES=autoreconf` e gere os diffs no configure.in.

4.4.3. Substituições Automáticas Simples

Substituições simples podem ser realizadas diretamente do Makefile do port usando o modo in-loco do sed(1). Isso é útil quando as alterações usam o valor de uma variável:

```
post-patch:
    @${REINPLACE_CMD} -e 's|/usr/local|${PREFIX}|g' ${WRKSRC}/Makefile
```

Use o sed(1) apenas para substituir conteúdo de variáveis. Você deve usar arquivos patch em vez do sed(1) para substituir conteúdo estático.

Muitas vezes, o software sendo portado usa a convenção CR/LF nos arquivos fonte. Isso pode causar problemas com correções adicionais, avisos do compilador ou execução de scripts (como `/bin/sh^M não encontrado`.) Para converter rapidamente todos os arquivos de CR/LF para apenas LF, adicione essa entrada ao Makefile do port:

```
USES=   dos2unix
```

Uma lista de arquivos específicos para conversão pode ser informada:

```
USES=   dos2unix
DOS2UNIX_FILES= util.c util.h
```

Use `DOS2UNIX_REGEX` para converter um grupo de arquivos em subdiretórios. Seu argumento é um find(1) compatível com expressão regular. Mais sobre o formato está em re_format(7). Esta opção é útil para converter todos os arquivos de uma determinada extensão. Por exemplo, converta todos os arquivos de código-fonte, deixando os arquivos binários intactos:

```
USES=   dos2unix
DOS2UNIX_REGEX= .*\.([ch]|cpp)
```

Uma opção similar é `DOS2UNIX_GLOB`, que executa o `find` para cada elemento listado nele.

```
USES=   dos2unix
```

```
DOS2UNIX_GLOB=  *.c *.cpp *.h
```

O diretório base para a conversão pode ser definido. Isso é útil quando há vários distfiles e vários arquivos contidos que requerem conversão de fim de linha.

```
USES=   dos2unix
DOS2UNIX_WRKSRC=    ${WRKDIR}
```

4.4.4. Corrigindo Condicionalmente

Alguns ports precisam de patches que são aplicados apenas para versões específicas do FreeBSD ou quando uma determinada opção é ativada ou desativada. Os patches condicionais são especificados colocando-se os caminhos completos para os arquivos de patch em `EXTRA_PATCHES`.

Exemplo 3. Aplicando um Patch para uma Versão Específica do FreeBSD

```
.include <bsd.port.options.mk>

# Patch in the iconv const qualifier before this
.if ${OPSYS} == FreeBSD && ${OSVERSION} < 1100069
EXTRA_PATCHES=  ${PATCHDIR}/extra-patch-fbsd10
.endif

.include <bsd.port.mk>
```

Exemplo 4. Aplicando Opcionalmente um Patch

Quando um option requer um patch, use `OPT_EXTRA_PATCHES` e `OPT_EXTRA_PATCHES_OFF` para fazer o patch condicional na opção `opt`. Veja Substituição de Variáveis Genéricas, `OPT_VARIABLE_` e `OPT_VARIABLE_OFF` Para maiores informações.

```
OPTIONS_DEFINE=   FOO BAR
FOO_EXTRA_PATCHES=  ${PATCHDIR}/extra-patch-foo
BAR_EXTRA_PATCHES_OFF=  ${PATCHDIR}/extra-patch-bar.c \
        ${PATCHDIR}/extra-patch-bar.h
```

Exemplo 5. Usando `EXTRA_PATCHES` Com um Diretório

As vezes, existem muitos patches que são necessários para um recurso, neste caso, é possível apontar `EXTRA_PATCHES` para um diretório, e ele aplicará automaticamente todos os arquivos nomeados como patch* nele.

Crie um subdiretório em ${PATCHDIR}, e mova os patches para ele. Por exemplo:

```
% ls -l files/foo-patches
-rw-r--r--  1 root  wheel    350 Jan 16 01:27 patch-Makefile.in
-rw-r--r--  1 root  wheel   3084 Jan 18 15:37 patch-configure
```

Então adicione isso ao Makefile:

```
OPTIONS_DEFINE= FOO
FOO_EXTRA_PATCHES=  ${PATCHDIR}/foo-patches
```

O framework irá então usar todos os arquivos nomeados patch* nesse diretório.

4.5. Configurando

Inclua quaisquer comandos de personalização adicionais no script configure e salve-o no subdiretório scripts. Como mencionado acima, também é possível fazer isso com targets no Makefile e/ou scripts com o nome pre-configure ou post-configure.

4.6. Manipulando a Entrada do Usuário

Se o port requer intervenção do usuário para build, configure ou install, defina `IS_INTERACTIVE` no Makefile. Isso fará com que os "overnight builds" pulem ele. Se o usuário definir a variável `BATCH` em seu ambiente (e se o usuário definir a variável `INTERATIVE`, então *apenas* aqueles ports que requerem interação serão compilados). Isso economizará muito tempo perdido no conjunto de máquinas que continuamente compilam ports (veja abaixo).

Também é recomendado que, se houver respostas padrão razoáveis para as perguntas, `PACKAGE_BUILDING` pode usado para desativar a intervenção do usuário quando o mesmo estiver definido. Isso nos permitirá compilar os pacotes para CDROMs e FTP.

Capítulo 5. Configurando o Makefile

Configurar o Makefile é bastante simples e, novamente, sugerimos examinar os exemplos existentes antes de começar. Além disso, há um Makefile de exemplo neste manual, então dê uma olhada e por favor siga a ordem das variáveis e seções naquele modelo para tornar o port mais fácil para os outros lerem.

Considere estes problemas em sequência durante o projeto do novo Makefile:

5.1. O Código Fonte Original

Ele está em `DISTDIR` como um tarball `gzip` e é chamado de algo como foozolix-1.2.tar.gz? Se assim for, vá para o próximo passo. Caso contrário, o formato do arquivo de distribuição pode necessitar da substituição de uma ou mais das variáveis `DISTVERSION`, `DISTNAME`, `EXTRACT_CMD`, `EXTRACT_BEFORE_ARGS`, `EXTRACT_AFTER_ARGS`, `EXTRACT_SUFX` ou `DISTFILES`.

Na pior das hipóteses, crie um target personalizado `do-extract` para substituir o padrão. Isso raramente é necessário.

5.1.1. Nomeando

A primeira parte do Makefile do port o nomeia, descreve seu número de versão e o lista na categoria correta.

5.1.2. `PORTNAME`

Setar `PORTNAME` ao nome base do software. Isso é usado como base para o pacote do FreeBSD, e para o `DISTNAME`.

> O nome do pacote deve ser único em toda a árvore de ports. Certifique-se de que o `PORTNAME` já não está em uso por um port existente, e que nenhum outro port já tem o mesmo `PKGBASE`. Se o nome já tiver sido usado, adicione `PKGNAMEPREFIX` ou `PKGNAMESUFFIX`.

5.1.3. Versões, `DISTVERSION` ou `PORTVERSION`

Setar `DISTVERSION` para o número da versão do software.

`PORTVERSION` é a versão usada para o pacote do FreeBSD. Será automaticamente derivado de `DISTVERSION` para ser compatível com o esquema de versionamento de pacotes do FreeBSD. Se a versão contiver *letras*, pode ser necessário definir `PORTVERSION` e não `DISTVERSION`.

> Não é possível utilizar `PORTVERSION` e `DISTVERSION` juntos, deve ser ser definido um de cada vez.

De tempos em tempos, alguns softwares usam um esquema de versão que não é compatível em como o `DISTVERSION` traduz a versão no `PORTVERSION`.

Ao atualizar um port, é possível usar o pkg-version(8)-t para verificar se a nova versão é maior ou menor do que antes. Veja Usando pkg-version(8) para comparar versões..

Exemplo 6. Usando pkg-version(8) para comparar versões.

`pkg version -t` recebe duas versões como argumentos, responderá com `<`, `=` ou `>` se a primeira versão for menor, igual ou maior que a segunda versão, respectivamente.

```
% pkg version -t 1.2 1.3
< ①
% pkg version -t 1.2 1.2
= ②
% pkg version -t 1.2 1.2.0
= ③
% pkg version -t 1.2 1.2.p1
> ④
% pkg version -t 1.2.a1 1.2.b1
< ⑤
% pkg version -t 1.2 1.2p1
< ⑥
```

① `1.2` é menor que `1.3`.

② `1.2` e `1.2` são iguais, pois têm a mesma versão.

③ `1.2` e `1.2.0` são iguais, pois valor vazio é igual a zero.

④ `1.2` é maior que `1.2.p1` por causa do `.p1`, pense em "pre-release 1".

⑤ `1.2.a1` é menor que `1.2.b1`, pense em "alfa" e "beta" e `a` é menor que `b`.

⑥ `1.2` é menor que `1.2p1` por causa do `2p1`, pense em "2, nível de patch 1" que é uma versão depois de qualquer `2.X` mas antes de `3`.

Aqui, `a`, `b` e `p` são usados como se significassem "alfa", "beta" ou "pre-release" e "nível de patch", mas elas são apenas letras e são classificados por ordem alfabética, portanto, qualquer letra pode ser utilizada, e elas serão ordenadas de forma adequada.

Tabela 1. Exemplos de `DISTVERSION` e de Derivações `PORTVERSION`

DISTVERSION	PORTVERSION
0.7.1d	0.7.1.d
10Alpha3	10.a3
3Beta7-pre2	3.b7.p2
8:f_17	8f.17

Exemplo 7. Usando `DISTVERSION`

Quando a versão contém apenas números separados por pontos, traços ou sublinhados, use `DISTVERSION`.

```
PORTNAME=   nekoto
DISTVERSION=    1.2-4
```

Isso irá gerar um `PORTVERSION 1.2.4`.

Exemplo 8. Usando `DISTVERSION` *Quando a Versão Começa com uma Letra ou um Prefixo*

Quando a versão começa ou termina com uma letra, um prefixo ou um sufixo que não faz parte da versão, use `DISTVERSIONPREFIX`, `DISTVERSION` e `DISTVERSIONSUFFIX`.

Se a versão for `v1.2-4`:

```
PORTNAME=   nekoto
DISTVERSIONPREFIX=  v
DISTVERSION=    1_2_4
```

Algumas vezes, projetos usando GitHub usará seu nome em suas versões. Por exemplo, a versão pode ser `nekoto-1.2-4`:

```
PORTNAME=   nekoto
DISTVERSIONPREFIX=  nekoto-
DISTVERSION=    1.2_4
```

Esses projetos também usam algumas strings no final da versão, por exemplo,`1.2-4_RELEASE`:

```
PORTNAME=   nekoto
DISTVERSION=    1.2-4
DISTVERSIONSUFFIX=  _RELEASE
```

Ou eles fazem ambos, por exemplo,`nekoto-1.2-4_RELEASE`:

```
PORTNAME=   nekoto
DISTVERSIONPREFIX=  nekoto-
DISTVERSION=    1.2-4
DISTVERSIONSUFFIX=  _RELEASE
```

`DISTVERSIONPREFIX` e `DISTVERSIONSUFFIX` não serão usados durante a construção do `PORTVERSION`, mas usado apenas em `DISTNAME`.

Todos exemplos irão gerar um PORTVERSION com valor 1.2.4.

Exemplo 9. Usando DISTVERSION Quando a Versão Contém Letras Significando "alpha", "beta" ou "pre-release"

Quando a versão contém números separados por pontos, traços ou underlines, e letras são usadas para significar "alpha", "beta" ou "pre-release", no sentido de que vem antes das versões sem letras, use DISTVERSION.

```
PORTNAME=   nekoto
DISTVERSION=    1.2-pre4
```

```
PORTNAME=   nekoto
DISTVERSION=    1.2p4
```

Ambos irão gerar um PORTVERSION com valor 1.2.p4 que é menor do que 1.2. pkg-version(8) pode ser usado para verificar esse fato:

```
% pkg version -t 1.2.p4 1.2
<
```

Exemplo 10. Não use DISTVERSION Quando a Versão Contém Letras que Significam "Nível de Patch"

Quando a versão contém letras que não significam "alpha", "beta" ou "pre", e estão mais para um "nível de patch", no sentido de que vem depois da versão sem as letras, use PORTVERSION.

```
PORTNAME=   nekoto
PORTVERSION=    1.2p4
```

Neste caso, usar DISTVERSION não é possível porque geraria uma versão 1.2.p4 o que seria menor que 1.2 e não maior.pkg-version(8) irá constatar isso:

```
% pkg version -t 1.2 1.2.p4
> ①
% pkg version -t 1.2 1.2p4
< ②
```

① 1.2 é maior que 1.2.p4, o que é *errado* nesse caso.

② 1.2 é menor que 1.2p4, que é o que era necessário.

Para alguns exemplos mais avançados de configuração do PORTVERSION, quando a versão do software não é realmente compatível com o FreeBSD, ou DISTNAME quando o arquivo de distribuição

não contém a versão em si, consulte DISTNAME.

5.1.4. PORTREVISION e PORTEPOCH

PORTREVISION

PORTREVISION é um valor monotonicamente crescente que é redefinido para 0 com cada incremento de DISTVERSION, normalmente toda vez que houver uma nova versão oficial do fornecedor. E se PORTREVISION é diferente de zero, o valor é anexado ao nome do pacote. Mudanças em PORTREVISION são usadas por ferramentas automatizadas como pkg-version(8) para determinar se um novo pacote está disponível.

PORTREVISION deve ser incrementado toda vez que uma alteração for feita no port onde se altera o pacote gerado de alguma forma. Isso inclui alterações que afetam apenas um pacote compilado com options não padrão.

Exemplos de quando PORTREVISION deve ser alterado:

- Adição de correções para corrigir vulnerabilidades de segurança, bugs ou para adicionar novas funcionalidades ao port.
- Alterações no Makefile do port para ativar ou desativar as opções de tempo de compilação no pacote.
- Alterações na lista de empacotamento ou no comportamento de tempo de instalação do pacote. Por exemplo, uma alteração em um script que gera dados iniciais para o pacote, como chaves de host ssh(1).
- Bump de versão da dependência de biblioteca compartilhada de um port (nesse caso, alguém tentando instalar o pacote antigo depois de instalar uma versão mais nova da dependência falhará, pois procurará a libfoo.x antiga em vez da libfoo.(x+1)).
- Mudanças silenciosas no distfile do port que possuem diferenças funcionais significativas. Por exemplo, mudanças no distfile que requerem uma correção para distinfo sem alteração correspondente para DISTVERSION, onde um `diff -ru` das versões antiga e nova mostra mudanças não triviais no código.

Exemplos de alterações que não requerem uma alteração no PORTREVISION:

- Mudanças de estilo no esqueleto do port sem alteração funcional ao que aparece no pacote resultante.
- Mudanças para MASTER_SITES ou outras alterações funcionais no port que não afetem o pacote resultante.
- Patches triviais para o distfile, como correção de erros de digitação, que não são importantes o suficiente para que os usuários do pacote tenham que se dar ao trabalho de atualizar.
- Correções de compilação que fazem com que um pacote se torne compilável onde antes estava falhando. Desde que as alterações não introduzam nenhuma mudança funcional em nenhuma outra plataforma na qual o port tenha sido compilado anteriormente. PORTREVISION reflete o conteúdo do pacote, se o pacote não foi compilado anteriormente, então não há necessidade de incrementar o PORTREVISION para registrar uma mudança.

Uma regra geral é decidir se a mudança em um port é algo que *algumas* pessoas se beneficiariam em ter. Por causa de um aprimoramento, conserto ou em virtude de que o novo pacote funcione de fato. Em seguida, pondere que, de fato, isso fará com que todos que regularmente atualizam sua árvore de ports sejam obrigados a atualiza-lo. Se sim, `PORTREVISION` deve ser incrementado.

> Pessoas usando pacotes binários *nunca* verão a atualização se `PORTREVISION` não for incrementado. Sem incrementar `PORTREVISION`, os package builders não têm como detectar a alteração e, portanto, não irão recompilar o pacote.

`PORTEPOCH`

De tempos em tempos, um fornecedor de software ou um mantenedor de port do FreeBSD fazem algo tolo e lançam uma versão de seu software que é numericamente menor que a versão anterior. Um exemplo disso é um port que vai de foo-20000801 para foo-1.0 (o primeiro será incorretamente tratado como uma versão mais nova, já que 20000801 é um valor numericamente maior que 1).

> Os resultados das comparações de números de versão nem sempre são óbvios. `pkg version` (veja pkg-version(8)) pode ser usado para testar a comparação de duas sequências de números de versão. Por exemplo:
>
> ```
> % pkg version -t 0.031 0.29
> >
> ```
>
> A saida `>` indica que a versão 0.031 é considerada maior que a versão 0.29, o que pode não ter sido óbvio para o mantenedor do port.

Em situações como essa, `PORTEPOCH` deve ser incrementado. E se `PORTEPOCH` é diferente de zero, ele é anexado ao nome do pacote conforme descrito na seção 0 acima. `PORTEPOCH` nunca deve ser diminuído ou redefinido para zero, porque isso faria com que a comparação com um pacote de uma época anterior falhasse. Por exemplo, o pacote não seria detectado como desatualizado. O novo número da versão, `1.0.1` no exemplo acima, ainda é numericamente menor que a versão anterior, 20000801, mas o sufixo `1` é tratado especialmente por ferramentas automatizadas e considerado maior que o sufixo `0` implícito no pacote anterior.

Remover ou resetar o `PORTEPOCH` incorretamente conduz ao luto eterno. Se a discussão acima não foi clara o suficiente, por favor consulte a Lista de discussão de ports do FreeBSD.

É esperado que `PORTEPOCH` não seja utilizado na maioria dos ports, e que seja feito o uso sensato do `DISTVERSION`, ou que o `PORTVERSION` seja usado com cuidado também, isso muitas vezes pode evitar que uma versão futura do software altere a estrutura da versão. No entanto, é necessário que os porters do FreeBSD tenham cuidado quando uma versão do fornecedor é feita sem um número de versão oficial - como um código de release "snapshot". A tentação é rotular a release com a data de lançamento, o que causará problemas como no exemplo acima, quando um novo release "oficial" é feito.

Por exemplo, se um snapshot de release é feito na data `20000917` e a versão anterior do software era a versão `1.2`, não use `20000917` no `DISTVERSION`. A maneira correta é um `DISTVERSION` com valor `1.2.20000917`, ou similar, para que a próxima versão, digamos `1.3`, ainda seja um valor

numericamente maior.

Exemplo de Uso `PORTREVISION` e `PORTEPOCH`

O port `gtkmumble`, versão `0.10` está comitado na coleção de ports:

```
PORTNAME=   gtkmumble
DISTVERSION=    0.10
```

`PKGNAME` torna-se `gtkmumble-0.10`.

Uma falha de segurança é descoberta, o que requer um patch local do FreeBSD. `PORTREVISION` é alterado de acordo.

```
PORTNAME=   gtkmumble
DISTVERSION=    0.10
PORTREVISION=   1
```

`PKGNAME` torna-se `gtkmumble-0.10_1`

Uma nova versão é lançada pelo fornecedor, numerada como `0.2` (acontece que o autor realmente pretendia que `0.10` significa-se realmente `0.1.0`, não "o que vem depois de 0.9" - oops, tarde demais agora). Como a nova versão secundária `2` é numericamente menor que a versão anterior `10`, `PORTEPOCH` deve ser incrementado para forçar manualmente que o novo pacote seja detectado como "mais recente". Como é uma nova versão do fornecedor, `PORTREVISION` é redefinido para 0 (ou removido do Makefile).

```
PORTNAME=   gtkmumble
DISTVERSION=    0.2
PORTEPOCH=  1
```

`PKGNAME` torna-se `gtkmumble-0.2,1`

O próximo lançamento é 0.3. Desde que `PORTEPOCH` nunca diminua, as variáveis de versão são agora:

```
PORTNAME=   gtkmumble
DISTVERSION=    0.3
PORTEPOCH=  1
```

`PKGNAME` torna-se `gtkmumble-0.3,1`

E se `PORTEPOCH` for redefinido para `0` com esta atualização, alguém que instalou o `gtkmumble-0.10_1` não detectaria o `gtkmumble-0.3` como pacote mais novo, desde que `3` ainda é numericamente menor que `10`. Lembre-se, este é o ponto principal de `PORTEPOCH` em primeiro lugar.

5.1.5. PKGNAMEPREFIX e PKGNAMESUFFIX

Duas variáveis opcionais, PKGNAMEPREFIX e PKGNAMESUFFIX, são combinadas com PORTNAME e PORTVERSION para formar PKGNAME como ${PKGNAMEPREFIX}${PORTNAME}${PKGNAMESUFFIX}-${PORTVERSION}. Certifique-se de que isto está de acordo com as nossas diretrizes para um bom nome de pacote. Em particular, o uso de um hífen (-) dentro de PORTVERSION *não é* permitido. Além disso, se o nome do pacote tiver o *language-* ou a parte *-compiled.specifics* (veja abaixo), use PKGNAMEPREFIX e PKGNAMESUFFIX, respectivamente. Não os faça parte de PORTNAME.

5.1.6. Convenções de Nomenclatura de Pacotes

Estas são as convenções a serem seguidas ao nomear pacotes. Isso é para facilitar a varredura do diretório de pacotes, já que existem milhares de pacotes e os usuários irão pegar ranço se eles machucarem seus olhos!

Nomes de pacotes tomam a forma de language_region-name-compiled.specifics-version.numbers.

O nome do pacote é definido como ${PKGNAMEPREFIX}${PORTNAME}${PKGNAMESUFFIX}-${PORTVERSION}. Certifique-se de definir as variáveis para estar em conformidade com esse formato.

language_region-

O FreeBSD se esforça para suportar a linguagem nativa de seus usuários. A parte *language-* é uma abreviação de duas letras da linguagem natural definida pela ISO-639 quando o port é específico para um determinado idioma. Exemplos são ja para japonês, ru para russo,vi para vietnamita, zh para o chinês, ko para coreano e de para alemão.

Se o port for específico de uma determinada região dentro da área de idioma, adicione também o código do país de duas letras. Exemplos são en_US para Inglês dos EUA e fr_CH para o Francês Suíço.

A parte *language-* é definida em PKGNAMEPREFIX.

name

Certifique-se de que o nome e a versão do port estejam claramente separados e colocados em PORTNAME e DISTVERSION. A única razão para PORTNAME conter uma parte da versão é se a distribuição upstream é realmente chamada dessa forma, como no textproc/libxml2 ou japanese/kinput2-freewnn. De outra forma, PORTNAME não pode conter informações específicas da versão. É normal que vários ports tenham o mesmo PORTNAME, como os ports www/apache* fazem; Nesse caso, versões diferentes (e entradas de índice diferentes) são distinguidas por valores PKGNAMEPREFIX e PKGNAMESUFFIX.

Há uma tradição de nomear módulos Perl 5 com sufixo p5- e convertendo o separador de dois pontos para um hífen. Por exemplo, o modulo Data::Dumper torna-se p5-Data-Dumper.

-compiled.specifics

Se o port pode ser construído com diferentes padrões codificados (geralmente parte do nome do diretório em uma família de ports), a parte *-compiled.specifics* indica os padrões compilados. O hífen é opcional. Exemplos são tamanho de papel e unidades de fonte.

A parte *-compiled.specifics* é definida em `PKGNAMESUFFIX`.

-version.numbers

A string da versão segue um hífen (`-`) e é uma lista separada por pontos de números inteiros e letras minúsculas. Em particular, não é permitido ter outro hífen dentro da string de versão. A única exceção é a string `pl` (significando "patchlevel"), que pode ser usado *apenas* quando não há números de versão maiores e menores no software. Se a versão do software tiver sequências como "alpha", "beta", "rc" ou "pre", use a primeira letra e coloque imediatamente após um ponto. Se a sequência da versão continuar após esses nomes, os números seguirão o alfabeto simples sem um ponto extra entre eles (por exemplo,`1.0b2`).

A ideia é facilitar a classificação dos ports observando a string de versão. Em particular, certifique-se de que os componentes do número da versão estejam sempre delimitados por um ponto e, se a data fizer parte da string, use o formato `dyyyy.mm.dd`, não `dd.mm.yyyy` ou o não compatível com o formato Y2K `yy.mm.dd`. É importante prefixar a versão com uma letra, aqui `d` (para data), no caso de uma versão com um número de versão real, que seria numericamente inferior a `yyyy`.

O nome do pacote deve ser único entre todos os ports, verifique se ainda não existe um port com o mesmo `PORTNAME` e se houver, adicione um dos `PKGNAMEPREFIX` ou `PKGNAMESUFFIX`.

Aqui estão alguns exemplos (reais) de como converter o nome como chamado pelos autores do software para um nome de pacote adequado, para cada linha, apenas um dos `DISTVERSION` ou `PORTVERSION` está definido, dependendo de qual seria usado no Makefile:

Tabela 2. Exemplos de Nomes de Pacotes

Nome da Distribuição	PKGNAMEPREFIX	PORTNAME	PKGNAMESUFFIX	DISTVERSION	PORTVERSION	Razão ou comentário
mule-2.2.2	(vazio)	mule	(vazio)	2.2.2		Nenhuma alteração é necessária
mule-1.0.1	(vazio)	mule	1	1.0.1		Esta é a versão 1 do mule e a versão 2 já existe
EmiClock-1.0.2	(vazio)	emiclock	(vazio)	1.0.2		Sem nomes em maiúsculas para programas individuais
rdist-1.3alpha	(vazio)	rdist	(vazio)	1.3alfa		Versão será `1.3.a`

Nome da Distribuição	PKGNAMEPREFIX	PORTNAME	PKGNAMESUFFIX	DISTVERSION	PORTVERSION	Razão ou comentário
es-0.9-beta1	(vazio)	es	(vazio)	0.9-beta1		Versão será `0.9.b1`
mailman-2.0rc3	(vazio)	mailman	(vazio)	2.0rc3		Versão será `2.0.r3`
v3.3beta021.src	(vazio)	tiff	(vazio)		3.3	O que diabos foi isso afinal?
tvtwm	(vazio)	tvtwm	(vazio)		p11	Nenhuma versão no nome do arquivo, use o que o upstream diz que é
piewm	(vazio)	piewm	(vazio)	1.0		Nenhuma versão no nome do arquivo, use o que o upstream diz que é
xvgr-2.10pl1	(vazio)	xvgr	(vazio)		2.10.pl1	Nesse caso,`pl1` significa nível de patch, então usar DISTVERSION não é possível.
gawk-2.15.6	ja-	gawk	(vazio)	2.15.6		Versão em japonês
psutils-1.13	(vazio)	psutils	-letter	1.13		Tamanho do papel codificado no tempo de compilação do pacote
pkfonts	(vazio)	pkfonts	300	1.0		Pacote para fontes de 300dpi

Se não houver absolutamente nenhum rastro de informações de versão co código fonte original e é improvável que o autor original vá liberar outra versão, basta definir a string de versão para `1.0` (como o exemplo `piewm` acima). Caso contrário, pergunte ao autor original ou use a string de data com valor de quando a código fonte foi lançado como (`dyyyy.mm.dd` ou `dyyyymmdd`) como a versão.

> Use qualquer letra. Aqui,`d` significa data, se o código for um repositório do Git, `g` seguido pela data de commit é normalmente utilizado, `s` para snapshot também é comum.

5.2. Categorização

5.2.1. `CATEGORIES`

Quando um pacote é criado, ele é colocado em /usr/ports/packages/All e links são feitos de um ou mais subdiretórios de /usr/ports/packages. Os nomes desses subdiretórios são especificados pela variável `CATEGORIES`. O objetivo é facilitar a vida do usuário quando ele estiver vasculhando a pilha de pacotes no site FTP ou no CD-ROM. Por favor, dê uma olhada na lista atual de categorias e escolha as que são adequadas para o port.

Esta lista também determina de onde, na árvore de ports, o port será importado. Se houver mais de uma categoria aqui, os arquivos do port devem ser colocados no subdiretório com o nome da primeira categoria. Veja abaixo para mais informação sobre como escolher as categorias certas.

5.2.2. Lista Atual de Categorias

Aqui está a lista atual de categorias de ports. As marcadas com um asterisco (*) são categorias *virtuais* - aquelas que não possuem um subdiretório correspondente na árvore de ports. Elas são usadas apenas como categorias secundárias e apenas para fins de pesquisa.

> Para categorias não virtuais, há uma descrição de uma linha em `COMMENT` no Makefile desse subdiretório.

Categoria	Descrição	Notas
accessibility	Ports para ajudar usuários com deficiências.	
afterstep *	Ports para apoiar o gerenciador de janelas AfterStep.	
arabic	Suporte ao idioma árabe".	
archivers	Ferramentas de arquivamento.	
astro	Ports astronômicos.	
audio	Suporte de som.	
benchmarks	Utilitários de benchmarking.	
biology	Software relacionado à biologia.	

Categoria	Descrição	Notas
cad	Ferramentas de desenho assistidas por computador.	
chinese	Suporte ao idioma chinês.	
comms	Software de comunicação.	Principalmente software para falar com o port serial.
converters	Conversores de código de caracteres.	
databases	Bancos de dados.	
deskutils	Coisas que costumavam estar na área de trabalho antes dos computadores serem inventados.	
devel	Utilitários de desenvolvimento.	Não coloque bibliotecas aqui só porque são bibliotecas. Elas *não deveriam* estar nesta categoria, a menos que elas realmente não pertençam a nenhum outro lugar.
dns	Software relacionado ao DNS.	
docs *	Meta-ports para documentação do FreeBSD.	
editors	Editores gerais.	Editores especializados entram na seção para essas ferramentas. Por exemplo, um editor de fórmula matemática math, e tem editores como uma segunda categoria.
elisp *	Emacs-lisp ports.	
emulators	Emuladores para outros sistemas operacionais.	Emuladores de terminal *não* estão aqui. Os baseados em X vão para o x11 e baseados em texto para qualquer comms ou misc, dependendo da funcionalidade exata.
enlightenment *	Ports relacionados com o gerenciador de janelas Enlightenment.	
finance	Aplicações monetárias, financeiras e relacionadas.	
french	Suporte ao idioma francês.	

Categoria	Descrição	Notas
ftp	Utilitários de cliente e servidor deFTP.	Se o port fala com FTP e HTTP, coloque-o em ftp com uma categoria secundária de www.
games	Jogos.	
geography *	Software relacionado à geografia.	
german	Suporte ao idioma alemão.	
gnome *	Ports do Projeto GNOME.	
gnustep *	Software relacionado ao ambiente de desktop GNUstep.	
graphics	Utilitários gráficos.	
hamradio *	Software para rádio amador.	
haskell *	Software relacionado à linguagem Haskell.	
hebrew	Suporte ao idioma hebraico.	
hungarian	Suporte de idioma húngaro.	
irc	Utilitários do Internet Relay Chat.	
japanese	Suporte ao idioma japonês.	
java	Software relacionado à linguagem Java™.	A categoria Java não deve ser única para um port. Salvo para ports diretamente relacionadas à linguagem Java, os mantenedores de ports também são encorajados a não usar Java como a principal categoria de um port.
kde *	Ports do Projeto KDE (genérico).	
kde-applications *	Aplicações do Projeto KDE.	
kde-frameworks *	Bibliotecas add-on do Projeto KDE para programação com Qt.	
kde-plasma *	Desktop do Projeto KDE.	
kld *	Módulos carregáveis do kernel.	
korean	Suporte ao idioma coreano.	
lang	Linguagens de programação.	
linux *	Aplicações Linux e utilitários de suporte.	

Categoria	Descrição	Notas
lisp *	Software relacionado à linguagem Lisp.	
mail	Mail software.	
mate *	Ports relacionado ao ambiente de desktop MATE, um fork do GNOME 2.	
math	Software de computação numérica e outras utilidades para matemática.	
mbone *	Aplicações MBone.	
misc	Utilitários diversos	Coisas que não pertencem em nenhum outro lugar. Se possível, tente encontrar uma categoria melhor para o port do que `misc`, como os ports tendem a ser negligenciados aqui.
multimedia	Software multimídia.	
net	Software de rede diversos.	
net-im	Software de mensagens instantâneas.	
net-mgmt	Software de gerenciamento de rede.	
net-p2p	Aplicativos de rede peer to peer.	
net-vpn *	Aplicativos de Rede Privada Virtual.	
news	Software de notícias USENET.	
parallel *	Aplicativos que lidam com o paralelismo na computação.	
pear *	Ports relacionados ao framework PHP Pear.	
perl5 *	Ports que exigem Perl versão 5 para rodar.	
plan9 *	Vários programas de Plan9.	
polish	Suporte ao idioma polonês".	
ports-mgmt	Ports para gerenciar, instalar e desenvolver ports e pacotes do FreeBSD.	
portuguese	Suporte ao idioma Português.	

Categoria	Descrição	Notas
print	Software de Impressão.	As ferramentas de editoração eletrônica (pré-visualizadores etc.) também pertencem aqui.
python *	Software relacionado a linguagem Python.	
ruby *	Software relacionado a linguagem Ruby.	
rubygems *	Ports de pacotes RubyGems.	
russian	Suporte de idioma russo.	
scheme *	Software relacionado à linguagem Scheme.	
science	Ports científicos que não se encaixam em outras categorias, como astro, biologia e matemática.	
security	Utilitários de segurança.	
shells	Linha de comando do shell.	
spanish *	Suporte ao idioma espanhol.	
sysutils	Utilidades do sistema.	
tcl *	Ports que usam o Tcl para rodar.	
textproc	Utilitários de processamento de texto.	Não inclui ferramentas de editoração eletrônica, que vão para print.
tk *	Ports que usam o Tk para rodar.	
ukrainian	Suporte de idioma Ucraniano.	
vietnamese	Suporte de idioma Vietnamita.	
wayland *	Ports para suportar o servidor de display Wayland.	
windowmaker *	Ports para suportar o gerenciador de janelas do WindowMaker.	
www	Software relacionado à World Wide Web.	O suporte ao idioma HTML também pertence aqui.

Categoria	Descrição	Notas
x11	O X Window System e seus amigos.	Esta categoria é apenas para software que suporta diretamente o sistema de janelas. Não coloque aplicativos regulares do X aqui. A maioria deles é usada em outras categorias x11- * (veja abaixo).
x11-clocks	X11 relógios.	
x11-drivers	Drivers X11.	
x11-fm	Gerentes de arquivos X11.	
x11-fonts	Fontes X11 e utilitários de fonte.	
x11-servers	Servidores X11.	
x11-themes	X11 temas.	
x11-toolkits	Kits de ferramentas X11.	
x11-wm	Gerentes de janela do X11.	
xfce *	Ports relacionados com o ambiente de trabalho Xfce.	
zope *	Zope suporte.	

5.2.3. Escolhendo a Categoria Correta

Como muitas das categorias se sobrepõem, escolher qual das categorias será a principal categoria do port pode ser entediante. Existem várias regras que governam essa questão. Aqui está a lista de prioridades, em ordem decrescente de precedência:

- A primeira categoria deve ser uma categoria física (veja acima). Isso é necessário para o empacotamento funcionar. Categorias virtuais e categorias físicas podem ser misturadas depois disso.
- As categorias específicas de idioma sempre vêm em primeiro lugar. Por exemplo, se o port instalar fontes X11 em japonês, a linha `CATEGORIES` deve ser japanese x11-fonts.
- Categorias específicas são listadas antes de outras menos específicas. Por exemplo, um editor de HTML é listado como www editors, e não ao contrário. Além disso, não insira net quando o port pertencer a qualquer uma das categorias irc, mail, news, security ou www, pois net está incluída implicitamente.
- x11 é usado como uma categoria secundária somente quando a categoria principal é uma linguagem natural. Em particular, não coloque x11 na linha de categoria em aplicações X.
- Os modes Emacs são colocados na mesma categoria de ports que a aplicação suportada pelo mode, e não em editors. Por exemplo, um mode Emacs para editar código fonte de alguma linguagem de programação entra em lang.
- Ports que instalam módulos do kernel carregáveis também têm a categoria virtual kld na sua

linha `CATEGORIES`. Esta é uma das coisas tratadas automaticamente adicionando `USES=kmod`.

- misc não aparece com nenhuma outra categoria não virtual. Se houver `misc` com outra categoria na linha `CATEGORIES`, isso significa que `misc` pode ser seguramente excluído e o port colocado apenas no outro subdiretório.
- Se o port realmente não pertencer em nenhum outro lugar, coloque-o em misc.

Se a categoria não estiver claramente definida, por favor, insira um comentário sobre isso na submissão do port no banco de dados de bugs, para que possamos discuti-lo antes de importá-lo. Como committer, envie uma mensagem para a lista de discussão de ports do FreeBSD, para podermos discutir isso primeiro. Com muita frequência, novos ports são importados na categoria errada, e depois são movidos imediatamente para a categoria correta.

5.2.4. Propondo uma Nova Categoria

Como a Coleção de Ports vem crescendo com o tempo, várias novas categorias também são adicionadas. Novas categorias podem ser categorias *virtuais*- aquelas que não possuem um subdiretório correspondente na árvore de ports - ou *físicas* - aquelas que possuem. Esta seção discute os problemas envolvidos na criação de uma nova categoria física. Leia atentamente antes de propor uma nova.

Nossa prática atual tem sido a de evitar a criação de uma nova categoria física, a menos que um grande número de ports logicamente pertençam a ela, ou os ports que pertenceriam a ela sejam um grupo logicamente distinto de interesse geral limitado (por exemplo, categorias relacionadas com as línguas humanas faladas), ou de preferência ambas.

A razão para isto é que tal mudança cria uma quantidade grande de trabalho tanto para os committers quanto para todos os usuários que rastreiam alterações na coleção de ports. Além disso, propostas de alteração de categorias parecem naturalmente atrair controvérsias. (Talvez isso seja porque não há um consenso claro sobre quando uma categoria é "grande o suficiente", nem quando as categorias devem ser apenas para propósitos de busca (e, portanto, qual número de categorias seria um número ideal), e assim por diante.)

Aqui está o procedimento:

1. Proponha a nova categoria na lista de discussão de ports do FreeBSD. Inclua uma justificativa detalhada para a nova categoria, incluindo por que as categorias existentes não são suficientes, e a lista de ports existentes propostos para a mudança. (Se houver novos ports pendentes no Bugzilla que caberia nessa categoria, liste-os também.) Se você for o mantenedor e/ou o apresentador, respectivamente, mencione isso, pois isso pode ajudar no caso.
2. Participe da discussão.
3. Se parecer que há apoio o suficiente para a ideia, registre um PR que inclua a lógica e a lista de ports existentes que precisam ser movidos. O ideal é que este PR também inclua essas alterações:
    - Makefiles para os novos ports, uma vez que sejam recopiados
    - Makefile para a nova categoria

    - Makefile para as categorias dos ports antigos
    - Makefiles para ports que dependem dos ports antigos
    - (para crédito extra, inclua os outros arquivos que precisam ser alterados, conforme o procedimento no Guia do Committer.)
4. Como isso afeta a infraestrutura do ports e envolve a movimentação e alteração de vários ports, pode ser necessário executar testes de regressão no cluster de build, e portanto, atribua o PR para a Equipe de Gerenciamento de Ports portmgr@FreeBSD.org.
5. Se esse PR for aprovado, um committer precisará seguir o restante do procedimento que é descrito no Guia do Committer.

A proposta de uma nova categoria virtual é semelhante à acima, mas muito menos trabalhoso, já que nenhum port terá que ser movido. Nesse caso, os únicos patches a serem incluídos no PR serão aqueles para adicionar a nova categoria na linha `CATEGORIES` dos ports afetados.

5.2.5. Propondo Reorganizar Todas as Categorias

Ocasionalmente alguém propõe reorganizar as categorias com uma estrutura de dois níveis, ou algum outro tipo de estrutura de palavras-chave. Até o momento, nada vem de nenhuma dessas propostas porque, embora sejam muito fáceis de fazer, o esforço envolvido com qualquer readequação de toda a coleção de ports existente é assustadora, para se dizer o mínimo. Por favor, leia o histórico dessas propostas nos arquivos da lista de discussão antes de postar essa idéia. Além disso, esteja preparado para ser desafiado a oferecer um protótipo funcional.

5.3. Os Arquivos de Distribuição

A segunda parte do Makefile descreve os arquivos que devem ser baixados para compilar o port e onde eles podem ser baixados.

5.3.1. `DISTNAME`

`DISTNAME` é o nome do port, conforme chamado pelos autores do software. `DISTNAME` é derivado de `${PORTNAME}-${DISTVERSIONPREFIX}${DISTVERSION}${DISTVERSIONSUFFIX}`, e se não estiver definido, `DISTVERSION` é derivado de `${PORTVERSION}`, portanto altere `DISTNAME` somente se necessário. `DISTNAME` é usado apenas em dois lugares. Primeiro, na lista de arquivos de distribuição (`DISTFILES`) padrão para `${DISTNAME} ${EXTRACT_SUFX}`. Em segundo lugar, espera-se que o arquivo de distribuição seja extraído em um subdiretório denominado `WRKSRC`, cujo padrão é work/${DISTNAME}.

Alguns nomes de distribuição de fornecedores que não se encaixam no `${PORTNAME}-${PORTVERSION}` -scheme podem ser tratados automaticamente configurando `DISTVERSIONPREFIX`, `DISTVERSION` e `DISTVERSIONSUFFIX`. `PORTVERSION` será derivado de `DISTVERSION` automaticamente.

> Apenas um dos `PORTVERSION` e `DISTVERSION` pode ser definido de cada vez. E se `DISTVERSION` não derivar um `PORTVERSION` correto, não use `DISTVERSION`.

Se o esquema de versão upstream puder ser derivado em um esquema de versão compatível com o ports, defina uma variável para a versão upstream, *não* use `DISTVERSION` como o nome da variável.

Defina PORTVERSION para a versão computada com base na variável criada, e defina DISTNAME adequadamente.

Se o esquema de versão upstream não puder ser facilmente configurado para um valor compatível com o ports, defina PORTVERSION para um valor sensato, e defina DISTNAME com PORTNAME com a versão literal do upstream.

Exemplo 11. Derivando PORTVERSION Manualmente

BIND9 usa um esquema de versão que não é compatível com as versões de ports (tem - em suas versões) e não pode ser derivado usando DISTVERSION porque após a versão 9.9.9, será lançado "patchlevels" na forma 9.9.9-P1. DISTVERSION iria traduzir isso para 9.9.9.p1, que no esquema de versionamento de ports significa 9.9.9 pré-release 1, que vem antes de 9.9.9 e não depois. Assim PORTVERSION é derivado manualmente de uma variável ISCVERSION para retornar 9.9.9p1.

A ordem na qual o framework do ports e o pkg ordenará as versões, é verificada usando o argumento -t do pkg-version(8):

```
% pkg version -t 9.9.9 9.9.9.p1
> ①
% pkg version -t 9.9.9 9.9.9p1
< ②
```

① O sinal > significa que o primeiro argumento passado em -t é maior que o segundo argumento. 9.9.9 é maior que 9.9.9.p1.

② O sinal < significa que o primeiro argumento passado em -t é menor que o segundo argumento. 9.9.9 é menor que 9.9.9p1.

No Makefile do port, por exemplo dns/bind99, é alcançado por:

```
PORTNAME=   bind
PORTVERSION=    ${ISCVERSION:S/-P/P/:S/b/.b/:S/a/.a/:S/rc/.rc/} ①
CATEGORIES= dns net
MASTER_SITES=   ISC/bind9/${ISCVERSION} ②
PKGNAMESUFFIX=  99
DISTNAME=   ${PORTNAME}-${ISCVERSION} ③

MAINTAINER= mat@FreeBSD.org
COMMENT=    BIND DNS suite with updated DNSSEC and DNS64

LICENSE=    ISCL

# ISC releases things like 9.8.0-P1 or 9.8.1rc1, which our versioning does not
like
ISCVERSION= 9.9.9-P6 ④
```

① Defina a versão upstream em ISCVERSION, com um comentário dizendo *porque* é necessário.

② Use ISCVERSION para obter um PORTVERSION compatível com o ports.

③ Use ISCVERSION diretamente para obter a URL correta para baixar o arquivo de distribuição.

④ Use ISCVERSION diretamente para nomear o arquivo de distribuição.

Exemplo 12. Derivar DISTNAME a partir de PORTVERSION

De tempos em tempos, o nome do arquivo de distribuição tem pouca ou nenhuma relação com a versão do software.

No comms/kermit, apenas o último elemento da versão está presente no arquivo de distribuição:

```
PORTNAME=   kermit
PORTVERSION=    9.0.304
CATEGORIES= comms ftp net
MASTER_SITES=   ftp://ftp.kermitproject.org/kermit/test/tar/
DISTNAME=   cku${PORTVERSION:E}-dev20 ①
```

① O modificador :E make(1) retorna o sufixo da variável, neste caso, 304. O arquivo de distribuição cku304-dev20.tar.gz é gerado corretamente.

Exemplo 13. Caso Exótico 1

Às vezes, não há relação entre o nome do software, sua versão e o arquivo de distribuição no qual ele é distribuído.

Do audio/libworkman:

```
PORTNAME=       libworkman
PORTVERSION=    1.4
CATEGORIES=     audio
MASTER_SITES=   LOCAL/jim
DISTNAME=       ${PORTNAME}-1999-06-20
```

Exemplo 14. Caso Exótico 2

No comms/librs232, o arquivo de distribuição não é versionado, portanto, DIST_SUBDIR é necessário:

```
PORTNAME=       librs232
PORTVERSION=    20160710
CATEGORIES=     comms
MASTER_SITES=   http://www.teuniz.net/RS-232/
DISTNAME=       RS-232
```

```
DIST_SUBDIR=    ${PORTNAME}-${PORTVERSION}
```

PKGNAMEPREFIX e PKGNAMESUFFIX não afetam o DISTNAME. Observe também que se WRKSRC for igual a ${WRKDIR}/${DISTNAME} enquanto o arquivo fonte original é nomeado para algo diferente de ${PORTNAME}-${PORTVERSION}${EXTRACT_SUFX}, deixe DISTNAME sozinho-- definir apenas DISTFILES é mais fácil que ambos DISTNAME e WRKSRC (e possivelmente EXTRACT_SUFX).

5.3.2. MASTER_SITES

Grave a parte do diretório do FTP/HTTP-URL apontando para o tarball original em MASTER_SITES. Não esqueça a barra final (/)!

A macro make irá tentar usar esta especificação para baixar o arquivo de distribuição com FETCH se não for possível encontrá-lo já no sistema.

Recomenda-se que vários sites sejam incluídos nesta lista, de preferência em diferentes continentes. Isso irá proteger contra problemas de rede amplos.

MASTER_SITES não deve estar em branco. Ele deve apontar para o site real que hospeda os arquivos de distribuição. Ele não pode apontar para web archives ou para os sites de cache dos arquivos de distribuição do FreeBSD. A única exceção a essa regra são ports que não possuem arquivos de distribuição. Por exemplo, meta-ports não possuem arquivos de distribuição, assim o MASTER_SITES não precisa ser definido.

Usando Variáveis MASTER_SITE_*

Abreviações de atalhos estão disponíveis para arquivos populares como o SourceForge (SOURCEFORGE), GNU (GNU), ou Perl CPAN (PERL_CPAN). MASTER_SITES pode usá-los diretamente:

```
MASTER_SITES=   GNU/make
```

O antigo formato expandido ainda funciona, mas todos os ports foram convertidos para o formato compacto. O formato expandido se parece com isto:

```
MASTER_SITES=       ${MASTER_SITE_GNU}
MASTER_SITE_SUBDIR= make
```

Estes valores e variáveis são definidos em Mk/bsd.sites.mk. Novas entradas são adicionadas com frequência, portanto, verifique a versão mais recente deste arquivo antes de enviar um port.

Para qualquer variável MASTER_SITE_FOO , a versão abreviada FOO pode ser utilizada. Por exemplo, use:

```
MASTER_SITES=	FOO
```

E se MASTER_SITE_SUBDIR for necessário, use isso:

```
MASTER_SITES=	FOO/bar
```

Alguns nomes MASTER_SITE_* são bastante longos e, para facilitar o uso, foram definidos atalhos:

*Tabela 3. Atalhos para Macros MASTER_SITE_**

Macro	Atalho
PERL_CPAN	CPAN
GITHUB	GH
GITHUB_CLOUD	GHC
LIBREOFFICE_DEV	LODEV
NETLIB	NL
RUBYGEMS	RG
SOURCEFORGE	SF

Macros Mágicas de MASTER_SITES

Várias macros "mágicas" existem para sites populares com uma estrutura de diretórios previsível. Para isso, basta usar a abreviação e o sistema escolherá um subdiretório automaticamente. Para um port nomeado Stardict, de versão 1.2.3 e hospedado no SourceForge, adicione esta linha:

```
MASTER_SITES=	SF
```

Implica em um subdiretório chamado /project/stardict/stardict/1.2.3. Se o diretório estiver incorreto, ele poderá ser substituído:

```
MASTER_SITES=	SF/stardict/WyabdcRealPeopleTTS/${PORTVERSION}
```

Isso também pode ser escrito como

```
MASTER_SITES=	SF
MASTER_SITE_SUBDIR= stardict/WyabdcRealPeopleTTS/${PORTVERSION}
```

Tabela 4. Macros Mágicas de MASTER_SITES

Macro	Subdiretório deduzido
APACHE_COMMONS_BINARIES	${PORTNAME:S,commons-,,}

Macro	Subdiretório deduzido
APACHE_COMMONS_SOURCE	${PORTNAME:S,commons-,,}
APACHE_JAKARTA	${PORTNAME:S,-,/,}/source
BERLIOS	${PORTNAME:tl}.berlios
CHEESESHOP	source/${DISTNAME:C/(.).*/\1/}/${DISTNAME:C/(.*)-[0-9].*/\1/}
CPAN	${PORTNAME:C/-.*//}
DEBIAN	pool/main/${PORTNAME:C/^((lib)?.).*$/\1/}/${PORTNAME}
FARSIGHT	${PORTNAME}
FESTIVAL	${PORTREVISION}
GCC	releases/${DISTNAME}
GENTOO	distfiles
GIMP	${PORTNAME}/${PORTVERSION:R}/
GH	${GH_ACCOUNT}/${GH_PROJECT}/tar.gz/${GH_TAGNAME}?dummy=/
GHC	${GH_ACCOUNT}/${GH_PROJECT}/
GNOME	sources/${PORTNAME}/${PORTVERSION:C/^(\.[0-9]).*/\1/}
GNU	${PORTNAME}
GNUPG	${PORTNAME}
GNU_ALPHA	${PORTNAME}
HORDE	${PORTNAME}
LODEV	${PORTNAME}
MATE	${PORTVERSION:C/^(\.[0-9]).*/\1/}
MOZDEV	${PORTNAME:tl}
NL	${PORTNAME}
QT	archive/qt/${PORTVERSION:R}
SAMBA	${PORTNAME}
SAVANNAH	${PORTNAME:tl}
SF	${PORTNAME:tl}/${PORTNAME:tl}/${PORTVERSION}

5.3.3. USE_GITHUB

Se o arquivo de distribuição vier de um commit ou tag específico no GitHub para o qual não há arquivo lançado oficialmente, há uma maneira fácil de definir o DISTNAME e MASTER_SITES corretos automaticamente. Estas variáveis estão disponíveis:

Tabela 5. USE_GITHUB *Descrição*

Variável	Descrição	Padrão
GH_ACCOUNT	Nome da conta do usuário do GitHub que hospeda o projeto	${PORTNAME}

Variável	Descrição	Padrão
GH_PROJECT	Nome do projeto no GitHub	${PORTNAME}
GH_TAGNAME	Nome da tag para download (2.0.1, hash, ...) Usar o nome de uma branch aqui é errado. Também é possível usar o hash de um ID de commit para gerar um snapshot.	${DISTVERSIONPREFIX}${DISTVERSION}${DISTVERSIONSUFFIX}
GH_SUBDIR	Quando o software precisa que um arquivo de distribuição adicional seja extraído em ${WRKSRC}, esta variável pode ser usada. Veja os exemplos em Baixando Múltiplos Arquivos do GitHub para maiores informações.	(none)
GH_TUPLE	GH_TUPLE permite colocar GH_ACCOUNT, GH_PROJECT, GH_TAGNAME e GH_SUBDIR em uma única variável. O formato é *conta*`:`*projeto*`:`*tagname*`:`*grupo*`/`*subdiretório*. O `/`_subdiretório _ é opcional. Isso é útil quando mais de um projeto no GitHub precisa ser utilizado.	

Não use GH_TUPLE para o arquivo de distribuição padrão, já que não tem nenhum padrão.

Exemplo 15. Uso Simples de USE_GITHUB

Ao tentar fazer um port para a versão 1.2.7 do pkg do usuário FreeBSD no github, em https://github.com/freebsd/pkg, O Makefile acabaria ficando assim (levemente simplificado para o exemplo):

```
PORTNAME=   pkg
DISTVERSION=    1.2.7

USE_GITHUB= yes
GH_ACCOUNT= freebsd
```

MASTER_SITES será automaticamente definido como GH GHC e WRKSRC para ${WRKDIR}/pkg-1.2.7.

Exemplo 16. Uso Mais Completo de `USE_GITHUB`

Ao tentar fazer um port para uma versão de desenvolvimento do pkg do usuário FreeBSD no github, em https://github.com/freebsd/pkg, o Makefile acaba ficando assim (levemente simplificado para o exemplo):

```
PORTNAME=   pkg-devel
DISTVERSION=    1.3.0.a.20140411

USE_GITHUB= yes
GH_ACCOUNT= freebsd
GH_PROJECT= pkg
GH_TAGNAME= 6dbb17b
```

`MASTER_SITES` será automaticamente definido para `GH GHC` e `WRKSRC` para `${WRKDIR}/pkg-6dbb17b`.

`20140411` é a data do commit referenciada em `GH_TAGNAME`, não a data em que é editado o Makefile, ou a data em que o commit é feito.

Exemplo 17. Uso de `USE_GITHUB` *com* `DISTVERSIONPREFIX`

De tempos em tempos, `GH_TAGNAME` é uma ligeira variação de `DISTVERSION`. Por exemplo, se a versão for `1.0.2`, e a tag `v1.0.2`. Nesses casos, é possível usar `DISTVERSIONPREFIX` ou `DISTVERSIONSUFFIX`:

```
PORTNAME=   foo
DISTVERSIONPREFIX=  v
DISTVERSION=    1.0.2

USE_GITHUB= yes
```

`GH_TAGNAME` será automaticamente definido para `v1.0.2`, enquanto `WRKSRC` será mantido como `${WRKDIR} /foo-1.0.2`.

Exemplo 18. Usando `USE_GITHUB` *Quando o Upstream Não Usa Versões*

Se nunca houve uma versão upstream, não invente uma como `0.1` ou `1.0`. Crie o port com um `DISTVERSION` de `gYYYYMMDD`, onde `g` é para Git e `YYYYMMDD` representa a data em que o commit é referenciado em `GH_TAGNAME`.

```
PORTNAME=   bar
DISTVERSION=    g20140411

USE_GITHUB= yes
GH_TAGNAME= c472d66b
```

Isso cria um esquema de controle de versão que é incrementado com o tempo e que ainda é menor do que a versão `0` (veja Usando pkg-version(8) para comparar versões. para mais informações do pkg-version(8)):

```
% pkg version -t g20140411 0
<
```

Isso significa que não será necessário usar o `PORTEPOCH` caso o upstream decida lançar versões no futuro.

Exemplo 19. Usando `USE_GITHUB` para Acessar um Commit Entre Duas Versões

Se a versão atual do software usa uma tag Git, e o port precisa ser atualizado para uma versão mais recente e intermediária, sem uma tag, use git-describe(1) para descobrir a versão a ser utilizada:

```
% git describe --tags f0038b1
v0.7.3-14-gf0038b1
```

`v0.7.3-14-gf0038b1` pode ser dividido em três partes:

`v0.7.3`
: Este é a última tag Git que aparece no histórico de commits antes do commit solicitado.

`-14`
: Isso significa que o commit solicitado, `f0038b1`, é o 14º commit após a tag `v0.7.3`.

`-gf0038b1`
: O `-g` significa "Git", e o `f0038b1` é o commit hash referenciado.

```
PORTNAME=   bar
DISTVERSIONPREFIX=  v
DISTVERSION=    0.7.3-14
DISTVERSIONSUFFIX=  -gf0038b1

USE_GITHUB= yes
```

Isso cria um esquema de versionamento que é incrementado com o tempo (bem, em cima de commits), e não entra em conflito com a criação de uma versão `0.7.4`. (Veja Usando pkg-version(8) para comparar versões. para detalhes do pkg-version(8)):

```
% pkg version -t 0.7.3 0.7.3.14
<
% pkg version -t 0.7.3.14 0.7.4
```

```
<
```

Se o commit solicitado é o mesmo que uma tag, uma descrição mais curta é mostrada por padrão. A versão mais longa é equivalente:

```
% git describe --tags c66c71d
v0.7.3
% git describe --tags --long c66c71d
v0.7.3-0-gc66c71d
```

Baixando Múltiplos Arquivos do GitHub

O framework `USE_GITHUB` também suporta a obtenção de vários arquivos de distribuição de diferentes locais no GitHub. Ele funciona de uma forma muito semelhante ao Múltiplos Arquivos de Distribuição ou Patches de Vários Locais.

Vários valores são adicionados a `GH_ACCOUNT`, `GH_PROJECT` e `GH_TAGNAME`. Cada valor diferente é atribuído a um grupo. O valor principal pode não ter nenhum grupo ou grupo `:DEFAULT`. Um valor pode ser omitido se for o mesmo que o padrão listado em `USE_GITHUB` Descrição.

`GH_TUPLE` também pode ser usado quando há muitos arquivos de distribuição. Isso ajuda a manter as informações de conta, projeto, tagname e grupo no mesmo lugar.

Para cada grupo, uma variável auxiliar `${WRKSRC_group}` é criada, contendo o diretório no qual o arquivo foi extraído. As variáveis `${WRKSRC_group}` podem ser usadas para mover diretórios durante o `post-extract`, ou para serem adicionadas em `CONFIGURE_ARGS`, ou o que for necessário para que o software seja compilado corretamente.

A parte do `:group` *deve* ser usada para *apenas um* arquivo de distribuição. Ela é usado como uma chave única e usá-la mais de uma vez irá sobrescrever os valores anteriores.

Como isso é apenas modificações de `DISTFILES` e `MASTER_SITES`, os nomes dos grupos devem obedecer às restrições de nomes de grupos descritas em Múltiplos Arquivos de Distribuição ou Patches de Vários Locais

Ao buscar vários arquivos do GitHub, às vezes o arquivo de distribuição padrão não é buscado no GitHub. Para desabilitar a busca da distribuição padrão, defina:

```
USE_GITHUB= nodefault
```

Ao utilizar `USE_GITHUB=nodefault`, o Makefile deve ter `DISTFILES` em seu bloco inicial. A definição deve ser:

```
DISTFILES=    ${DISTNAME}${EXTRACT_SUFX}
```

Exemplo 20. Uso de USE_GITHUB com Vários Arquivos de Distribuição

De tempos em tempos é necessário baixar mais de um arquivo de distribuição. Por exemplo, quando o repositório git do upstream usa submódulos. Isso pode ser feito facilmente usando grupos nas variáveis GH_*:

```
PORTNAME=   foo
DISTVERSION=    1.0.2

USE_GITHUB= yes
GH_ACCOUNT= bar:icons,contrib
GH_PROJECT= foo-icons:icons foo-contrib:contrib
GH_TAGNAME= 1.0:icons fa579bc:contrib
GH_SUBDIR=  ext/icons:icons

CONFIGURE_ARGS= --with-contrib=${WRKSRC_contrib}
```

Isso irá baixar três arquivos de distribuição do github. O padrão vem de foo/foo versão 1.0.2. O segundo, com o grupo icons, vem de bar/foo-icons versão 1.0. O terceiro vem de bar/foo-contrib e usa o commit do Git fa579bc. Os arquivos de distribuição são nomeados foo-foo-1.0.2_GH0.tar.gz, bar-foo-icons-1.0_GH0.tar.gz e bar-foo-contrib-fa579bc_GH0.tar.gz.

Todos os arquivos de distribuição são extraídos em ${WRKDIR} em seus respectivos subdiretórios. O arquivo padrão ainda é extraído em ${WRKSRC}, nesse caso, ${WRKDIR}/foo-1.0.2. Cada arquivo de distribuição adicional é extraído em ${WRKSRC_group}. Aqui, para o grupo icons, chamado de ${WRKSRC_icons}, será ${WRKDIR}/foo-icons-1.0. O arquivo com o grupo contrib é chamado de ${WRKSRC_contrib} e contém ${WRKDIR}/foo-contrib-fa579bc.

O sistema de compilação do software espera encontrar os ícones em um subdiretório ext/icons em seus fontes, então GH_SUBDIR é usado. GH_SUBDIR garante que ext exista, mas não que ext/icons também exista. Então isso acontece:

```
post-extract:
      @${MV} ${WRKSRC_icons} ${WRKSRC}/ext/icons
```

Exemplo 21. Uso de USE_GITHUB com Vários Arquivos de Distribuição Usando GH_TUPLE

Isto é funcionalmente equivalente a Uso de USE_GITHUB com Vários Arquivos de Distribuição mas usando GH_TUPLE:

```
PORTNAME=   foo
DISTVERSION=    1.0.2
```

```
USE_GITHUB= yes
GH_TUPLE=   bar:foo-icons:1.0:icons/ext/icons \
        bar:foo-contrib:fa579bc:contrib

CONFIGURE_ARGS= --with-contrib=${WRKSRC_contrib}
```

Agrupamento foi usado no exemplo anterior com `bar:icons, contrib`. Algumas informações redundantes estão presentes com `GH_TUPLE` porque o uso de agrupamento não é possível.

Exemplo 22. Como Usar `USE_GITHUB` com Submodulos Git?

Ports com o GitHub como um repositório upstream às vezes usam submódulos. Veja git-submodule(1) para maiores informações.

O problema com submódulos é que cada um é um repositório separado. Como tal, cada um deve ser buscado separadamente.

Usando finances/moneymanagerex como exemplo, seu repositório GitHub é https://github.com/moneymanagerex/moneymanagerex. Tem um arquivo .gitmodules na raiz. Este arquivo descreve todos os sub módulos usados neste repositório e lista os repositórios adicionais necessários. Este arquivo irá dizer quais repositórios adicionais são necessários:

```
[submodule "lib/wxsqlite3"]
    path = lib/wxsqlite3
    url = https://github.com/utelle/wxsqlite3.git
[submodule "3rd/mongoose"]
    path = 3rd/mongoose
    url = https://github.com/cesanta/mongoose.git
[submodule "3rd/LuaGlue"]
    path = 3rd/LuaGlue
    url = https://github.com/moneymanagerex/LuaGlue.git
[submodule "3rd/cgitemplate"]
    path = 3rd/cgitemplate
    url = https://github.com/moneymanagerex/html-template.git
[...]
```

A única informação que falta nesse arquivo é a hash ou tag de commit para usar na versão. Esta informação é encontrada após a clonagem do repositório:

```
% git clone --recurse-submodules
https://github.com/moneymanagerex/moneymanagerex.git
Cloning into 'moneymanagerex'...
remote: Counting objects: 32387, done.
[...]
Submodule '3rd/LuaGlue' (https://github.com/moneymanagerex/LuaGlue.git) registered
for path '3rd/LuaGlue'
Submodule '3rd/cgitemplate' (https://github.com/moneymanagerex/html-template.git)
registered for path '3rd/cgitemplate'
```

```
Submodule '3rd/mongoose' (https://github.com/cesanta/mongoose.git) registered for
path '3rd/mongoose'
Submodule 'lib/wxsqlite3' (https://github.com/utelle/wxsqlite3.git) registered for
path 'lib/wxsqlite3'
[...]
Cloning into
'/home/mat/work/freebsd/ports/finance/moneymanagerex/moneymanagerex/3rd/LuaGlue'..
.
Cloning into
'/home/mat/work/freebsd/ports/finance/moneymanagerex/moneymanagerex/3rd/cgitemplat
e'...
Cloning into
'/home/mat/work/freebsd/ports/finance/moneymanagerex/moneymanagerex/3rd/mongoose'.
..
Cloning into
'/home/mat/work/freebsd/ports/finance/moneymanagerex/moneymanagerex/lib/wxsqlite3'
...
[...]
Submodule path '3rd/LuaGlue': checked out
'c51d11a247ee4d1e9817dfa2a8da8d9e2f97ae3b'
Submodule path '3rd/cgitemplate': checked out
'cd434eeeb35904ebcd3d718ba29c281a649b192c'
Submodule path '3rd/mongoose': checked out
'2140e5992ab9a3a9a34ce9a281abf57f00f95cda'
Submodule path 'lib/wxsqlite3': checked out
'fb66eb230d8aed21dec273b38c7c054dcb7d6b51'
[...]
% cd moneymanagerex
% git submodule status
 c51d11a247ee4d1e9817dfa2a8da8d9e2f97ae3b 3rd/LuaGlue (heads/master)
 cd434eeeb35904ebcd3d718ba29c281a649b192c 3rd/cgitemplate (cd434ee)
 2140e5992ab9a3a9a34ce9a281abf57f00f95cda 3rd/mongoose (6.2-138-g2140e59)
 fb66eb230d8aed21dec273b38c7c054dcb7d6b51 lib/wxsqlite3 (v3.4.0)
[...]
```

Também pode ser encontrado no GitHub. Cada subdiretório que é um submódulo é mostrado como *diretório* `@` *hash*, por exemplo,`mongoose @ 2140e59`.

Embora a obtenção das informações pelo GitHub pareça mais fácil, as informações encontradas usando `git submodule status` fornecerá informações mais significativas. Por exemplo, o commit hash de `lib/wxsqlite3 fb66eb2` corresponde a `v3.4.0`. Ambos podem ser usados, mas quando uma tag estiver disponível, use-a.

Agora que todas as informações necessárias foram reunidas, o Makefile pode ser escrito (somente as linhas relacionadas ao GitHub são mostradas):

```
PORTNAME=   moneymanagerex
DISTVERSIONPREFIX=  v
```

```
DISTVERSION=    1.3.0

USE_GITHUB= yes
GH_TUPLE=   utelle:wxsqlite3:v3.4.0:wxsqlite3/lib/wxsqlite3 \
        moneymanagerex:LuaGlue:c51d11a:lua_glue/3rd/LuaGlue \
        moneymanagerex:html-template:cd434ee:html_template/3rd/cgitemplate \
        cesanta:mongoose:2140e59:mongoose/3rd/mongoose \
        [...]
```

5.3.4. USE_GITLAB

Semelhante ao GitHub, se o arquivo de distribuição vier de gitlab.com ou se estiver hospedado com o software GitLab, essas variáveis estão disponíveis para uso e talvez precisem ser definidas.

Tabela 6. `USE_GITLAB` *Descrição*

Variável	Descrição	Padrão
`GL_SITE`	Nome do site que hospeda o projeto GitLab	https://gitlab.com
`GL_ACCOUNT`	Nome da conta do usuário do GitLab hospedando o projeto	`${PORTNAME}`
`GL_PROJECT`	Nome do projeto em GitLab	`${PORTNAME}`
`GL_COMMIT`	O hash de commit para download. Deve ser o hash hex sha1 completo de 160 bits e 40 caracteres. Essa é uma variável obrigatória para GitLab.	`(none)`
`GL_SUBDIR`	Quando o software precisa de um arquivo de distribuição adicional para ser extraído com `${WRKSRC}`, esta variável pode ser usada. Veja os exemplos em Baixando Múltiplos Arquivos do GitLab para maiores informações.	(none)

Variável	Descrição	Padrão
GL_TUPLE	GL_TUPLE permite colocar GL_SITE, GL_ACCOUNT, GL_PROJECT, GL_COMMIT, e GL_SUBDIR dentro de uma única variável. O formato é *site*`:`*conta*`:`*projeto*`:`*commit*`:`*grupo*`/subdiretório. O site:` e `/`*subdiretório* são opcionais. Isso ajuda quando é necessário baixar arquivos de mais de um projeto GitLab.	

Exemplo 23. Uso Simples de USE_GITLAB

Ao tentar fazer um port para a versão 1.14 do libsignon-glib do usuário accounts-sso do gitlab.com, em https://gitlab.com/accounts-sso/libsignon-glib, O Makefile acabaria ficando assim para buscar os arquivos de distribuição:

```
PORTNAME=   libsignon-glib
DISTVERSION=    1.14

USE_GITLAB= yes
GL_ACCOUNT= accounts-sso
GL_COMMIT=  e90302e342bfd27bc8c9132ab9d0ea3d8723fd03
```

Ele terá automaticamente MASTER_SITES definido como gitlab.com e WRKSRC para ${WRKDIR}/libsignon-glib-e90302e342bfd27bc8c9132ab9d0ea3d8723fd03-e90302e342bfd27bc8c9132ab9d0ea3d8723fd03.

Exemplo 24. Uso Mais Completo de USE_GITLAB

Um uso mais completo do exemplo acima é se o port não tiver controle de versão e foobar do usuário foo no projeto bar em um GitLab auto hospedado em https://gitlab.example.com, o Makefile acaba ficando assim para buscar os arquivos de distribuição:

```
PORTNAME=   foobar
DISTVERSION=    g20170906

USE_GITLAB= yes
GL_SITE=    https://gitlab.example.com
GL_ACCOUNT= foo
GL_PROJECT= bar
GL_COMMIT=  9c1669ce60c3f4f5eb43df874d7314483fb3f8a6
```

Terá MASTER_SITES definido como "https://gitlab.example.com" e WRKSRC para ${WRKDIR}/bar-

9c1669ce60c3f4f5eb43df874d7314483fb3f8a6-9c1669ce60c3f4f5eb43df874d7314483fb3f8a6.

20170906 é a data do commit referenciada em GL_COMMIT, não a data em que o Makefile é editado, ou a data em que o commit para a árvore de ports do FreeBSD é feito.

O protocolo, porta e webroot do GL_SITE podem ser modificados na mesma variável.

Baixando Múltiplos Arquivos do GitLab

O framework USE_GITLAB também suporta a busca de vários arquivos de distribuição de diferentes locais de GitLab e sites hospedados no GitLab. Ele funciona de uma forma muito semelhante ao Múltiplos Arquivos de Distribuição ou Patches de Vários Locais e Baixando Múltiplos Arquivos do GitLab.

Vários valores são adicionados a GL_SITE, GL_ACCOUNT, GL_PROJECT e GL_COMMIT. Cada valor diferente é atribuído a um grupo. USE_GITLAB Descrição.

GL_TUPLE também pode ser usado quando há muitos arquivos de distribuição. Isso ajuda a manter as informações de site, conta, projeto, commit e grupo no mesmo local.

Para cada grupo, uma variável auxiliar ${WRKSRC_group} é criada, contendo o diretório no qual o arquivo foi extraído. As variáveis ${WRKSRC_group} podem ser usadas para mover diretórios durante o post-extract, ou para serem adicionadas em CONFIGURE_ARGS, ou o que for necessário para que o software seja compilado corretamente.

A parte do :group *deve* ser usada para *apenas um* arquivo de distribuição. Ela é usado como uma chave única e usá-la mais de uma vez irá sobrescrever os valores anteriores.

Como isso é apenas modificações de DISTFILES e MASTER_SITES, os nomes dos grupos devem obedecer às restrições de nomes de grupos descritas em Múltiplos Arquivos de Distribuição ou Patches de Vários Locais

Ao buscar vários arquivos usando GitLab, às vezes, o arquivo de distribuição padrão não é obtido de um GitLab. Para desativar a busca do arquivo de distribuição padrão, defina:

```
USE_GITLAB= nodefault
```

Ao utilizar USE_GITLAB=nodefault, o Makefile deve ter DISTFILES em seu bloco inicial. A definição deve ser:

```
DISTFILES=    ${DISTNAME}${EXTRACT_SUFX}
```

Exemplo 25. Uso de USE_GITLAB com Vários Arquivos de Distribuição

De tempos em tempos, é necessário buscar mais de um arquivo de distribuição. Por exemplo, quando o repositório git do upstream usa submódulos. Isso pode ser feito facilmente usando grupos nas variáveis GL_*:

```
PORTNAME=   foo
DISTVERSION=    1.0.2

USE_GITLAB= yes
GL_SITE=    https://gitlab.example.com:9434/gitlab:icons
GL_ACCOUNT= bar:icons,contrib
GL_PROJECT= foo-icons:icons foo-contrib:contrib
GL_COMMIT=  c189207a55da45305c884fe2b50e086fcad4724b
ae7368cab1ca7ca754b38d49da064df87968ffe4:icons
9e4dd76ad9b38f33fdb417a4c01935958d5acd2a:contrib
GL_SUBDIR=  ext/icons:icons

CONFIGURE_ARGS= --with-contrib=${WRKSRC_contrib}
```

Isso irá buscar dois arquivos de distribuição do gitlab.com e um de gitlab.example.com hospedado com GitLab. O padrão vem de https://gitlab.com/foo/foo e o commit é c189207a55da45305c884fe2b50e086fcad4724b. O segundo, com o grupo icons, vem de https://gitlab.example.com:9434/gitlab/bar/foo-icons e o commit é ae7368cab1ca7ca754b38d49da064df87968ffe4. O terceiro vem de https://gitlab.com/bar/foo-contrib e o commit é 9e4dd76ad9b38f33fdb417a4c01935958d5acd2a. Os arquivos de distribuição são nomeados foo-foo-c189207a55da45305c884fe2b50e086fcad4724b_GL0.tar.gz, bar-foo-icons-ae7368cab1ca7ca754b38d49da064df87968ffe4_GL0.tar.gz e bar-foo-contrib-9e4dd76ad9b38f33fdb417a4c01935958d5acd2a_GL0.tar.gz.

Todos os arquivos de distribuição são extraídos no ${WRKDIR} em seus respectivos subdiretórios. O arquivo padrão ainda é extraído no ${WRKSRC}, nesse caso, ${WRKDIR}/foo-c189207a55da45305c884fe2b50e086fcad4724b-c189207a55da45305c884fe2b50e086fcad4724b. Cada arquivo de distribuição adicional é extraído em ${WRKSRC_group}. Aqui, para o grupo icons, é chamado ${WRKSRC_icons} e contém ${WRKDIR}/foo-icons-ae7368cab1ca7ca754b38d49da064df87968ffe4-ae7368cab1ca7ca754b38d49da064df87968ffe4. O arquivo com o grupo contrib é chamado ${WRKSRC_contrib} e contém ${WRKDIR}/foo-contrib-9e4dd76ad9b38f33fdb417a4c01935958d5acd2a-9e4dd76ad9b38f33fdb417a4c01935958d5acd2a.

O sistema de compilação do software espera encontrar os ícones em um subdiretório ext/icons em seus fontes, então GL_SUBDIR é usado.GL_SUBDIR garante que ext existe, mas não que ext/icons também exista. Então isso acontece:

```
post-extract:
        @${MV} ${WRKSRC_icons} ${WRKSRC}/ext/icons
```

Exemplo 26. Uso de USE_GITLAB com Vários Arquivos de Distribuição Usando GL_TUPLE

Isto é funcionalmente equivalente a Uso de USE_GITLAB com Vários Arquivos de Distribuição mas usando GL_TUPLE:

```
PORTNAME=   foo
DISTVERSION=    1.0.2

USE_GITLAB= yes
GL_COMMIT=  c189207a55da45305c884fe2b50e086fcad4724b
GL_TUPLE=   https://gitlab.example.com:9434/gitlab:bar:foo-
icons:ae7368cab1ca7ca754b38d49da064df87968ffe4:icons/ext/icons \
        bar:foo-contrib:9e4dd76ad9b38f33fdb417a4c01935958d5acd2a:contrib

CONFIGURE_ARGS= --with-contrib=${WRKSRC_contrib}
```

Agrupamento foi usado no exemplo anterior com bar:icons,contrib. Algumas informações redundantes estão presentes com GL_TUPLE porque o uso de agrupamento não é possível.

5.3.5. EXTRACT_SUFX

Se houver um arquivo de distribuição e ele usar um sufixo diferente para indicar o mecanismo de compactação, defina EXTRACT_SUFX.

Por exemplo, se o arquivo de distribuição foi nomeado foo.tar.gzip em vez do mais comum foo.tar.gz, escreva:

```
DISTNAME=   foo
EXTRACT_SUFX=   .tar.gzip
```

O USES=tar[:xxx], USES=lha ou USES=zip define automaticamente EXTRACT_SUFX com as extensões de arquivo mais comuns, conforme necessário, consulte Usando Macros USES para mais detalhes. Se nenhum destes estiver definido, o EXTRACT_SUFX padrão é .tar.gz.

Como EXTRACT_SUFX é usado apenas em DISTFILES, apenas defina um deles..

5.3.6. DISTFILES

Às vezes os nomes dos arquivos a serem baixados não têm semelhança com o nome do port. Por exemplo, pode ser chamado source.tar.gz ou similar. Em outros casos, o código-fonte do aplicativo pode estar em vários arquivos diferentes, e todos eles devem ser baixados.

Se este for o caso, defina DISTFILES para ser uma lista separada por espaços de todos os arquivos que devem ser baixados.

```
DISTFILES=  source1.tar.gz source2.tar.gz
```

Se não for definido explicitamente, o DISTFILES padrão é ${DISTNAME}${EXTRACT_SUFX}.

5.3.7. EXTRACT_ONLY

Se apenas alguns dos DISTFILES devem ser extraídos-- por exemplo, um deles é o código-fonte, enquanto outro é um documento não compactado - liste os nomes dos arquivos que devem ser extraídos em EXTRACT_ONLY.

```
DISTFILES=  source.tar.gz manual.html
EXTRACT_ONLY=   source.tar.gz
```

Quando nenhum dos DISTFILES precisam ser descompactados, deixe vazio o EXTRACT_ONLY.

```
EXTRACT_ONLY=
```

5.3.8. PATCHFILES

Se o port requer alguns patches adicionais que estão disponíveis por FTP ou HTTP, defina PATCHFILES para os nomes dos arquivos e PATCH_SITES para a URL do diretório que os contém (o formato é o mesmo do MASTER_SITES).

Se o patch não for relativo ao inicio da árvore do código fonte (isto é, WRKSRC) porque contém alguns pathnames extras, defina PATCH_DIST_STRIP adequadamente. Por exemplo, se todos os pathnames no patch tiverem um `foozolix-1.0 /` extra na frente dos nomes dos arquivos, então defina PATCH_DIST_STRIP=-p1.

Não se preocupe se os patches estiverem compactados; eles serão descompactados automaticamente se os nomes dos arquivos terminarem com .Z, .gz, .bz2 ou .xz.

Se o patch for distribuído com alguns outros arquivos, como documentação, em um arquivo compactado, o uso de PATCHFILES não será possível. Se for esse o caso, adicione o nome e a localização do arquivo do patch em DISTFILES e MASTER_SITES. Então, use EXTRA_PATCHES para apontar para esses arquivos e o bsd.port.mk irá aplicá-los automaticamente. Em particular, *não* copie os arquivos de patch em ${PATCHDIR}. Esse diretório pode não ter permissão de escrita.

Se houver vários patches e eles precisarem de valores mistos para o parâmetro strip, ele poderá ser adicionado ao lado do nome do patch em PATCHFILES, por exemplo:

```
PATCHFILES= patch1 patch2:-p1
```

Isto não entra em conflito com o recurso de agrupamento de sites principais, adicionando um grupo também funciona:

```
PATCHFILES= patch2:-p1:source2
```

O arquivo será extraído junto com o arquivo de código fonte, então não há necessidade de explicitamente extraí-lo se ele for um arquivo compactado normal. Tome cuidado extra para não sobrescrever algo que já existe nesse diretório caso faça a extração manualmente. Também não se esqueça de adicionar um comando para remover o patch copiado no target `pre-clean`.

5.3.9. Múltiplos Arquivos de Distribuição ou Patches de Vários Locais

(Considere isto como um "tópico avançado"; a princípio, aqueles que são novos neste documento podem desejar pular esta seção).

Esta seção contém informações sobre o mecanismo de busca conhecido como `MASTER_SITES:n` e `MASTER_SITES_NN`. Vamos nos referir a este mecanismo como `MASTER_SITES:n`.

Um pouco de background primeiro. O OpenBSD tem um ótimo recurso dentro do `DISTFILES` e `PATCHFILES` que permite que arquivos e pacthes sejam pós fixados com identificadores `:n`. Aqui, `n` pode ser qualquer palavra que contenha `[0-9a-zA-Z_]` e signifique uma designação de grupo. Por exemplo:

```
DISTFILES=  alpha:0 beta:1
```

No OpenBSD, arquivo de distribuição alpha será associado com a variável `MASTER_SITES0` em vez da nossa comum `MASTER_SITES` e beta com `MASTER_SITES1`.

Esta é uma característica muito interessante que pode diminuir a busca sem fim pelo site de download correto.

Apenas imagine 2 arquivos em `DISTFILES` e 20 sites em `MASTER_SITES`, os sites são extremamente lentos e beta é hospedado em todas as entradas do `MASTER_SITES` e alfa só pode ser encontrado no 20º site. Seria um desperdício checar todos eles se o mantenedor soubesse isso de antemão, não seria? Não é um bom começo para aquele lindo fim de semana!

Agora que você já tem uma ideia, imagine mais `DISTFILES` e mais `MASTER_SITES`. Certamente nosso "distfiles survey meister" irá ser apreciado pelo alívio nas conexões de rede que isso trará.

Nas próximas seções, as informações seguirão a implementação do FreeBSD desta idéia. Nós melhoramos um pouco o conceito do OpenBSD.

Os nomes dos grupos não podem ter traços neles (`-`), na verdade, eles não podem ter nenhum caractere fora do range `[a-zA-Z0-9_]`. Isso porque, enquantomake(1) está ok com nomes de variáveis contendo traços, sh(1)não.

Informação Simplificada

Esta seção explica como preparar rapidamente a busca de vários arquivos de distribuição e patches de diferentes sites e subdiretórios. Descrevemos aqui um caso de uso de `MASTER_SITES:n`. Isso será suficiente para a maioria dos cenários. Informações mais detalhadas estão disponíveis em Informação Detalhada.

Alguns aplicativos consistem em vários arquivos de distribuição que devem ser baixados de vários sites diferentes. Por exemplo, Ghostscript consiste no núcleo do programa e, em seguida, um grande número de arquivos de driver que são usados dependendo da impressora do usuário. Alguns desses arquivos de driver são fornecidos com o núcleo, mas muitos outros devem ser baixados de uma variedade de sites diferentes.

Para suportar isso, cada entrada no `DISTFILES` pode ser seguida por dois pontos e um "nome de grupo". Cada site listado em `MASTER_SITES` é então seguido por dois pontos, e o grupo que indica quais arquivos de distribuição são baixados deste site.

Por exemplo, considere um aplicativo com a divisão do código fonte em duas partes, source1.tar.gz e source2.tar.gz, que deve ser baixado de dois sites diferentes. O Makefile do port incluiria linhas como Uso Simplificado de `MASTER_SITES:n` com Um Arquivo Por Site.

Exemplo 27. Uso Simplificado de `MASTER_SITES:n` com Um Arquivo Por Site

```
MASTER_SITES=   ftp://ftp1.example.com/:source1 \
        http://www.example.com/:source2
DISTFILES=  source1.tar.gz:source1 \
        source2.tar.gz:source2
```

Vários arquivos de distribuição podem ter o mesmo grupo. Continuando o exemplo anterior, suponha que houvesse um terceiro distfile, source3.tar.gz, que é baixado do `ftp.example2.com`. O Makefile seria então escrito como Uso Simplificado de `MASTER_SITES:n` com Mais de Um Arquivo Por Site.

Exemplo 28. Uso Simplificado de `MASTER_SITES:n` com Mais de Um Arquivo Por Site

```
MASTER_SITES=   ftp://ftp.example.com/:source1 \
        http://www.example.com/:source2
DISTFILES=  source1.tar.gz:source1 \
        source2.tar.gz:source2 \
        source3.tar.gz:source2
```

Informação Detalhada

Ok, então o exemplo anterior não refletiu as necessidades do novo port? Nesta seção vamos explicar em detalhes como o mecanismo de busca avançado `MASTER_SITES:n` funciona e como ele pode ser usado.

1. Elementos podem ser pós-fixados com `:n` onde *n* é `, isso é, _n_ poderia conceitualmente ser qualquer string alfanumérica, mas vamos limitá-lo a `[a-zA-Z_][0-9a-zA-Z_]` por enquanto.

   Além disso, a verificação de strings é case sensitive, ou seja, `n` é diferente de `N`.

   No entanto, essas palavras não podem ser usadas para finalidades de pós-fixação, pois elas produzem um significado especial: `default`, `all` e `ALL` (estes são usados internamente no item ii). Além disso, `DEFAULT` é uma palavra de propósito especial (verifique o item 3).

2. Elementos pós-fixados com `:n` pertence ao grupo `n`, `:m` pertence ao grupo `m` e assim por diante.
3. Elementos que não estão pós-fixados são desagrupados, todos eles pertencem ao grupo especial `DEFAULT`. Quaisquer elementos pós-fixados com `DEFAULT` estão apenas sendo redundantes, a menos que um elemento pertença a ambos `DEFAULT` e outros grupos ao mesmo tempo (verifique o item 5).

   Esses exemplos são equivalentes, mas o primeiro é o preferido:

   ```
   MASTER_SITES=   alpha
   ```

   ```
   MASTER_SITES=   alpha:DEFAULT
   ```

4. Grupos não são exclusivos, um elemento pode pertencer a vários grupos diferentes ao mesmo tempo e um grupo pode ter vários elementos diferentes ou nenhum.
5. Quando um elemento pertence a vários grupos ao mesmo tempo, use uma vírgula (`,`).

   Em vez de repetir isso várias vezes, cada vez com uma pós-fixação diferente, podemos listar vários grupos de uma vez em uma única pós-fixação. Por exemplo, `:m,n,o` marca um elemento que pertence ao grupo `m`, `n` e `o`.

   Todos esses exemplos são equivalentes, mas o último é o preferido:

   ```
   MASTER_SITES=   alpha alpha:SOME_SITE
   ```

   ```
   MASTER_SITES=   alpha:DEFAULT alpha:SOME_SITE
   ```

   ```
   MASTER_SITES=   alpha:SOME_SITE,DEFAULT
   ```

   ```
   MASTER_SITES=   alpha:DEFAULT,SOME_SITE
   ```

6. Todos os sites dentro de um determinado grupo são ordernados de acordo com `MASTER_SORT_AWK`. Todos os grupos dentro de `MASTER_SITES` e `PATCH_SITES` são ordenados também.
7. A semântica de grupo pode ser usada em qualquer uma das variáveis `MASTER_SITES`, `PATCH_SITES`,

MASTER_SITE_SUBDIR, PATCH_SITE_SUBDIR, DISTFILES e PATCHFILES de acordo com esta sintaxe:

a. Todos elementos MASTER_SITES, PATCH_SITES, MASTER_SITE_SUBDIR e PATCH_SITE_SUBDIR devem ser terminados com o caractere barra /. Se algum elemento pertencer a algum grupo, o grupo de pós-fixação :n deve vir logo após o terminador /. O mecanismo MASTER_SITES:n depende da existência do terminador / para evitar confundir elementos onde um :n é uma parte válida do elemento com ocorrências em que :n denota grupo n. Para fins de compatibilidade, uma vez que o terminador / não for necessário antes em ambos elementos MASTER_SITE_SUBDIR e PATCH_SITE_SUBDIR, se o caractere precedente imediato da pós-fixação não for / então :n será considerada uma parte válida do elemento em vez de uma pós-fixação de grupo, mesmo que um elemento n seja pós-fixado. Veja ambos Uso Detalhado de MASTER_SITES:n no MASTER_SITE_SUBDIR e Uso Detalhado de MASTER_SITES:n com Vírgula, Vários Arquivos, Vários Sites e Vários Subdiretórios.

Exemplo 29. Uso Detalhado de MASTER_SITES:n no MASTER_SITE_SUBDIR

```
MASTER_SITE_SUBDIR= old:n new/:NEW
```

- Diretórios dentro do grupo DEFAULT → old:n
- Diretórios dentro do grupo NEW → new

Exemplo 30. Uso Detalhado de MASTER_SITES:n com Vírgula, Vários Arquivos, Vários Sites e Vários Subdiretórios

```
MASTER_SITES=   http://site1/%SUBDIR%/http://site2/:DEFAULT \
        http://site3/:group3 http://site4/:group4 \
        http://site5/:group5 http://site6/:group6 \
        http://site7/:DEFAULT,group6 \
        http://site8/%SUBDIR%/:group6,group7 \
        http://site9/:group8
DISTFILES=  file1 file2:DEFAULT file3:group3 \
        file4:group4,group5,group6 file5:grouping \
        file6:group7
MASTER_SITE_SUBDIR= directory-trial:1 directory-n/:groupn \
        directory-one/:group6,DEFAULT \
        directory
```

O exemplo anterior resulta em uma busca detalhada. Os sites são listados na ordem exata em que serão usados.

- arquivo1 será obtido a partir de
  - MASTER_SITE_OVERRIDE
  - http://site1/directory-trial:1/
  - http://site1/directory-one/
  - http://site1/directory/

    - http://site2/
    - http://site7/
    - MASTER_SITE_BACKUP
  - arquivo2 será baixado exatamente como o arquivo1 já que ambos pertencem ao mesmo grupo
    - MASTER_SITE_OVERRIDE
    - http://site1/directory-trial:1/
    - http://site1/directory-one/
    - http://site1/directory/
    - http://site2/
    - http://site7/
    - MASTER_SITE_BACKUP
  - arquivo3 será obtido a partir de
    - MASTER_SITE_OVERRIDE
    - http://site3/
    - MASTER_SITE_BACKUP
  - arquivo4 será obtido a partir de
    - MASTER_SITE_OVERRIDE
    - http://site4/
    - http://site5/
    - http://site6/
    - http://site7/
    - http://site8/directory-one/
    - MASTER_SITE_BACKUP
  - arquivo5 será obtido a partir de
    - MASTER_SITE_OVERRIDE
    - MASTER_SITE_BACKUP
  - file6 será obtido a partir de
    - MASTER_SITE_OVERRIDE
    - http://site8/
    - MASTER_SITE_BACKUP

8. Como posso agrupar uma das macros especiais de bsd.sites.mk, por exemplo, SourceForge (SF)?

   Isso foi simplificado o máximo possível. Veja Uso Detalhado de MASTER_SITES:n com SourceForge (SF).

Exemplo 31. Uso Detalhado de `MASTER_SITES:n` com SourceForge (`SF`)

```
MASTER_SITES=   http://site1/SF/something/1.0:sourceforge,TEST
DISTFILES=  something.tar.gz:sourceforge
```

something.tar.gz será obtido por todos os sites do SourceForge.

9. Como eu uso isso com `PATCH*`?

   Todos os exemplos foram feitos com `MASTER*` mas eles funcionam exatamente da mesma forma com `PATCH*` como pode ser visto em Uso Simplificado de `MASTER_SITES:n` com `PATCH_SITES`.

   Exemplo 32. Uso Simplificado de `MASTER_SITES:n` com `PATCH_SITES`

```
PATCH_SITES=    http://site1/ http://site2/:test
PATCHFILES= patch1:test
```

O que Muda para os Ports? O que Não Funciona?

i. Todos os ports atuais permanecem os mesmos. A feature `MASTER_SITES:n` só é ativada se houver elementos pós-fixados com `:n` como elementos de acordo com as regras de sintaxe acima, especialmente como mostrado no item 7.

1. Os targets de port permanecem os mesmos: `checksum`, `makesum`, `patch`, `configure`, `build`, etc. Com as exceções óbvias de `do-fetch`, `fetch-list`, `master-sites` e `patch-sites`.

   - `do-fetch`: implementa o novo agrupamento pós-fixado `DISTFILES` e `PATCHFILES` com seus elementos de grupo correspondentes dentro de ambos `MASTER_SITES` e `PATCH_SITES` que usam elementos de grupo correspondentes dentro de ambos `MASTER_SITE_SUBDIR` e `PATCH_SITE_SUBDIR`. Verifique Uso Detalhado de `MASTER_SITES:n` com Vírgula, Vários Arquivos, Vários Sites e Vários Subdiretórios.
   - `fetch-list`: funciona como o antigo `fetch-list`, com a exceção de que faz agrupamentos exatamente como o `do-fetch`.
   - `master-sites` e `patch-sites`: (incompatível com versões mais antigas) somente retorna os elementos do grupo `DEFAULT`; na verdade, eles executam os targets `master-sites-default` e `patch-sites-default` respectivamente.

     Além disso, usar o target `master-sites-all` ou `patch-sites-all` é o preferido para verificar diretamente `MASTER_SITES` ou `PATCH_SITES`. Além disso, não é garantido que a checagem direta funcione em versões futuras. Veja B para obter mais informações sobre esses novos tagets de port.

2. Novos Targets de Port

   a. Existem targets `master-sites-n` e `patch-sites-n` que listarão os elementos do respectivo grupo *n* dentro de `MASTER_SITES` e `PATCH_SITES` respectivamente. Por exemplo, ambos `master-`

sites-DEFAULT e patch-sites-DEFAULT retornarão os elementos do grupo DEFAULT, master-sites-test e patch-sites-test do grupo test.

a. Há novos targets master-sites-all e patch-sites-all que fazem o trabalho dos antigos master-sites e patch-sites. Eles retornam os elementos de todos os grupos como se todos pertencessem ao mesmo grupo, com a ressalva de que lista tantos MASTER_SITE_BACKUP e MASTER_SITE_OVERRIDE como existem grupos definidos dentro de qualquer DISTFILES ou PATCHFILES; respectivamente para master-sites-all e patch-sites-all.

5.3.10. DIST_SUBDIR

Não deixe o /usr/ports/distfiles bagunçado. Se um port exigir que muitos arquivos sejam baixados, ou que contenha um arquivo que tenha um nome que possa entrar em conflito com outros ports (por exemplo, Makefile), defina DIST_SUBDIR com o nome do port (${PORTNAME} ou ${PKGNAMEPREFIX}${PORTNAME}). Isso vai mudar o DISTDIR do padrão /usr/ports/distfiles para /usr/ports/distfiles/${DIST_SUBDIR},e assim, será colocado tudo o que é necessário para o port nesse subdiretório.

Ele também examinará o subdiretório com o mesmo nome no site principal de backup em http://distcache.FreeBSD.org (Configurar o DISTDIR explicitamente no Makefile não fará isso funcionar, então por favor use DIST_SUBDIR.)

Isso não afeta o MASTER_SITES definido no Makefile.

5.4. MAINTAINER

Defina seu endereço de email aqui. Por favor. *:-)*

Apenas um único endereço sem a parte de comentário é permitido como um valor para MAINTAINER. O formato usado é user@hostname.domain. Por favor, não inclua nenhum texto descritivo, como um nome nesta entrada. Isso confunde a infraestrutura do Ports e a maioria das ferramentas que a usam.

O mantenedor é responsável por manter o port atualizado e garantir que elo funcione corretamente. Para obter uma descrição detalhada das responsabilidades de um mantenedor de port, consulte O desafio para os mantenedores de port.

Um mantenedor se voluntaria para manter um port em bom estado de funcionamento. Os mantenedores têm a responsabilidade primária por seus ports, mas não possuem propriedade exclusiva. Os ports existem para o benefício da comunidade e, na realidade, pertencem à comunidade. O que isso significa é que outras pessoas além do mantenedor, também podem fazer alterações em um port. Grandes mudanças na Coleção de Ports podem exigir mudanças em muitos ports. A Equipe de Gerenciamento do Ports do FreeBSD ou membros de outras equipes podem modificar ports para corrigir problemas de dependência ou outros problemas, como um bump de versão para uma atualização de biblioteca compartilhada.

> Alguns tipos de correções tem "aprovação implícita" da Equipe de Gerenciamento do Ports portmgr@FreeBSD.org, permitindo que qualquer committer conserte essas categorias de problemas em qualquer port. Essas correções não precisam da aprovação do mantenedor.
>
> Aprovação implícita para a maioria dos ports se aplicam para correções como mudanças de infraestrutura, trivialidades e correções *testadas* de compilação e execução. A lista atual está disponibilizada em Seção Ports do Guia dos Committers.

Outras alterações no port serão enviadas ao mantenedor para revisão e aprovação antes de se fazer o commit. Se o mantenedor não responder a uma solicitação de atualização após duas semanas (excluindo os principais feriados), isso será considerado como timeout do mantenedor, e a atualização poderá ser feita sem a aprovação explícita do mesmo. Se o mantenedor não responder dentro de três meses, ou se houver três timeouts consecutivos, então o mantenedor é considerado ausente, e todas os seus ports podem ser atribuídos de volta para à comunidade. Exceções para isso são quaisquer ports mantidos pela Equipe de Gerenciamento de Ports portmgr@FreeBSD.org ou pela Equipe de Oficias de Segurança security-officer@FreeBSD.org. Nenhum commit não autorizado pode ser feito em ports mantidos por esses grupos.

Reservamo-nos o direito de modificar as submissões do mantenedor para melhor adequar as políticas e os estilos existentes da Coleção de Ports sem aprovação explicita do remetente ou do mantenedor. Além disso, grandes alterações de infraestrutura podem resultar na modificação de um port sem o consentimento do mantenedor. Estes tipos de alterações nunca irão afetar a funcionalidade do port.

A Equipe de Gerenciamento de Ports portmgr@FreeBSD.org reserva o direito de revogar ou substituir a propriedade de mantenedor de qualquer pessoa por qualquer motivo, e a Equipe de Oficiais de Segurança security-officer@FreeBSD.org reserva o direito de revogar ou substituir a propriedade de mantenedor por razões de segurança.

5.5. COMMENT

O comentário é uma descrição de uma linha de um port mostrada por `pkg info`. Por favor, siga estas regras ao compor:

1. A string COMMENT deve ter 70 caracteres ou menos.
2. *Não* inclua o nome do pacote ou o número da versão do software.
3. O comentário deve começar com uma letra maiúscula e terminar sem um ponto final.
4. Não comece com um artigo indefinido (isto é, A ou Um).
5. Capitalize nomes como Apache, JavaScript ou Perl.
6. Use uma vírgula serial para listas de palavras: "verde, vermelho, e azul."
7. Verifique erros de ortografia.

Aqui está um exemplo:

```
COMMENT=    Cat chasing a mouse all over the screen
```

A variável COMMENT vem depois da variável MAINTAINER no Makefile.

5.6. Licenças

Cada port deve documentar a licença sob a qual está disponível. Se não for uma licença aprovada pelo OSI, também deve documentar quaisquer restrições à redistribuição.

5.6.1. LICENSE

Um nome abreviado para a licença ou licenças se mais de uma licença for aplicada.

Se for uma das licenças listadas no Lista de Licenças Predefinidas, apenas as variáveis LICENSE_FILE e LICENSE_DISTFILES podem ser definidas.

Se esta for uma licença que não tenha sido definida na infraestrutura de ports (veja Lista de Licenças Predefinidas), LICENSE_PERMS e LICENSE_NAME devem ser definidos, juntamente com LICENSE_FILE ou LICENSE_TEXT. LICENSE_DISTFILES e LICENSE_GROUPS também podem ser definidos, mas não é necessário.

As licenças pré-definidas são mostradas em Lista de Licenças Predefinidas. A lista atual está sempre disponível em Mk/bsd.licenses.db.mk.

Exemplo 33. Uso Mais Simples, Licenças Predefinidas

Quando o README de algum software diz "This software is under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version." mas não fornece o arquivo de licença, use isto:

```
LICENSE=    LGPL21+
```

Quando o software fornece o arquivo de licença, use isto:

```
LICENSE=    LGPL21+
LICENSE_FILE=   ${WRKSRC}/COPYING
```

Para as licenças predefinidas, as permissões padrão são dist-mirror dist-sell pkg-mirror pkg-sell auto-accept.

Tabela 7. Lista de Licenças Predefinidas

Nome Curto	Nome	Grupo	Permissões
AGPLv3	GNU Affero General Public License version 3	FSF GPL OSI	(padrão)
AGPLv3+	GNU Affero General Public License version 3 (ou maior)	FSF GPL OSI	(padrão)
APACHE10	Apache License 1.0	FSF	(padrão)
APACHE11	Apache License 1.1	FSF OSI	(padrão)
APACHE20	Apache License 2.0	FSF OSI	(padrão)
ART10	Artistic License version 1.0	OSI	(padrão)
ART20	Artistic License version 2.0	FSF GPL OSI	(padrão)
ARTPERL10	Artistic License (perl) version 1.0	OSI	(padrão)
BSD	BSD license Generic Version (deprecated)	FSF OSI COPYFREE	(padrão)
BSD2CLAUSE	BSD 2-clause "Simplified" License	FSF OSI COPYFREE	(padrão)
BSD3CLAUSE	BSD 3-clause "New" or "Revised" License	FSF OSI COPYFREE	(padrão)
BSD4CLAUSE	BSD 4-clause "Original" or "Old" License	FSF	(padrão)
BSL	Boost Software License	FSF OSI COPYFREE	(padrão)
CC-BY-1.0	Creative Commons Attribution 1.0		(padrão)
CC-BY-2.0	Creative Commons Attribution 2.0		(padrão)
CC-BY-2.5	Creative Commons Attribution 2.5		(padrão)
CC-BY-3.0	Creative Commons Attribution 3.0		(padrão)
CC-BY-4.0	Creative Commons Attribution 4.0		(padrão)
CC-BY-NC-1.0	Creative Commons Attribution Non Commercial 1.0		dist-mirror pkg-mirror auto-accept

Nome Curto	Nome	Grupo	Permissões
`CC-BY-NC-2.0`	Creative Commons Attribution Non Commercial 2.0		`dist-mirror` `pkg-mirror` `auto-accept`
`CC-BY-NC-2.5`	Creative Commons Attribution Non Commercial 2.5		`dist-mirror` `pkg-mirror` `auto-accept`
`CC-BY-NC-3.0`	Creative Commons Attribution Non Commercial 3.0		`dist-mirror` `pkg-mirror` `auto-accept`
`CC-BY-NC-4.0`	Creative Commons Attribution Non Commercial 4.0		`dist-mirror` `pkg-mirror` `auto-accept`
`CC-BY-NC-ND-1.0`	Creative Commons Attribution Non Commercial No Derivatives 1.0		`dist-mirror` `pkg-mirror` `auto-accept`
`CC-BY-NC-ND-2.0`	Creative Commons Attribution Non Commercial No Derivatives 2.0		`dist-mirror` `pkg-mirror` `auto-accept`
`CC-BY-NC-ND-2.5`	Creative Commons Attribution Non Commercial No Derivatives 2.5		`dist-mirror` `pkg-mirror` `auto-accept`
`CC-BY-NC-ND-3.0`	Creative Commons Attribution Non Commercial No Derivatives 3.0		`dist-mirror` `pkg-mirror` `auto-accept`
`CC-BY-NC-ND-4.0`	Creative Commons Attribution Non Commercial No Derivatives 4.0		`dist-mirror` `pkg-mirror` `auto-accept`
`CC-BY-NC-SA-1.0`	Creative Commons Attribution Non Commercial Share Alike 1.0		`dist-mirror` `pkg-mirror` `auto-accept`
`CC-BY-NC-SA-2.0`	Creative Commons Attribution Non Commercial Share Alike 2.0		`dist-mirror` `pkg-mirror` `auto-accept`

Nome Curto	Nome	Grupo	Permissões
`CC-BY-NC-SA-2.5`	Creative Commons Attribution Non Commercial Share Alike 2.5		`dist-mirror` `pkg-mirror` `auto-accept`
`CC-BY-NC-SA-3.0`	Creative Commons Attribution Non Commercial Share Alike 3.0		`dist-mirror` `pkg-mirror` `auto-accept`
`CC-BY-NC-SA-4.0`	Creative Commons Attribution Non Commercial Share Alike 4.0		`dist-mirror` `pkg-mirror` `auto-accept`
`CC-BY-ND-1.0`	Creative Commons Attribution No Derivatives 1.0		(padrão)
`CC-BY-ND-2.0`	Creative Commons Attribution No Derivatives 2.0		(padrão)
`CC-BY-ND-2.5`	Creative Commons Attribution No Derivatives 2.5		(padrão)
`CC-BY-ND-3.0`	Creative Commons Attribution No Derivatives 3.0		(padrão)
`CC-BY-ND-4.0`	Creative Commons Attribution No Derivatives 4.0		(padrão)
`CC-BY-SA-1.0`	Creative Commons Attribution Share Alike 1.0		(padrão)
`CC-BY-SA-2.0`	Creative Commons Attribution Compartilhar Alike 2.0		(padrão)
`CC-BY-SA-2.5`	Creative Commons Attribution Share Alike 2.5		(padrão)
`CC-BY-SA-3.0`	Creative Commons Attribution Share Alike 3.0		(padrão)

Nome Curto	Nome	Grupo	Permissões
`CC-BY-SA-4.0`	Creative Commons Attribution Share Alike 4.0		(padrão)
`CC0-1.0`	Creative Commons Zero v1.0 Universal	`FSF GPL COPYFREE`	(padrão)
`CDDL`	Common Development and Distribution License	`FSF OSI`	(padrão)
`CPAL-1.0`	Common Public Attribution License	`FSF OSI`	(padrão)
`ClArtistic`	Clarified Artistic License	`FSF GPL OSI`	(padrão)
`EPL`	Eclipse Public License	`FSF OSI`	(padrão)
`GFDL`	GNU Free Documentation License	`FSF`	(padrão)
`GMGPL`	GNAT Modified General Public License	`FSF GPL OSI`	(padrão)
`GPLv1`	GNU General Public License version 1	`FSF GPL OSI`	(padrão)
`GPLv1+`	GNU General Public License version 1 (or later)	`FSF GPL OSI`	(padrão)
`GPLv2`	GNU General Public License version 2	`FSF GPL OSI`	(padrão)
`GPLv2+`	GNU General Public License version 2 (or later)	`FSF GPL OSI`	(padrão)
`GPLv3`	GNU General Public License version 3	`FSF GPL OSI`	(padrão)
`GPLv3+`	GNU General Public License version 3 (or later)	`FSF GPL OSI`	(padrão)
`GPLv3RLE`	GNU GPL version 3 Runtime Library Exception	`FSF GPL OSI`	(padrão)
`GPLv3RLE+`	GNU GPL version 3 Runtime Library Exception (or later)	`FSF GPL OSI`	(padrão)
`ISCL`	Internet Systems Consortium License	`FSF GPL OSI COPYFREE`	(padrão)

Nome Curto	Nome	Grupo	Permissões
`LGPL20`	GNU Library General Public License version 2.0	`FSF GPL OSI`	(padrão)
`LGPL20+`	GNU Library General Public License version 2.0 (or later)	`FSF GPL OSI`	(padrão)
`LGPL21`	GNU Lesser General Public License version 2.1	`FSF GPL OSI`	(padrão)
`LGPL21+`	GNU Lesser General Public License version 2.1 (or later)	`FSF GPL OSI`	(padrão)
`LGPL3`	GNU Lesser General Public License version 3	`FSF GPL OSI`	(padrão)
`LGPL3+`	GNU Lesser General Public License version 3 (or later)	`FSF GPL OSI`	(padrão)
`LPPL10`	LaTeX Project Public License version 1.0	`FSF OSI`	`dist-mirror dist-sell`
`LPPL11`	LaTeX Project Public License version 1.1	`FSF OSI`	`dist-mirror dist-sell`
`LPPL12`	LaTeX Project Public License version 1.2	`FSF OSI`	`dist-mirror dist-sell`
`LPPL13`	LaTeX Project Public License version 1.3	`FSF OSI`	`dist-mirror dist-sell`
`LPPL13a`	LaTeX Project Public License version 1.3a	`FSF OSI`	`dist-mirror dist-sell`
`LPPL13b`	LaTeX Project Public License version 1.3b	`FSF OSI`	`dist-mirror dist-sell`
`LPPL13c`	LaTeX Project Public License version 1.3c	`FSF OSI`	`dist-mirror dist-sell`
`MIT`	MIT license / X11 license	`COPYFREE FSF GPL OSI`	(padrão)
`MPL10`	Mozilla Public License version 1.0	`FSF OSI`	(padrão)
`MPL11`	Mozilla Public License version 1.1	`FSF OSI`	(padrão)
`MPL20`	Mozilla Public License version 2.0	`FSF OSI`	(padrão)

Nome Curto	Nome	Grupo	Permissões
NCSA	University of Illinois/NCSA Open Source License	COPYFREE FSF GPL OSI	(padrão)
NONE	No license specified		none
OFL10	SIL Open Font License version 1.0 (http://scripts.sil.org/OFL)	FONTS	(padrão)
OFL11	SIL Open Font License version 1.1 (http://scripts.sil.org/OFL)	FONTS	(padrão)
OWL	Open Works License (owl.apotheon.org)	COPYFREE	(padrão)
OpenSSL	Licença OpenSSL	FSF	(padrão)
PD	Public Domain	GPL COPYFREE	(padrão)
PHP202	PHP License version 2.02	FSF OSI	(padrão)
PHP30	PHP License version 3.0	FSF OSI	(padrão)
PHP301	PHP License versão 3.01	FSF OSI	(padrão)
PSFL	Python Software Foundation License	FSF GPL OSI	(padrão)
PostgreSQL	PostgreSQL License	FSF GPL OSI COPYFREE	(padrão)
RUBY	Ruby License	FSF	(padrão)
UNLICENSE	The Unlicense	COPYFREE FSF GPL	(padrão)
WTFPL	Do What the Fuck You Want To Public License version 2	GPL FSF COPYFREE	(padrão)
WTFPL1	Do What the Fuck You Want To Public License version 1	GPL FSF COPYFREE	(padrão)
ZLIB	zlib License	GPL FSF OSI	(padrão)
ZPL21	Zope Public License version 2.1	GPL OSI	(padrão)

5.6.2. LICENSE_PERMS e LICENSE_PERMS_NAME

Permissões. Use none se vazio.

`dist-mirror`

A redistribuição dos arquivos de distribuição é permitida. Os arquivos de distribuição serão adicionados ao FreeBSD CDN `MASTER_SITE_BACKUP`.

`no-dist-mirror`

A redistribuição dos arquivos de distribuição é proibida. Isso é equivalente a `RESTRICT`. Os arquivos de distribuição *não* serão adicionados ao FreeBSD CDN `MASTER_SITE_BACKUP`.

`dist-sell`

A venda de arquivos de distribuição é permitida. Os arquivos de distribuição estarão presentes nas imagens do instalador.

`no-dist-sell`

A venda de arquivos de distribuição é proibida. Isso é equivalente a `NO_CDROM`.

`pkg-mirror`

É permitida a redistribuição gratuita do pacote. O pacote será distribuído na CDN de pacotes do FreeBSD https://pkg.freebsd.org/.

`no-pkg-mirror`

É proibida a redistribuição gratuita do pacote. Equivalente à definir `NO_PACKAGE`. O pacote *não* será distribuído a partir da CDN de pacotes do FreeBSD https://pkg.freebsd.org/.

`pkg-sell`

A venda do pacote é permitida. O pacote estará presente nas imagens do instalador.

`no-pkg-sell`

A venda de pacotes é proibida. Isso é equivalente a definir `NO_CDROM`. O pacote *não* estará presente nas imagens do instalador.

`auto-accept`

A licença é aceita por padrão. Os prompts para aceitar uma licença não são exibidos a menos que o usuário tenha definido `LICENSES_ASK`. Use isto, a menos que a licença indique que o usuário deve aceitar os termos da licença.

`no-auto-accept`

A licença não é aceita por padrão. O usuário sempre será solicitado a confirmar a aceitação desta licença. Isso deve ser usado se a licença declarar que o usuário deve aceitar seus termos.

Quando ambos `permission` e `no-permission` estiverem presentes o `no-permission` vai cancelar a `permission`.

Quando `permission` não estiver presente, é considerado uma `no-permission`.

Algumas permissões que estiverem faltando, impedirão que um port (e todos os ports dependendo dele) sejam utilizados pelos usuários do pacote:

> Um port sem a permissão `auto-accept` nunca será compilado e todos os ports dependendo dele serão ignorados.
>
> Um port sem a permissão `pkg-mirror` será removido, assim como todos os ports que dependam dele, isso depois da compilação e então eles nunca serão distribuídos.

Exemplo 34. Licença Não Padrão

Leia os termos da licença e traduza-os usando as permissões disponíveis.

```
LICENSE=        UNKNOWN
LICENSE_NAME=   unknown
LICENSE_TEXT=   Esse programa NÃO é de domínio publico.\
                pode ser distribuído livremente para propósitos não comerciais
apenas,\
                e NÃO HÁ GARANTIA PARA ESSE PROGRAMA.
LICENSE_PERMS=  dist-mirror no-dist-sell pkg-mirror no-pkg-sell auto-accept
```

Exemplo 35. Licenças Padrão e Não Padrão

Leia os termos da licença e expresse-os usando as permissões disponíveis. Em caso de dúvida, peça orientação na lista de discussão de ports do FreeBSD.

```
LICENSE=        WARSOW GPLv2
LICENSE_COMB=   multi
LICENSE_NAME_WARSOW=    Warsow Content License
LICENSE_FILE_WARSOW=    ${WRKSRC}/docs/license.txt
LICENSE_PERMS_WARSOW=   dist-mirror pkg-mirror auto-accept
```

Quando as permissões das licenças GPLv2 e UNKNOWN são misturadas, o port termina com `dist-mirror dist-sell pkg-mirror pkg-sell auto-accept dist-mirror no-dist-sell pkg-mirror no-pkg-sell auto-accept`. O `no-permissions` cancela as *permissions*. A lista resultante de permissões é *dist-mirror pkg-mirror auto-accept*. Os arquivos de distribuição e os pacotes não estarão disponíveis nas imagens do instalador.

5.6.3. `LICENSE_GROUPS` e `LICENSE_GROUPS_NAME`

Grupos que a licença pertence.

Lista de Grupos de Licenças Predefinidas

`FSF`

Aprovada pela Free Software Foundation, veja FSF Licensing & Compliance Team.

`GPL`

Compatível com GPL

`OSI`

Aprovado pelo OSI, veja a pagina Open Source Initiative Open Source Licenses.

`COPYFREE`

Segue a Copyfree Standard Definition, consulte a pagina Copyfree Licenses.

`FONTS`

Licenças de fonte

5.6.4. `LICENSE_NAME` e `LICENSE_NAME_NAME`

Nome completo da licença.

Exemplo 36. `LICENSE_NAME`

```
LICENSE=        UNRAR
LICENSE_NAME=   UnRAR License
LICENSE_FILE=   ${WRKSRC}/license.txt
LICENSE_PERMS=  dist-mirror dist-sell pkg-mirror pkg-sell auto-accept
```

5.6.5. `LICENSE_FILE` e `LICENSE_FILE_NOME`

Caminho completo para o arquivo que contém o texto da licença, geralmente ${WRKSRC}/some/file. Se o arquivo não estiver no distfile e seu conteúdo for muito longo para ser colocado em `LICENSE_TEXT`, insira o texto em um novo arquivo em ${FILESDIR}.

Exemplo 37. `LICENSE_FILE`

```
LICENSE=    GPLv3+
LICENSE_FILE=   ${WRKSRC}/COPYING
```

5.6.6. `LICENSE_TEXT` e `LICENSE_TEXT_NAME`

Texto para usar como uma licença. Útil quando a licença não está nos arquivos de distribuição e seu texto é curto.

Exemplo 38. `LICENSE_TEXT`

```
LICENSE=        UNKNOWN
LICENSE_NAME=   unknown
LICENSE_TEXT=   This program is NOT in public domain.\
                It can be freely distributed for non-commercial purposes only,\
                and THERE IS NO WARRANTY FOR THIS PROGRAM.
LICENSE_PERMS=  dist-mirror no-dist-sell pkg-mirror no-pkg-sell auto-accept
```

5.6.7. LICENSE_DISTFILES e LICENSE_DISTFILES_NAME

Os arquivos de distribuição aos quais as licenças se aplicam. O padrão é para todos os arquivos de distribuição.

Exemplo 39. LICENSE_DISTFILES

Usado quando os arquivos de distribuição não possuem a mesma licença. Por exemplo, um possui uma licença de código e outro possui alguns trabalhos de arte que não podem ser redistribuídos:

```
MASTER_SITES=   SF/some-game
DISTFILES=      ${DISTNAME}${EXTRACT_SUFX} artwork.zip

LICENSE=        BSD3CLAUSE ARTWORK
LICENSE_COMB=   dual
LICENSE_NAME_ARTWORK=      The game artwork license
LICENSE_TEXT_ARTWORK=      The README says that the files cannot be redistributed
LICENSE_PERMS_ARTWORK=     pkg-mirror pkg-sell auto-accept
LICENSE_DISTFILES_BSD3CLAUSE=   ${DISTNAME}${EXTRACT_SUFX}
LICENSE_DISTFILES_ARTWORK= artwork.zip
```

5.6.8. LICENSE_COMB

Defina como `multi` se todas as licenças se aplicarem. Defina como `dual` se qualquer uma das licenças se aplica. O padrão é definido para `single`.

Exemplo 40. Licenças Duplas

Quando um port diz "This software may be distributed under the GNU General Public License or the Artistic License">, isso significa que qualquer licença pode ser usada. Use isto:

```
LICENSE=    ART10 GPLv1
LICENSE_COMB=   dual
```

Se os arquivos de licença forem fornecidos, use assim:

```
LICENSE=    ART10 GPLv1
LICENSE_COMB=   dual
LICENSE_FILE_ART10=     ${WRKSRC}/Artistic
LICENSE_FILE_GPLv1=     ${WRKSRC}/Copying
```

Exemplo 41. Múltiplas Licenças

Quando parte de um port tem uma licença, e outra parte tem uma licença diferente, use `multi`:

```
LICENSE=    GPLv2 LGPL21+
LICENSE_COMB=   multi
```

5.7. PORTSCOUT

Portscout é um utilitário de verificação de distfile automatizado para a Coleção de Ports do FreeBSD, descrito em detalhes em Portscout: o Scanner de Distfile de Ports do FreeBSD.

PORTSCOUT define condições especiais dentro das quais o scanner distfile do Portscout é restrito.

Situações em que o PORTSCOUT é configurado:

- Quando distfiles precisam ser ignorados, seja para versões específicas, ou para pequenas revisões específicas. Por exemplo, para excluir a versão *8.2* das verificações de versão de distfile porque é conhecido por estar quebrado, adicione:

  ```
  PORTSCOUT=  ignore:8.2
  ```

- Quando versões específicas ou revisões maiores e menores específicas de um distfile devem ser verificadas. Por exemplo, se somente a versão *0.6.4* deve ser monitorado porque versões mais recentes têm problemas de compatibilidade com o FreeBSD, adicione:

  ```
  PORTSCOUT=  limit:^0\.6\.4
  ```

- Quando os URLs que listam as versões disponíveis diferem dos URLs de download. Por exemplo, para limitar as verificações de versão do arquivo distfile à página de download para o port databases/pgtune, adicione:

  ```
  PORTSCOUT=  site:http://pgfoundry.org/frs/?group_id=1000416
  ```

5.8. Dependências

Muitos ports dependem de outros ports. Esta é uma característica muito conveniente da maioria dos sistemas operacionais Unix-like, incluindo FreeBSD. Vários ports podem compartilhar uma dependência comum, ao invés de agrupar essa dependência com cada port ou pacote que precisa dela. Há sete variáveis que podem ser usadas para garantir que todos os bits necessários estejam na máquina do usuário. Existem também algumas variáveis de dependência pré-suportadas para casos comuns, além de algumas outras para controlar o comportamento das dependências.

> Quando o software possui dependências extras que fornecem recursos extras, as dependências básicas listadas em *_DEPENDS devem incluir as dependências extras que beneficiariam a maioria dos usuários. As dependências básicas nunca devem ser um conjunto de dependências "mínima". O objetivo não é incluir todas as

dependências possíveis. Inclua apenas aquelas que beneficiarão a maioria das pessoas.

5.8.1. LIB_DEPENDS

Esta variável especifica as bibliotecas compartilhadas das quais este port depende. É uma lista de tuplas *lib:dir* onde *lib* é o nome da biblioteca compartilhada, *dir* é o diretório no qual encontrá-lo, caso não esteja disponível. Por exemplo,

```
LIB_DEPENDS=   libjpeg.so:graphics/jpeg
```

irá verificar se há uma biblioteca jpeg compartilhada com qualquer versão no subdiretório graphics/jpeg da árvore de ports para compilar e instalar se não for encontrado.

A dependência é verificada duas vezes, uma vez dentro do target `build` e depois dentro do target `install`. Além disso, o nome da dependência é colocado no pacote para que o `pkg-install` (veja pkg-install(8)) a instale automaticamente se a mesma não estiver no sistema do usuário.

5.8.2. RUN_DEPENDS

Esta variável especifica arquivos executáveis ou arquivos para os quais este port depende durante o tempo de execução. É uma lista de tuplas path:dir:[`target`] onde *path* é o nome do executável ou arquivo,*dir* é o diretório no qual encontrá-lo, caso não esteja disponível, e o *target* é o target para chamar nesse diretório. E se o *path* começar com uma barra (/), ele será tratado como um arquivo e sua existência é testada com `test -e`; caso contrário, é assumido como um executável e `which -s` é usado para determinar se o programa existe no caminho de pesquisa.

Por exemplo,

```
RUN_DEPENDS=    ${LOCALBASE}/news/bin/innd:news/inn \
        xmlcatmgr:textproc/xmlcatmgr
```

irá verificar se o arquivo ou diretório /usr/local/news/bin/innd existe, e compilar e instalá-lo a partir do subdiretório news/inn da árvore de ports, caso não seja encontrado. Ele também verá se um executável chamado `xmlcatmgr` está no caminho de pesquisa em textproc/xmlcatmgr para compilar e instalar se não for encontrado.

Nesse caso, `innd` é na verdade um executável; se um executável estiver em um local que não deve estar no caminho de pesquisa, use o nome do caminho completo.

A pesquisa oficial `PATH` usado no cluster de construção de ports é

```
/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
```

A dependência é verificada a partir do target `install`. Além disso, o nome da dependência é colocado no pacote para que o `pkg-install` (veja pkg-install(8)) a instale automaticamente se a mesma não estiver no sistema do usuário. A parte *target* pode ser omitida se for igual a `DEPENDS_TARGET`.

Uma situação bastante comum é quando `RUN_DEPENDS` é literalmente o mesmo que `BUILD_DEPENDS`, especialmente se o software portado é escrito em uma linguagem de script ou se requer o mesmo ambiente de compilação e tempo de execução. Neste caso, é tentador e intuitivo atribuir diretamente um ao outro:

```
RUN_DEPENDS=    ${BUILD_DEPENDS}
```

No entanto, essa atribuição pode poluir as dependências de tempo de execução com entradas não definidas no `BUILD_DEPENDS` original do port. Isso acontece por causa de uma avaliação preguiçosa de atribuição de variáveis do make(1). Considere um Makefile com `USES_*`, que são processados por ports/Mk/bsd.*.mk para aumentar as dependências iniciais de compilação. Por exemplo, `USES=gmake` adiciona devel/gmake para `BUILD_DEPENDS`. Para evitar que essas dependências adicionais poluam `RUN_DEPENDS`, crie outra variável com o conteúdo atual de `BUILD_DEPENDS` e atribua-a para ambos `BUILD_DEPENDS` e `RUN_DEPENDS`:

```
MY_DEPENDS= some:devel/some \
        other:lang/other
BUILD_DEPENDS=  ${MY_DEPENDS}
RUN_DEPENDS=    ${MY_DEPENDS}
```

Não use `:=` para atribuir `BUILD_DEPENDS` para `RUN_DEPENDS` ou vice-versa. Todas as variáveis são expandidas imediatamente, o que é exatamente a coisa errada a fazer e quase sempre um fracasso.

5.8.3. BUILD_DEPENDS

Esta variável especifica executáveis ou arquivos que este port requer para ser compilado. Como `RUN_DEPENDS`, ela é uma lista de tuplas path:dir:[`target`]. Por exemplo,

```
BUILD_DEPENDS=  unzip:archivers/unzip
```

irá procurar por um executável chamado `unzip`, e ir para o subdiretório archivers/unzip da árvore de ports para compilar e instalar se não for encontrado.

"build" aqui significa tudo, desde a extração até a compilação. A dependência é verificada a partir do target `extract`. A parte do *target* pode ser omitida se for igual a `DEPENDS_TARGET`

5.8.4. FETCH_DEPENDS

Esta variável especifica executáveis ou arquivos que este port requer para fazer os downloads. Como os dois anteriores, é uma lista de tuplas path:dir:[target]. Por exemplo,

```
FETCH_DEPENDS=  ncftp2:net/ncftp2
```

irá procurar por um executável chamado ncftp2 e ir para o subdiretório net/ncftp2 da árvore de ports para compilar e instalar se não for encontrado.

A dependência é verificada a partir do target fetch. A parte *target* pode ser omitida se for igual a DEPENDS_TARGET.

5.8.5. EXTRACT_DEPENDS

Esta variável especifica executáveis ou arquivos que este port requer para extração. Como no anterior, é uma lista de tuplas path:dir:[target]. Por exemplo,

```
EXTRACT_DEPENDS=    unzip:archivers/unzip
```

irá procurar por um executável chamado unzip, e ir para o subdiretório archivers/unzip da árvore de ports para compilar e instalar se não for encontrado.

A dependência é verificada a partir do target extract. A parte *target* pode ser omitida se for igual a DEPENDS_TARGET.

Use esta variável somente se a extração ainda não funcionar (o padrão usa tar) e não funciona com USES=tar, USES=lha ou USES=zip descrito em Usando Macros USES.

5.8.6. PATCH_DEPENDS

Esta variável especifica executáveis ou arquivos que este port requer para aplicar patches. Como no anterior, é uma lista de path:dir:[target]. Por exemplo,

```
PATCH_DEPENDS=  ${NONEXISTENT}:java/jfc:extract
```

vai descer para o subdiretório java/jfc da árvore de ports para extraí-lo.

A dependência é verificada a partir do target patch. A parte *target* pode ser omitida se for igual a DEPENDS_TARGET.

5.8.7. USES

Parâmetros podem ser adicionados para definir diferentes recursos e dependências usados pelo port. Eles são especificados adicionando esta linha ao Makefile:

```
USES= feature[:arguments]
```

Para a lista completa de valores, por favor veja o Usando Macros USES.

USES não pode ser atribuído após a inclusão de bsd.port.pre.mk.

5.9. O USE_*

Diversas variáveis existem para definir dependências comuns compartilhadas por muitos ports. O uso é opcional, mas ajuda a reduzir a verbosidade dos Makefiles de port . Cada um deles é denominado como USES_*. Essas variáveis podem ser usadas apenas no Makefile do port e ports/Mk/bsd.*.mk. Elas não são destinadas a opções configuráveis pelo usuário - use PORT_OPTIONS para esse propósito.

É *sempre* incorreto definir qualquer USE_* dentro de /etc/make.conf. Por exemplo, definindo

```
USE_GCC=X.Y
```

(onde XY é o número da versão) adicionaria uma dependência do gccXY para cada port, incluindo lang/gccXY em si!

Tabela 8. USE_*

Variável	Significa
USE_GCC	O port requer GCC (gcc ou {gcc-plus-plus}) para compilar. Alguns ports precisam de qualquer versão do GCC, algumas exigem versões modernas e recentes. Normalmente, é configurado para qualquer (neste caso, o GCC da base seria usado em versões do FreeBSD que ainda o possuem, ou o port lang/gcc seria instalado quando o compilador C/C++ padrão for o Clang); ou yes (significa usar sempre GCC estável e moderno do port lang/gcc). A versão exata também pode ser especificada, com um valor como 4.7. A versão mínima exigida pode ser especificada como 4.6+. O GCC do sistema base é usado quando satisfaz a versão solicitada, caso contrário, um compilador apropriado é compilado a partir do port, e CC e CXX são ajustados em conformidade. USE_GCC irá registrar uma dependência de tempo de compilação e uma de tempo de execução.

Variáveis relacionadas ao gmake e configure são descritos em Mecanismos de Compilação, enquanto autoconf, automake e libtool são descritos em Usando o GNU Autotools. Variáveis relacionadas ao Perl são descritas em Usando Perl. Variáveis X11 são listadas em Usando o X11. Usando o GNOME lida com o GNOME e Usando o KDE com variáveis relacionadas ao KDE. Usando Java documenta variáveis Java, enquanto Aplicações Web contém informações sobre Apache, PHP e módulos PEAR. Python é discutido em Usando Python, e Ruby em Usando Ruby. Usando SDL fornece variáveis usadas para aplicações SDL e, finalmente, Usando o Xfce contém informações sobre o Xfce.

5.9.1. Versão Mínima de uma Dependência

Uma versão mínima de uma dependência pode ser especificada em qualquer *_DEPENDS, exceto LIB_DEPENDS, usando esta sintaxe:

```
p5-Spiffy>=0.26:devel/p5-Spiffy
```

O primeiro campo contém um nome de pacote dependente, que deve corresponder à entrada no banco de dados de pacotes, um sinal de comparação e uma versão do pacote. A dependência é satisfeita se o p5-Spiffy-0.26 ou mais recente estiver instalado na máquina.

5.9.2. Notas sobre Dependências

Como mencionado acima, o target padrão para chamar quando uma dependência é necessária é o `DEPENDS_TARGET`. Seu padrão é o `install`. Esta é uma variável de usuário; nunca é definido em um Makefile de port. Se o port precisar de uma maneira especial de lidar com uma dependência, use a parte `:target` de `*_DEPENDS` em vez de redefinir `DEPENDS_TARGET`.

Quando rodar `make clean`, as dependências de port também são limpas automaticamente. Se isso não for desejável, defina `NOCLEANDEPENDS` no ambiente. Isto pode ser particularmente desejável se o port tiver algo que demore muito tempo para recompilar em sua lista de dependências, como o KDE, o GNOME ou o Mozilla.

Para depender de outro port incondicionalmente, use a variável `${NONEXISTENT}` no primeiro campo do `BUILD_DEPENDS` ou `RUN_DEPENDS`. Use isto somente quando o código fonte do outro port for necessário. Tempo de compilação pode ser economizado especificando o target também. Por exemplo

```
BUILD_DEPENDS=  ${NONEXISTENT}:graphics/jpeg:extract
```

sempre descerá para o port `jpeg` e extrai-lo.

5.9.3. Dependências Circulares são Fatais

Não insira nenhuma dependência circular na árvore de ports!

A tecnologia de compilação de ports não tolera dependências circulares. Se uma for inserida, alguém, em algum lugar do mundo, terá sua instalação do FreeBSD quebrada quase que imediatamente, e muitos outros rapidamente terão o mesmo problema. Estes erros podem ser realmente difíceis de serem detectados. Em caso de dúvida, antes de fazer qualquer alteração, certifique-se de executar: `cd /usr/ports; make index`. Esse processo pode ser muito lento em máquinas mais antigas, mas pode evitar dor de cabeça para um grande número de pessoas, incluindo você.

5.9.4. Problemas Causados por Dependências Automáticas

Dependências devem ser declaradas explicitamente ou usando o framework OPTIONS. Usar outros métodos, como a detecção automática, dificulta a indexação, o que causa problemas para o gerenciamento de ports e pacotes.

Exemplo 42. Declaração Errada de uma Dependência Opcional

```
.include <bsd.port.pre.mk>

.if exists(${LOCALBASE}/bin/foo)
LIB_DEPENDS=    libbar.so:foo/bar
.endif
```

O problema em tentar adicionar dependências automaticamente é que os arquivos e configurações fora de um port individual podem ser alterados a qualquer momento. Por exemplo: um índice é construído, depois um lote de ports é instalado. Mas um dos ports instala o arquivo testado. O índice então fica incorreto, porque um port instalado inesperadamente tem uma nova dependência. O índice ainda pode estar errado mesmo após a recriação, se outros ports também determinarem a necessidade de dependências com base na existência de outros arquivos.

Exemplo 43. Declaração Correta de uma Dependência Opcional

```
OPTIONS_DEFINE= BAR
BAR_DESC=   Calling cellphones via bar

BAR_LIB_DEPENDS=    libbar.so:foo/bar
```

Testar variáveis de opções é o método correto. Ele não causará inconsistências no índice de um lote de ports, desde que as opções tenham sido definidas antes da construção do índice. Scripts simples podem ser usados para automatizar a compilação, instalação e atualização desses ports e seus pacotes.

5.10. Ports Slaves e `MASTERDIR`

Se o port precisar criar versões ligeiramente diferentes de pacotes fazendo com que uma variável (por exemplo, resolução ou tamanho de papel) assuma valores diferentes, crie um subdiretório por pacote para facilitar aos usuários a visualização do que fazer, mas tente compartilhar o máximo possível de arquivos entre os ports. Normalmente, usando variáveis inteligentemente, apenas um Makefile bem curto será necessário em todos, exceto em um dos diretórios. No Makefile solitário, use `MASTERDIR` para especificar o diretório onde o restante dos arquivos estão. Além disso, use uma variável como parte de `PKGNAMESUFFIX` para que os pacotes tenham nomes diferentes.

Isso será melhor demonstrado por um exemplo. Isso é parte de print/pkfonts300/Makefile;

```
PORTNAME=   pkfonts${RESOLUTION}
PORTVERSION=    1.0
DISTFILES=  pk${RESOLUTION}.tar.gz

PLIST=      ${PKGDIR}/pkg-plist.${RESOLUTION}

.if !defined(RESOLUTION)
RESOLUTION= 300
.else
.if ${RESOLUTION} != 118 && ${RESOLUTION} != 240 && \
    ${RESOLUTION} != 300 && ${RESOLUTION} != 360 && \
    ${RESOLUTION} != 400 && ${RESOLUTION} != 600
.BEGIN:
    @${ECHO_MSG} "Error: invalid value for RESOLUTION: \"${RESOLUTION}\""
    @${ECHO_MSG} "Possible values are: 118, 240, 300, 360, 400 and 600."
    @${FALSE}
```

```
.endif
.endif
```

print/pkfonts300 também tem todos os patches, arquivos de pacotes, etc. Rodando `make` nele, será assumido o valor padrão para a resolução (300) e o port será compilado normalmente.

Quanto às outras resoluções, este é o print/pkfonts360/Makefile*completo*:

```
RESOLUTION= 360
MASTERDIR=  ${.CURDIR}/../pkfonts300

.include    "${MASTERDIR}/Makefile"
```

(print/pkfonts118/Makefile, print/pkfonts600/Makefile, e todos os outros são semelhantes). A definição de `MASTERDIR` diz ao bsd.port.mk que o conjunto regular de subdiretórios como `FILESDIR` e `SCRIPTDIR` podem ser encontrados em pkfonts300. A linha `RESOLUTION=360` irá substituir a linha `RESOLUTION=300` em pkfonts300/Makefile e o port será compilado com a resolução definida para 360.

5.11. Páginas de Manual

Se o port instala a sua árvore de manuais em outro lugar diferente de `PREFIX`, use `MANDIRS` para especificar esses diretórios. Note que os arquivos correspondentes às páginas de manual devem ser colocados no pkg-plist junto com o resto dos arquivos. O propósito do `MANDIRS` é ativar a compactação automática de páginas de manual, portanto, os nomes dos arquivos são sufixados com .gz.

5.12. Arquivos de Informação

Se o pacote precisar instalar arquivos de informações GNU, liste-os na variável `INFO` (sem o sufixo `.info`), uma entrada por documento. Presume-se que esses arquivos estejam instalados em PREFIX/INFO_PATH. Mude `INFO_PATH` se o pacote usa um local diferente. Contudo, isto não é recomendado. Essas entradas contêm apenas o caminho relativo para PREFIX/INFO_PATH. Por exemplo, lang/gcc34 instala arquivos de informações em PREFIX/INFO_PATH/gcc34 e `INFO` será algo assim:

```
INFO=   gcc34/cpp gcc34/cppinternals gcc34/g77 ...
```

O código apropriado de instalação/desinstalação será automaticamente adicionado ao arquivo pkg-plist temporário antes do registro do pacote.

5.13. Opções do Makefile

Muitas aplicações podem ser compiladas com configurações opcionais ou diferentes. Exemplos podem ser a escolha de linguagem natural (humana), GUI versus linha de comando ou qual tipo de banco de dados será suportado. Os usuários podem precisar de uma configuração diferente do

padrão, portanto o sistema de ports fornece ganchos em que o autor do port pode usar para controlar qual variante será compilada. Suportar essas opções corretamente fará com que os usuários fiquem felizes, e efetivamente forneça dois ou mais ports pelo preço de um.

5.13.1. OPTIONS

Background

OPTIONS_* fornece ao usuário que está instalando o port uma caixa de diálogo mostrando as opções disponíveis e, em seguida, salva essas opções em ${PORT_DBDIR}/${OPTIONS_NAME}/options. Na próxima vez que o port for compilado, as opções serão reutilizadas. O padrão de PORT_DBDIR é /var/db/ports. OPTIONS_NAME é a origem do port com um underline como o separador de espaço, por exemplo, dns/bind99 será dns_bind99.

Quando o usuário executa make config (ou executa make build pela primeira vez), o framework verifica ${PORT_DBDIR}/${OPTIONS_NAME}/options. Se esse arquivo não existir, os valores de OPTIONS_* são usados e uma caixa de diálogo é exibida onde as opções podem ser ativadas ou desativadas. Então as options são salvas e as variáveis configuradas são utilizadas ao compilar o port.

Se uma nova versão do port adicionar novas OPTIONS, a caixa de diálogo será apresentada ao usuário, já preenchido com os valores salvos das antigas OPTIONS.

make showconfig mostra a configuração salva. Use make rmconfig para remover a configuração salva.

Sintaxe

OPTIONS_DEFINE contém uma lista de OPTIONS para serem utilizadas. Elas são independentes umas das outras e não são agrupadas:

```
OPTIONS_DEFINE= OPT1 OPT2
```

Uma vez definido, as OPTIONS são descritas (opcionalmente, mas fortemente recomendado):

```
OPT1_DESC=  Describe OPT1
OPT2_DESC=  Describe OPT2
OPT3_DESC=  Describe OPT3
OPT4_DESC=  Describe OPT4
OPT5_DESC=  Describe OPT5
OPT6_DESC=  Describe OPT6
```

ports/Mk/bsd.options.desc.mk possui descrições para muitas OPTIONS comuns. Geralmente são úteis, mas podem ser substituas se a descrição for insuficiente para o port.

Ao descrever as opções, visualize-as da perspectiva do usuário: "Qual funcionalidade ela muda?" e "Por que eu iria querer habilitar ela?" Não repita apenas o nome. Por exemplo, descrever a opção NLS como "incluir suporte NLS"

não ajuda o usuário, que já pode ver o nome da opção, mas pode não saber o que isso significa. Descrevendo-a como "Suporte a idiomas nativos por meio de utilitários gettext" é muito mais útil.

Os nomes das opções são sempre em letras maiúsculas. Não podem estar misturadas ou apenas em minúsculo.

`OPTIONS` podem ser agrupadas como opções radio, onde apenas uma escolha de cada grupo é permitida:

```
OPTIONS_SINGLE=     SG1
OPTIONS_SINGLE_SG1= OPT3 OPT4
```

Deve estar sempre selecionada uma de cada `OPTIONS_SINGLE` para as opções serem válidas. Uma opção de cada grupo *deve* ser adicionada a `OPTIONS_DEFAULT`.

`OPTIONS` podem ser agrupadas como opções radio, onde nenhuma ou apenas uma escolha de cada grupo é permitida:

```
OPTIONS_RADIO=      RG1
OPTIONS_RADIO_RG1=  OPT7 OPT8
```

`OPTIONS` também pode ser agrupadas como listas de "múltipla-escolha", onde *pelo menos uma* opção deve estar habilitada:

```
OPTIONS_MULTI=      MG1
OPTIONS_MULTI_MG1=  OPT5 OPT6
```

`OPTIONS` também pode ser agrupadas como listas de "múltipla-escolha", onde nenhuma ou qualquer opção pode ser ativada:

```
OPTIONS_GROUP=      GG1
OPTIONS_GROUP_GG1=  OPT9 OPT10
```

`OPTIONS` são desativadas por padrão, a menos que estejam listadas em `OPTIONS_DEFAULT`:

```
OPTIONS_DEFAULT=    OPT1 OPT3 OPT6
```

Definições de `OPTIONS` devem aparecer antes da inclusão de bsd.port.options.mk. Valores de `PORT_OPTIONS` só podem ser testados após a inclusão de bsd.port.options.mk. Inclusão de bsd.port.pre.mk pode ser usado também, e ainda é amplamente usado em ports escritos antes da introdução de bsd.port.options.mk. Mas esteja ciente de que algumas variáveis não funcionarão como esperado após a inclusão de bsd.port.pre.mk, tipicamente algumas flags `USE_*`.

Exemplo 44. Uso Simples de `OPTIONS`

```
OPTIONS_DEFINE= FOO BAR
OPTIONS_DEFAULT=FOO

FOO_DESC=   Option foo support
BAR_DESC=   Feature bar support

# Will add --with-foo / --without-foo
FOO_CONFIGURE_WITH= foo
BAR_RUN_DEPENDS=    bar:bar/bar

.include <bsd.port.mk>
```

Exemplo 45. Verificar `OPTIONS` *Desmacadas*

```
.if ! ${PORT_OPTIONS:MEXAMPLES}
CONFIGURE_ARGS+=--without-examples
.endif
```

O formato acima não é recomendado. O método preferido é usar um configure knob para realmente ativar e desativar o recurso coincidindo com a opção:

```
# Will add --with-examples / --without-examples
EXAMPLES_CONFIGURE_WITH=    examples
```

Exemplo 46. Uso Prático de `OPTIONS`

```
OPTIONS_DEFINE=     EXAMPLES
OPTIONS_DEFAULT=    PGSQL LDAP SSL

OPTIONS_SINGLE=     BACKEND
OPTIONS_SINGLE_BACKEND= MYSQL PGSQL BDB

OPTIONS_MULTI=      AUTH
OPTIONS_MULTI_AUTH= LDAP PAM SSL

EXAMPLES_DESC=      Install extra examples
MYSQL_DESC=     Use MySQL as backend
PGSQL_DESC=     Use PostgreSQL as backend
BDB_DESC=       Use Berkeley DB as backend
LDAP_DESC=      Build with LDAP authentication support
PAM_DESC=       Build with PAM support
SSL_DESC=       Build with OpenSSL support
```

```
# Will add USE_PGSQL=yes
PGSQL_USE=  pgsql=yes
# Will add --enable-postgres / --disable-postgres
PGSQL_CONFIGURE_ENABLE= postgres

ICU_LIB_DEPENDS=    libicuuc.so:devel/icu

# Will add --with-examples / --without-examples
EXAMPLES_CONFIGURE_WITH=    examples

# Check other OPTIONS

.include <bsd.port.mk>
```

Opções Padrão

Essas opções estão sempre ativadas por padrão.

- `DOCS` — build and install documentation.
- `NLS` — Native Language Support.
- `EXAMPLES` — build and install examples.
- `IPV6` — IPv6 protocol support.

Não há necessidade de adicioná-las em `OPTIONS_DEFAULT`. Para ativá-las e mostra-las na caixa de diálogo de seleção de opções, elas devem ser adicionadas em `OPTIONS_DEFINE`.

5.13.2. Feature de Ativação Automática

Ao usar um script configure GNU, fique de olho em quais recursos opcionais são ativados por detecção automática. Desative explicitamente os recursos opcionais que não são necessários, adicionando `--without-xxx` ou `--disable-xxx` em `CONFIGURE_ARGS`.

Exemplo 47. Manipulação Incorreta de uma Opção

```
.if ${PORT_OPTIONS:MFOO}
LIB_DEPENDS+=       libfoo.so:devel/foo
CONFIGURE_ARGS+=    --enable-foo
.endif
```

No exemplo acima, imagine que uma biblioteca libfoo está instalada no sistema. O usuário não quer que este aplicativo use libfoo, então ele desabilitou a opção na caixa de diálogo do `make config`. Mas o script configure do aplicativo detecta a biblioteca presente no sistema e inclui seu suporte no executável resultante. Agora, quando o usuário decide remover libfoo do sistema, o sistema de

ports não protesta (nenhuma dependência de libfoo foi registrada), e então o aplicativo quebra.

Exemplo 48. Manuseio Correto de uma Opção

```
FOO_LIB_DEPENDS=        libfoo.so:devel/foo
# Will add --enable-foo / --disable-foo
FOO_CONFIGURE_ENABLE=   foo
```

Sob algumas circunstâncias, a sintaxe condicional abreviada pode causar problemas com construções complexas. Os erros são geralmente `Malformed conditional`, e uma sintaxe alternativa pode ser usada.

```
.if !empty(VARIABLE:MVALUE)
```

como uma alternativa para

```
.if ${VARIABLE:MVALUE}
```

5.13.3. Assistentes de Opções

Existem algumas macros para ajudar a simplificar valores condicionais que diferem com base nas opções definidas. Para facilitar o acesso, é fornecida uma lista abrangente:

`PLIST_SUB`, `SUB_LIST`

Para geração automática de `%%OPT%%` e `%%NO_OPT%%`, veja `OPTIONS_SUB`.

Para uso mais complexo, veja Substituição de Variáveis Genéricas, `OPT_VARIABLE_` e `OPT_VARIABLE_OFF`.

`CONFIGURE_ARGS`

Para `--enable-x` e `--disable-x`, veja `OPT_CONFIGURE_ENABLE`.

Para `--with-x` e `--without-x`, veja `OPT_CONFIGURE_WITH`.

Para todos os outros casos, veja `OPT_CONFIGURE_ON` e `OPT_CONFIGURE_OFF`.

`CMAKE_ARGS`

Para argumentos que são booleanos (`on`, `off`, `true`, `false`, `0`, `1`) veja `OPT_CMAKE_BOOL` e `OPT_CMAKE_BOOL_OFF`.

Para todos os outros casos, veja `OPT_CMAKE_ON` e `OPT_CMAKE_OFF`.

`MESON_ARGS`

Para argumentos que precisam de `true` ou `false`, veja `OPT_MESON_TRUE` e `OPT_MESON_FALSE`.

Para argumentos que precisam de `yes` ou `no`, use `OPT_MESON_YES` e `OPT_MESON_NO`.

Para argumentos que precisam de `true` ou `false`, veja `OPT_MESON_ENABLED` e `OPT_MESON_DISABLED`.

Para todos os outros casos, use `OPT_MESON_ON` e `OPT_MESON_OFF`.

`QMAKE_ARGS`

Veja `OPT_QMAKE_ON` e `OPT_QMAKE_OFF`.

`USE_*`

Veja `OPT_USE` e `OPT_USE_OFF`.

`*_DEPENDS`

Veja Dependências, `OPT_DEPTYPE_` e `OPT_DEPTYPE_OFF`.

*** (Qualquer variável)**

As variáveis mais usadas possuem assistentes diretos, veja Substituição de Variáveis Genéricas, `OPT_VARIABLE_` e `OPT_VARIABLE_OFF`.

Para qualquer variável sem um assistente específico, veja `OPT_VARS` e `OPT_VARS_OFF`.

Dependências de opções

Quando uma opção precisa de outra opção para funcionar, veja `OPT_IMPLIES`.

Conflitos de opções

Quando uma opção não funciona se outra também estiver ativada, consulte `OPT_PREVENTS` e `OPT_PREVENTS_MSG`.

Targets para Build

Quando uma opção precisa de algum processamento extra, veja Targets Adicionais de Compilação, `target-OPT-on` e `target-OPT-off`.

`OPTIONS_SUB`

Se `OPTIONS_SUB` está definido com `yes` então cada uma das opções adicionadas a `OPTIONS_DEFINE` será adicionada em `PLIST_SUB` e `SUB_LIST`, por exemplo:

```
OPTIONS_DEFINE= OPT1
OPTIONS_SUB=    yes
```

é equivalente a:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
PLIST_SUB+= OPT1="" NO_OPT1="@comment "
```

```
SUB_LIST+=  OPT1="" NO_OPT1="@comment "
.else
PLIST_SUB+= OPT1="@comment " NO_OPT1=""
SUB_LIST+=  OPT1="@comment " NO_OPT1=""
.endif
```

O valor de `OPTIONS_SUB` é ignorado. Definindo-o com qualquer valor irá adicionar entradas `PLIST_SUB` e `SUB_LIST` para *todas* as opções.

`OPT_USE` e `OPT_USE_OFF`

Quando a opção *OPT* é selecionada, para cada par `key=value` em `OPT_USE`, *value* é anexado ao `USE_KEY` correspondente. E se *value* tiver espaços, substitua-os por vírgulas e eles serão alterados de volta para espaços durante o processamento. `OPT_USE_OFF` funciona da mesma maneira, quando `OPT` *não for* selecionada. Por exemplo:

```
OPTIONS_DEFINE= OPT1
OPT1_USES=  xorg
OPT1_USE=   mysql=yes xorg=x11,xextproto,xext,xrandr
OPT1_USE_OFF=   openssl=yes
```

é equivalente a:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
USE_MYSQL=  yes
USES+=      xorg
USE_XORG=   x11 xextproto xext xrandr
.else
USE_OPENSSL=    yes
.endif
```

Assistentes `CONFIGURE_ARGS`

`OPT_CONFIGURE_ENABLE`

Quando a opção *OPT* é selecionada, para cada *valor* em `OPT_CONFIGURE_ENABLE`, `--enable-valor` será anexado a `CONFIGURE_ARGS`. Quando a opção OPT *não for* selecionada, `--disable-valor` será anexado a `CONFIGURE_ARGS`. Um argumento opcional pode ser especificado com um símbolo `=`. Este argumento é apenas anexado na entrada de opção do script configure `--enable-valor`. Por exemplo:

```
OPTIONS_DEFINE= OPT1 OPT2
OPT1_CONFIGURE_ENABLE=  test1 test2
```

```
OPT2_CONFIGURE_ENABLE=  test2=exhaustive
```

é equivalente a:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
CONFIGURE_ARGS+=    --enable-test1 --enable-test2
.else
CONFIGURE_ARGS+=    --disable-test1 --disable-test2
.endif

.if ${PORT_OPTIONS:MOPT2}
CONFIGURE_ARGS+=    --enable-test2=exhaustive
.else
CONFIGURE_ARGS+=    --disable-test2
.endif
```

`OPT_CONFIGURE_WITH`

Quando a opção *OPT* é selecionada, para cada *valor* em `OPT_CONFIGURE_WITH`, `--with-valor` será anexado a `CONFIGURE_ARGS`. Quando a opção OPT_não for_ selecionada, `--without-valor` será anexado a `CONFIGURE_ARGS`. Um argumento opcional pode ser especificado com um símbolo `=`. Este argumento é apenas anexado na entrada de opção do script configure `--with-valor`. Por exemplo:

```
OPTIONS_DEFINE= OPT1 OPT2
OPT1_CONFIGURE_WITH=    test1
OPT2_CONFIGURE_WITH=    test2=exhaustive
```

é equivalente a:

```
OPTIONS_DEFINE= OPT1 OPT2

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
CONFIGURE_ARGS+=    --with-test1
.else
CONFIGURE_ARGS+=    --without-test1
.endif

.if ${PORT_OPTIONS:MOPT2}
CONFIGURE_ARGS+=    --with-test2=exhaustive
.else
CONFIGURE_ARGS+=    --without-test2
```

```
.endif
```

OPT_CONFIGURE_ON e OPT_CONFIGURE_OFF

Quando a opção *OPT* é selecionada, o valor de OPT_CONFIGURE_ON, se definido, é anexado a CONFIGURE_ARGS. OPT_CONFIGURE_OFF funciona da mesma maneira, quando OPT *não for* selecionada. Por exemplo:

```
OPTIONS_DEFINE= OPT1
OPT1_CONFIGURE_ON=  --add-test
OPT1_CONFIGURE_OFF= --no-test
```

é equivalente a:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
CONFIGURE_ARGS+=    --add-test
.else
CONFIGURE_ARGS+=    --no-test
.endif
```

Na maioria das vezes, os assistentes em OPT_CONFIGURE_ENABLE e OPT_CONFIGURE_WITH fornecem uma funcionalidade mais curta e abrangente.

Assistentes CMAKE_ARGS

OPT_CMAKE_ON e OPT_CMAKE_OFF

Quando a opção *OPT* é selecionada, o valor de OPT_CMAKE_ON, se definido, é anexado a CMAKE_ARGS. OPT_CMAKE_OFF funciona da mesma maneira, mas quando OPT *não for* selecionada. Por exemplo:

```
OPTIONS_DEFINE= OPT1
OPT1_CMAKE_ON=  -DTEST:BOOL=true -DDEBUG:BOOL=true
OPT1_CMAKE_OFF= -DOPTIMIZE:BOOL=true
```

é equivalente a:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
CMAKE_ARGS+=    -DTEST:BOOL=true -DDEBUG:BOOL=true
```

```
.else
CMAKE_ARGS+=    -DOPTIMIZE:BOOL=true
.endif
```

Veja OPT_CMAKE_BOOL e OPT_CMAKE_BOOL_OFF para um assistente mais curto quando o valor for booleano.

OPT_CMAKE_BOOL e OPT_CMAKE_BOOL_OFF

Quando a opção *OPT* é selecionada, para cada *valor* em OPT_CMAKE_BOOL, -Dvalor:BOOL=true será anexado a CMAKE_ARGS. Quando a opção OPT_não for_ selecionada, -Dvalor:BOOL=false será anexado a CONFIGURE_ARGS. O OPT_CMAKE_BOOL_OFF é o oposto, -Dvalor:BOOL=false será anexado a CMAKE_ARGS quando a opção é selecionada, e a entrada -Dvalor:BOOL=true quando a opção *não for* selecionada. Por exemplo:

```
OPTIONS_DEFINE= OPT1
OPT1_CMAKE_BOOL=    TEST DEBUG
OPT1_CMAKE_BOOL_OFF=    OPTIMIZE
```

é equivalente a:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
CMAKE_ARGS+=    -DTEST:BOOL=true -DDEBUG:BOOL=true \
        -DOPTIMIZE:BOOL=false
.else
CMAKE_ARGS+=    -DTEST:BOOL=false -DDEBUG:BOOL=false \
        -DOPTIMIZE:BOOL=true
.endif
```

Assistentes MESON_ARGS

OPT_MESON_ON e OPT_MESON_OFF

Quando a opção *OPT* é selecionada, o valor de OPT_MESON_ON, se definido, é anexado a MESON_ARGS. OPT_MESON_OFF funciona da mesma maneira, quando OPT *não for* selecionada. Por exemplo:

```
OPTIONS_DEFINE= OPT1
OPT1_MESON_ON=  -Dopt=1
OPT1_MESON_OFF= -Dopt=2
```

é equivalente a:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
MESON_ARGS+=    -Dopt=1
.else
MESON_ARGS+=    -Dopt=2
.endif
```

`OPT_MESON_TRUE` e `OPT_MESON_FALSE`

Quando a opção *OPT* é selecionada, para cada *valor* em `OPT_MESON_TRUE`, `-Dvalor=true` será anexado a `MESON_ARGS`. Quando a opção OPT_não for_ selecionada, `-Dvalor=false` será anexado a `MESON_ARGS`. O `OPT_MESON_FALSE` é o oposto, a entrada `-Dvalor=false` será anexado a `MESON_ARGS` quando a opção for selecionada e a entrada `-Dvalor=true` quando a opção *não for* selecionada. Por exemplo:

```
OPTIONS_DEFINE= OPT1
OPT1_MESON_TRUE=    test debug
OPT1_MESON_FALSE=   optimize
```

é equivalente a:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
MESON_ARGS+=    -Dtest=true -Ddebug=true \
        -Doptimize=false
.else
MESON_ARGS+=    -Dtest=false -Ddebug=false \
        -Doptimize=true
.endif
```

`OPT_MESON_YES` e `OPT_MESON_NO`

Quando a opção *OPT* é selecionada, para cada *entrada* dentro da variável `OPT_MESON_YES` a entrada `-D__=yes` é anexada a variável `MESON_ARGS`. Quando a opção OPT *não é* selecionada, então a entrada `-D=no` é anexada a variável `MESON_ARGS`. O `OPT_MESON_NO` é o oposto, a entrada `-D=no` é anexada a variável `MESON_ARGS` quando a opção é selecionada e a entrada `-D=yes` quando a opção *não é* selecionada. Por exemplo:

```
OPTIONS_DEFINE= OPT1
OPT1_MESON_YES= test debug
OPT1_MESON_NO=  optimize
```

é equivalente a:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
MESON_ARGS+=    -Dtest=yes -Ddebug=yes \
        -Doptimize=no
.else
MESON_ARGS+=    -Dtest=no -Ddebug=no \
        -Doptimize=yes
.endif
```

`OPT_MESON_ENABLED` e `OPT_MESON_DISABLED`

Quando a opção *OPT* é selecionada, para cada *valor* em `OPT_MESON_ENABLED`, `-Dvalor=enabled` será anexado a `MESON_ARGS`. Quando a opção OPT_não for_ selecionada, `-Dvalor=disabled` será anexado a `MESON_ARGS`. O `OPT_MESON_DISABLED` é o oposto, a entrada `-Dvalor=disabled` será anexado a `MESON_ARGS` quando a opção for selecionada e a entrada `-Dvalor=enabled` quando a opção *não for* selecionada. Por exemplo:

```
OPTIONS_DEFINE= OPT1
OPT1_MESON_ENABLED= test
OPT1_MESON_DISABLED=    debug
```

é equivalente a:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
MESON_ARGS+=    -Dtest=enabled -Ddebug=disabled
.else
MESON_ARGS+=    -Dtest=disabled -Ddebug=enabled
.endif
```

`OPT_QMAKE_ON` e `OPT_QMAKE_OFF`

Quando a opção *OPT* é selecionada, o valor de `OPT_QMAKE_ON`, se definido, é anexado a `QMAKE_ARGS`. `OPT_QMAKE_OFF` funciona da mesma maneira, quando `OPT` *não for* selecionada. Por exemplo:

```
OPTIONS_DEFINE= OPT1
OPT1_QMAKE_ON=  -DTEST:BOOL=true
OPT1_QMAKE_OFF= -DPRODUCTION:BOOL=true
```

é equivalente a:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
QMAKE_ARGS+=    -DTEST:BOOL=true
.else
QMAKE_ARGS+=    -DPRODUCTION:BOOL=true
.endif
```

OPT_IMPLIES

Fornece uma maneira de adicionar dependências entre as opções.

Quando *OPT* for selecionada, todas as opções listadas nesta variável também serão selecionadas. Usando o OPT_CONFIGURE_ENABLE descrito anteriormente para demonstrar:

```
OPTIONS_DEFINE= OPT1 OPT2
OPT1_IMPLIES=   OPT2

OPT1_CONFIGURE_ENABLE=  opt1
OPT2_CONFIGURE_ENABLE=  opt2
```

É equivalente a:

```
OPTIONS_DEFINE= OPT1 OPT2

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
CONFIGURE_ARGS+=    --enable-opt1
.else
CONFIGURE_ARGS+=    --disable-opt1
.endif

.if ${PORT_OPTIONS:MOPT2} || ${PORT_OPTIONS:MOPT1}
CONFIGURE_ARGS+=    --enable-opt2
.else
CONFIGURE_ARGS+=    --disable-opt2
.endif
```

Exemplo 49. Uso Simples de OPT_IMPLIES

Este port tem uma opção X11 e uma opção GNOME que precisa da opção X11 selecionada para poder compilar.

```
OPTIONS_DEFINE= X11 GNOME
OPTIONS_DEFAULT=    X11

X11_USES=   xorg
X11_USE=    xorg=xi,xextproto
GNOME_USE=  gnome=gtk30
GNOME_IMPLIES=  X11
```

`OPT_PREVENTS` e `OPT_PREVENTS_MSG`

Fornece uma maneira de adicionar conflitos entre as opções.

Quando *OPT* for selecionada, todas as opções listadas em `OPT_PREVENTS` devem estar desmarcadas. Se `OPT_PREVENTS_MSG` estiver definido e um conflito for acionado, seu conteúdo será exibido explicando o por que do conflito. Por exemplo:

```
OPTIONS_DEFINE= OPT1 OPT2
OPT1_PREVENTS=  OPT2
OPT1_PREVENTS_MSG=  OPT1 and OPT2 enable conflicting options
```

É aproximadamente equivalente a:

```
OPTIONS_DEFINE= OPT1 OPT2

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT2} && ${PORT_OPTIONS:MOPT1}
BROKEN= Option OPT1 conflicts with OPT2 (select only one)
.endif
```

A única diferença é que o primeiro irá apresentar um erro depois de executar `make config`, sugerindo alterar as opções selecionadas.

Exemplo 50. Uso Simples de `OPT_PREVENTS`

Este port tem as opções `X509` e `SCTP`. Ambas as opções adicionam patches, mas os patches entram em conflito uns com os outros, então eles não podem ser selecionados ao mesmo tempo.

```
OPTIONS_DEFINE= X509 SCTP

SCTP_PATCHFILES=    ${PORTNAME}-6.8p1-sctp-2573.patch.gz:-p1
SCTP_CONFIGURE_WITH=    sctp

X509_PATCH_SITES=   http://www.roumenpetrov.info/openssh/x509/:x509
X509_PATCHFILES=    ${PORTNAME}-7.0p1+x509-8.5.diff.gz:-p1:x509
```

```
X509_PREVENTS=      SCTP
X509_PREVENTS_MSG=  X509 and SCTP patches conflict
```

OPT_VARS e OPT_VARS_OFF

Fornece uma maneira genérica de definir e acrescentar valores em variáveis.

Antes de usar OPT_VARS e OPT_VARS_OFF, veja se já não existe um assistente mais específico disponível em Substituição de Variáveis Genéricas, OPT_VARIABLE_ e OPT_VARIABLE_OFF.

Quando a opção *OPT* está selecionada e OPT_VARS definido, os pares chave=valor e chave+=valor são avaliados a partir da variável OPT_VARS. Um = sobrescreve o valor existente da CHAVE, um += acrescenta o valor a chave. OPT_VARS_OFF funciona da mesma maneira, quando a opção OPT *não for* selecionada.

```
OPTIONS_DEFINE= OPT1 OPT2 OPT3
OPT1_VARS=  also_build+=bin1
OPT2_VARS=  also_build+=bin2
OPT3_VARS=  bin3_build=yes
OPT3_VARS_OFF=  bin3_build=no

MAKE_ARGS=  ALSO_BUILD="${ALSO_BUILD}" BIN3_BUILD="${BIN3_BUILD}"
```

é equivalente a:

```
OPTIONS_DEFINE= OPT1 OPT2

MAKE_ARGS=  ALSO_BUILD="${ALSO_BUILD}" BIN3_BUILD="${BIN3_BUILD}"

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
ALSO_BUILD+=    bin1
.endif

.if ${PORT_OPTIONS:MOPT2}
ALSO_BUILD+=    bin2
.endif

.if ${PORT_OPTIONS:MOPT2}
BIN3_BUILD= yes
.else
BIN3_BUILD= no
.endif
```

Valores contendo espaços em branco devem ser colocados entre aspas:

```
OPT_VARS=   foo="bar baz"
```

Isso se deve ao jeito que a variável de expansão make(1) lida com espaço em branco. Quando a opção `OPT_VARS=foo=bar baz` é expandida, a variável acaba contendo duas strings, `foo=bar` e `baz`. Mas quem está submetendo o código provavelmente pretendia que houvesse apenas uma string, `foo=bar baz`. Inserir o valor entre aspas impede que o espaço em branco seja usado como um delimitador.

Além disso, *não* adicione espaços extras após o símbolo `var=` e antes do valor, pois assim também seria dividido o valor em duas strings. *Isso não irá funcionar*:

```
OPT_VARS=   foo=    bar
```

Dependências, `OPT_DEPTYPE_` e `OPT_DEPTYPE_OFF`

Para qualquer um desses tipos de dependência:

- `PKG_DEPENDS`
- `EXTRACT_DEPENDS`
- `PATCH_DEPENDS`
- `FETCH_DEPENDS`
- `BUILD_DEPENDS`
- `LIB_DEPENDS`
- `RUN_DEPENDS`

Quando opção *OPT* é selecionada, o valor de `OPT_DEPTYPE`, se definido, é anexado a `DEPTYPE`. `OPT_DEPTYPE_OFF` funciona da mesma forma, quando `OPT` *não for* selecionada. Por exemplo:

```
OPTIONS_DEFINE= OPT1
OPT1_LIB_DEPENDS=   liba.so:devel/a
OPT1_LIB_DEPENDS_OFF=   libb.so:devel/b
```

é equivalente a:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
LIB_DEPENDS+=   liba.so:devel/a
```

```
.else
LIB_DEPENDS+=   libb.so:devel/b
.endif
```

Substituição de Variáveis Genéricas, `OPT_VARIABLE_` e `OPT_VARIABLE_OFF`

Para qualquer uma destas variáveis:

- `ALL_TARGET`
- `BINARY_ALIAS`
- `BROKEN`
- `CATEGORIES`
- `CFLAGS`
- `CONFIGURE_ENV`
- `CONFLICTS`
- `CONFLICTS_BUILD`
- `CONFLICTS_INSTALL`
- `CPPFLAGS`
- `CXXFLAGS`
- `DESKTOP_ENTRIES`
- `DISTFILES`
- `EXTRACT_ONLY`
- `EXTRA_PATCHES`
- `GH_ACCOUNT`
- `GH_PROJECT`
- `GH_SUBDIR`
- `GH_TAGNAME`
- `GH_TUPLE`
- `GL_ACCOUNT`
- `GL_COMMIT`
- `GL_PROJECT`
- `GL_SITE`
- `GL_SUBDIR`
- `GL_TUPLE`
- `IGNORE`
- `INFO`
- `INSTALL_TARGET`

- LDFLAGS
- LIBS
- MAKE_ARGS
- MAKE_ENV
- MASTER_SITES
- PATCHFILES
- PATCH_SITES
- PLIST_DIRS
- PLIST_FILES
- PLIST_SUB
- PORTDOCS
- PORTEXAMPLES
- SUB_FILES
- SUB_LIST
- TEST_TARGET
- USES

Quando a opção *OPT* é selecionada, o valor da variável OPT_ABOVEVARIABLE, se definido, é anexado a ABOVEVARIABLE. OPT_ABOVEVARIABLE_OFF funciona da mesma maneira, quando OPT *não for* selecionada. Por exemplo:

```
OPTIONS_DEFINE= OPT1
OPT1_USES=  gmake
OPT1_CFLAGS_OFF=    -DTEST
```

é equivalente a:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
USES+=      gmake
.else
CFLAGS+=    -DTEST
.endif
```

Algumas variáveis não estão nesta lista, em particular PKGNAMEPREFIX e PKGNAMESUFFIX. Isso é intencional. Um port *não deve* mudar seu nome quando alguma de suas opções forem alteradas.

Algumas dessas variáveis, pelo menos `ALL_TARGET`, `DISTFILES` e `INSTALL_TARGET`, tem seus valores padrão definidos *depois* das opções serem processadas.

Com estas linhas no Makefile:

```
ALL_TARGET= all

DOCS_ALL_TARGET=    doc
```

Se a opção `DOCS` estiver ativada, `ALL_TARGET` terá o valor `all doc`; se a opção estiver desativada, ela terá o valor `all`.

Com apenas a linha do assistente de opções no Makefile:

```
DOCS_ALL_TARGET=    doc
```

Se a opção `DOCS` estiver ativada, `ALL_TARGET` terá o valor `doc`; se a opção estiver desativada, ela terá o valor `all`.

Targets Adicionais de Compilação, `target-OPT-on` e `target-OPT-off`

Estes targets de Makefile podem aceitar targets extras de compilação:

- `pre-fetch`
- `do-fetch`
- `post-fetch`
- `pre-extract`
- `do-extract`
- `post-extract`
- `pre-patch`
- `do-patch`
- `post-patch`
- `pre-configure`
- `do-configure`
- `post-configure`
- `pre-build`
- `do-build`
- `post-build`
- `pre-install`
- `do-install`

- post-install
- post-stage
- pre-package
- do-package
- post-package

Quando a opção *OPT* é selecionada, o target TARGET-OPT-on, se definido, é executado após TARGET. TARGET-OPT-off funciona da mesma maneira, quando OPT *não for* selecionada. Por exemplo:

```
OPTIONS_DEFINE= OPT1

post-patch-OPT1-on:
    @${REINPLACE_CMD} -e '/opt1/s|/usr/bin/|${EXAMPLESDIR}/|' ${WRKSRC}/Makefile

post-patch-OPT1-off:
    @${REINPLACE_CMD} -e '/opt1/s|/usr/bin/|${PREFIX}/bin/|' ${WRKSRC}/Makefile
```

é equivalente a:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

post-patch:
.if ${PORT_OPTIONS:MOPT1}
    @${REINPLACE_CMD} -e '/opt1/s|/usr/bin/|${EXAMPLESDIR}/|' ${WRKSRC}/Makefile
.else
    @${REINPLACE_CMD} -e '/opt1/s|/usr/bin/|${PREFIX}/bin/|' ${WRKSRC}/Makefile
.endif
```

5.14. Especificando o Diretório de Trabalho

Cada port é extraído em um diretório de trabalho, que deve ter permissão de escrita. O sistema de ports tem por padrão os DISTFILES descompactado em um diretório chamado ${DISTNAME}. Em outras palavras, se o Makefile tem:

```
PORTNAME=   foo
DISTVERSION=    1.0
```

então os arquivos de distribuição do port contêm um diretório de nível superior, foo-1.0, e o resto dos arquivos estão localizados nesse diretório.

Diversas variáveis podem ser substituídas se não for esse o caso.

5.14.1. WRKSRC

A variável lista o nome do diretório que é criado quando os distfiles do aplicativo são extraídos. Se o exemplo anterior for extraído em um diretório chamado foo (e não foo-1.0) escreva:

```
WRKSRC= ${WRKDIR}/foo
```

ou possivelmente

```
WRKSRC= ${WRKDIR}/${PORTNAME}
```

5.14.2. WRKSRC_SUBDIR

Se o código fonte necessário para o port estiver em um subdiretório do arquivo de distribuição extraído, defina WRKSRC_SUBDIR para esse diretório.

```
WRKSRC_SUBDIR=  src
```

5.14.3. NO_WRKSUBDIR

Se o port não extrair para nenhum subdiretório, então configure NO_WRKSUBDIR para indicar isso.

```
NO_WRKSUBDIR=   yes
```

> Porque WRKDIR é o único diretório que deve ter permissão de escrita durante a compilação e é usado para armazenar muitos arquivos que registram o status da compilação, a extração do port será forçada para um subdiretório.

5.15. Manipulando Conflitos

Existem três variáveis diferentes para registrar um conflito entre pacotes e ports: CONFLICTS, CONFLICTS_INSTALL e CONFLICTS_BUILD.

> As variáveis de conflito definem automaticamente a variável IGNORE, que é mais amplamente documentada em Marcando um Port não Instalável com a variável BROKEN FORBIDDEN ou IGNORE.

Ao remover um dos vários ports conflitados, é aconselhável reter CONFLICTS nos outros ports por alguns meses para atender usuários que apenas fazem atualizações de vez em quando.

CONFLICTS_INSTALL

Se o pacote não puder coexistir com outros pacotes (devido a conflitos de arquivos, incompatibilidades de tempo de execução, etc.). A checagem CONFLICTS_INSTALL é feita após o

estágio de compilação e antes do estágio de instalação.

`CONFLICTS_BUILD`

Se o port não puder ser compilado quando outros ports específicos já estiverem instalados. Conflitos de compilação não serão registrados no pacote final.

`CONFLICTS`

Se o port não puder ser compilado quando um certo port estiver instalado e o pacote final não puder coexistir com o outro pacote. A checagem `CONFLICTS` é feita antes do estágio de compilação e antes do estágio de instalação.

O conteúdo mais comum de uma dessas variáveis é o pacote base de outro port. O pacote base é o nome do pacote sem a versão, ele pode ser obtido executando `make -V PKGBASE`.

Exemplo 51. Uso básico de `CONFLICTS_*`

dns/bind99 não pode ser instalado se dns/bind910 está presente porque eles instalam os mesmos arquivos. Primeiro, reúna o pacote base para usar:

```
% make -C dns/bind99 -V PKGBASE
bind99
% make -C dns/bind910 -V PKGBASE
bind910
```

Então adicione ao Makefile do dns/bind99:

```
CONFLICTS_INSTALL=  bind910
```

E adicione ao Makefile do dns/bind910:

```
CONFLICTS_INSTALL=  bind99
```

Às vezes, apenas uma versão de outro port é incompatível, neste caso, use o nome completo do pacote, com a versão, e use shell globs, como `*` e `?` para garantir que todas as versões possíveis sejam correspondidas.

Exemplo 52. Usando `CONFLICTS_*` Com Globs.

Nas versões 2.0 até 2.4.1_2, deskutils/gnotime instalava uma versão integrada de databases/qof.

Para refletir este passado, o Makefile do database/qof contém:

```
CONFLICTS_INSTALL=  gnotime-2.[0-3]* \
            gnotime-2.4.0* gnotime-2.4.1 \
            gnotime-2.4.1_[12]
```

As primeira entrada corresponde as versões `2.0` até `2.3`, a segunda corresponde todas as revisões de `2.4.0`, a terceira corresponde a versão exata `2.4.1`, e a última corresponde a primeira e segunda revisão da versão `2.4.1`.

deskutils/gnotime não possui nenhuma linha de conflitos porque sua versão atual não conflita com mais nada.

5.16. Instalando Arquivos

O estágio `install` é muito importante para o usuário final porque ele adiciona arquivos ao sistema. Todos os comandos adicionais de estágios `*-install` dos Makefile's de port devem ser mostrados na tela. *Não* silencie esses comandos com `@` ou `.SILENT`.

5.16.1. Macros `INSTALL_*`

Use as macros fornecidas em bsd.port.mk para garantir a propriedade correta dos arquivos nos targets `*-install` do port. Defina a propriedade diretamente em pkg-plist com as entradas correspondentes, como `@(owner,group,)`, `@owner owner`, e `@group group`. Esses operadores funcionam até serem substituídos, ou até o final do pkg-plist, lembre-se de redefini-los depois que eles não forem mais necessários. O valor de propriedade padrão é `root:wheel`. Veja Keywords Básicas para maiores informações.

- `INSTALL_PROGRAM` é um comando para instalar executáveis binários.
- `INSTALL_SCRIPT` é um comando para instalar scripts executáveis.
- `INSTALL_LIB` é um comando para instalar bibliotecas compartilhadas (mas não bibliotecas estáticas).
- `INSTALL_KLD` é um comando para instalar módulos carregáveis do kernel. Algumas arquiteturas não gostam de ter os módulos otimizados (stripped), então use este comando em vez de `INSTALL_PROGRAM`.
- `INSTALL_DATA` é um comando para instalar dados compartilháveis, incluindo bibliotecas estáticas.
- `INSTALL_MAN` é um comando para instalar manpages e outras documentações (ele não realiza nenhuma compactação).

Estas variáveis parametrizam o comando install(1) com as flags apropriadas para cada situação.

Não use `INSTALL_LIB` para instalar bibliotecas estáticas, porque otimiza-las (strip) torna-as sem utilidade. Use `INSTALL_DATA` neste caso.

5.16.2. Otimizando (Stripping) Binários e Bibliotecas Compartilhadas

Os binários instalados devem ser otimizados (stripped). Não otimize (strip) os binários manualmente, a menos que seja absolutamente necessário. A macro `INSTALL_PROGRAM` instala e otimiza (strip) o binário ao mesmo tempo. A macro `INSTALL_LIB` faz o mesmo com as bibliotecas compartilhadas.

Quando um arquivo deve ser otimizado (stripped), mas as macros INSTALL_PROGRAM e INSTALL_LIB não são desejadas, ${STRIP_CMD} otimiza (strips) o programa ou a biblioteca compartilhada. Isso geralmente é feito no target post-install. Por exemplo:

```
post-install:
    ${STRIP_CMD} ${STAGEDIR}${PREFIX}/bin/xdl
```

Quando vários arquivos precisam ser otimizados (stripped):

```
post-install:
.for l in geometry media body track world
    ${STRIP_CMD} ${STAGEDIR}${PREFIX}/lib/lib${PORTNAME}-${l}.so.0
.endfor
```

Use file(1) em um arquivo para determinar se ele foi otimizado (stripped). Binários são relatados por file(1) como stripped ou not stripped. Além disso,strip(1) irá detectar programas que já foram otimizados (stripped) e retornar o comando sem erros.

Quando WITH_DEBUG estiver definido, os arquivos elf *não devem* ser otimizados (stripped).

As variáveis (STRIP_CMD, INSTALL_PROGRAM, INSTALL_LIB, ...) e USES fornecidas pelo framework lidam com isso automaticamente.

Alguns softwares, adicionam -s em seus LDFLAGS, neste caso, ou remova o -s se WITH_DEBUG estiver definido, ou remova o incondicionalmente e use STRIP_CMD em post-install.

5.16.3. Instalando uma Árvore Inteira de Arquivos

Às vezes, um grande número de arquivos devem ser instalados preservando sua organização hierárquica. Por exemplo, copiando de uma árvore de diretórios inteira do WRKSRC para um diretório de destino sob PREFIX. Observe que PREFIX, EXEMPLESDIR, DATADIR e outras variáveis de caminho sempre devem ser precedidas por STAGEDIR para respeitar o staging (ver Staging).

Existem duas macros para essa situação. A vantagem de usar essas macros em vez de cp é que elas garantem a propriedade e permissão adequada dos arquivos nos arquivos de destino. A primeira macro, COPYTREE_BIN, irá definir todos os arquivos instalados como sendo executáveis, sendo assim, adequado para instalações em PREFIX/bin. A segunda macro,COPYTREE_SHARE, não define permissões de execução nos arquivos e, portanto, é adequado para instalar arquivos sob o destino PREFIX/share.

```
post-install:
    ${MKDIR} ${STAGEDIR}${EXAMPLESDIR}
    (cd ${WRKSRC}/examples && ${COPYTREE_SHARE} .${STAGEDIR}${EXAMPLESDIR})
```

Este exemplo irá instalar o conteúdo do diretório exemples do distfile do fornecedor para o local de exemplos apropriado do port.

```
post-install:
    ${MKDIR} ${STAGEDIR}${DATADIR}/summer
    (cd ${WRKSRC}/temperatures && ${COPYTREE_SHARE} "June July August"
${STAGEDIR}${DATADIR}/summer)
```

E este exemplo irá instalar os dados dos meses de verão no subdiretório summer de um DATADIR.

Argumentos `find` adicionais podem ser passados através do terceiro argumento para `COPYTREE_*`. Por exemplo, para instalar todos os arquivos do primeiro exemplo, exceto Makefiles, é possível usar esses comandos.

```
post-install:
    ${MKDIR} ${STAGEDIR}${EXAMPLESDIR}
    (cd ${WRKSRC}/examples && \
    ${COPYTREE_SHARE} .${STAGEDIR}${EXAMPLESDIR} "! -name Makefile")
```

Essas macros não adicionam os arquivos instalados em pkg-plist. Eles devem ser adicionados manualmente. Para documentação opcional (`PORTDOCS`, veja Instalar Documentação Adicional) e exemplos (`PORTEXAMPLES`), os prefixos `%%PORTDOCS%%` ou `%%PORTEXAMPLES%%` devem ser prefixados no pkg-plist.

5.16.4. Instalar Documentação Adicional

Se o software tiver alguma documentação diferente do manual padrão e páginas de informações úteis para o usuário, instale-os em `DOCSDIR`. Isso pode ser feito como no item anterior, no target `post-install`.

Crie um novo diretório para o port. O nome do diretório é `DOCSDIR`. Isso geralmente é igual a `PORTNAME`. No entanto, se o usuário desejar que versões diferentes do port sejam instaladas ao mesmo tempo, `PKGNAME` pode ser usado.

Já que apenas os arquivos listados no pkg-plist são instalados, é seguro sempre instalar documentações no `STAGEDIR` (veja Staging). Por isso, blocos `.if` são necessários apenas quando os arquivos forem grandes o suficiente para causarem sobrecarga significativa de I/O.

```
post-install:
    ${MKDIR} ${STAGEDIR}${DOCSDIR}
    ${INSTALL_MAN} ${WRKSRC}/docs/xvdocs.ps ${STAGEDIR}${DOCSDIR}
```

Por outro lado, se houver uma opção DOCS no port, instale a documentação em um taget `post-install-DOCS-on`. Esses targets são descritos em Targets Adicionais de Compilação, `target-OPT-on` e `target-OPT-off`.

Aqui estão algumas variáveis úteis e como elas são expandidas por padrão quando usadas no

Makefile:

- DATADIR é expandido para PREFIX/shared/PORTNAME.
- DATADIR_REL é expandido para share/PORTNAME.
- DOCSDIR é expandido para PREFIX/shared/doc/PORTNAME.
- DOCSDIR_REL é expandido para share/doc/PORTNAME.
- EXEMPLESDIR é expandido para PREFIX/shared/examples/PORTNAME.
- EXAMPLESDIR_REL é expandido para share/examples/PORTNAME.

> A opção DOCS controla apenas a documentação adicional instalada em DOCSDIR. Não se aplica a páginas de manual e páginas de informações padrão. Arquivos instalados em EXEMPLESDIR são controlados pela opção EXEMPLES.

Essas variáveis são exportadas para PLIST_SUB. Quando possível, seus valores aparecerão como nomes de caminho relativos ao PREFIX. Isso é, por padrão share/doc/PORTNAME será substituído por %%DOCSDIR%% na lista de empacotamento e assim por diante. (Saiba mais sobre substituições pkg-plistaqui.)

Todos os arquivos e diretórios de documentação instalados condicionalmente são incluídos no pkg-plist com o prefixo %%PORTDOCS%%, por exemplo:

```
%%PORTDOCS%%%DOCSDIR%%/AUTHORS
%%PORTDOCS%%%DOCSDIR%%/CONTACT
```

Como uma alternativa para listar os arquivos de documentação em pkg-plist, um port pode definir a variável PORTDOCS com uma lista de nomes de arquivo e padrões shell glob para adicionar à lista de empacotamento final. Os nomes serão relativos a DOCSDIR. Portanto, um port que utiliza PORTDOCS e usa um local não padrão para sua documentação, deve definir DOCSDIR adequadamente. Se um diretório estiver listado em PORTDOCS ou ser correspondido por um padrão glob dessa variável, toda a sub árvore de arquivos e diretórios contidos serão registrados na lista final de empacotamento. Se a opção DOCS estiver desmarcada, os arquivos e diretórios listados em PORTDOCS não serão instalados ou adicionados à lista de empacotamento do port. A instalação da documentação em PORTDOCS como mostrado acima fica a cargo do port. Um exemplo típico de utilização PORTDOCS:

```
PORTDOCS=   README.* ChangeLog docs/*
```

> O equivalente de PORTDOCS para arquivos instalados em DATADIR e EXEMPLESDIR são PORTDATA e PORTEXAMPLES, respectivamente.
>
> O conteúdo de pkg-message é exibido na instalação. Veja a seção sobre o uso do pkg-message para mais detalhes. pkg-message não precisa ser adicionado ao pkg-plist.

5.16.5. Subdiretórios Sob PREFIX

Tente deixar o port colocar os arquivos nos subdiretórios corretos de PREFIX. Alguns ports juntam tudo e colocam os arquivos em um subdiretório com o nome do port, o que é incorreto. Além disso, muitos ports colocam todos arquivos, exceto binários, arquivos header e páginas de manual, em um subdiretório de lib, o que não funciona bem com o paradigma BSD. Muitos dos arquivos devem ser movidos para um desses diretórios: etc(setup/arquivos de configuração), libexec (executáveis iniciados internamente), sbin (executáveis para super-usuários/gerentes), info (documentação para o navegador de informações) ou share (arquivos independentes de arquitetura). Veja hier(7) para detalhes; as regras que regem /usr praticamente se aplicam a /usr/local também. A exceção são os ports que lidam com "notícias" USENET. Eles podem usar PREFIX/news como um destino para seus arquivos.

5.17. Use BINARY_ALIAS para Renomear Comandos Em Vez de Aplicar Patch na Compilação

Quando BINARY_ALIAS é definido, ele criará links simbólicos dos comandos fornecidos, em um diretório que será prefixado para o PATH.

Use-o para substituir comandos codificados na fase de compilação sem ter aplicar nenhum patch nos arquivos de compilação.

Exemplo 53. Usando BINARY_ALIAS para Deixar gsed Disponível como sed

Alguns ports esperam que o sed se comporte como o GNU sed e utilizam recursos que o sed(1) não possui. GNU sed está disponível em textproc/gsed no FreeBSD.

Use BINARY_ALIAS para substituir sed com gsed durante a compilação:

```
BUILD_DEPENDS=  gsed:textproc/gsed
...
BINARY_ALIAS=   sed=gsed
```

Exemplo 54. Usando BINARY_ALIAS Para Fornecer Aliases para Comandos python3 Codificado

Um port que possui uma referência codificada para python3 em seus scripts de compilação precisará ter ele disponível no PATH em tempo de compilação. Use BINARY_ALIAS para criar um alias que aponte para o binário certo do Python 3:

```
USES=   python:3.4+,build
...
BINARY_ALIAS=   python3=${PYTHON_CMD}
```

Veja Usando Python para mais informações sobre USES=python.

Aliases binários são criados após as dependências fornecidas via `BUILD_DEPENDS` e `LIB_DEPENDS` serem processadas e antes do target `configure`. Isso leva a várias limitações. Por exemplo, os programas instalados via `TEST_DEPENDS` não podem ser usados para criar um alias binário, pois as dependências de teste especificadas desta forma são processadas após a criação dos aliases binários.

Capítulo 6. Considerações Especiais

Esta seção explica as coisas mais comuns a se considerar ao criar um port.

6.1. Staging

bsd.port.mk espera que os ports trabalhem com um "stage directory". Isso significa que um port não deve instalar arquivos diretamente nos diretórios de destino regulares (isto é, sob o `PREFIX`, por exemplo), mas em um diretório separado a partir do qual o pacote será construído. Em muitos casos, isso não requer privilégios de root, tornando possível criar pacotes como um usuário não privilegiado. Com o staging, o port é compilado e instalado no diretório sde estágio, `STAGEDIR`. Um pacote é criado a partir do diretório de estágio e, em seguida, instalado no sistema. As ferramentas Automake referem-se a este conceito como `DESTDIR`, mas no FreeBSD, `DESTDIR` tem um significado diferente (veja `PREFIX` e `DESTDIR`).

> Nenhum port *realmente* precisa de root. Ele pode ser evitado principalmente usando `USES=uidfix`. Se o port ainda executa comandos como chown(8), chgrp(1) ou força o proprietário ou grupo com install(1) então use `USES=fakeroot` para enganar essas chamadas. Algumas modificações no Makefile do port serão necessárias.

Os meta ports, ou ports que não instalam arquivos por si mesmos e apenas dependem de outros ports, devem evitar extrair desnecessariamente mtree(8) para o diretório de estágio. Este é o layout básico do diretório do pacote, e estes diretórios vazios serão vistos como órfãos. Para prevenir extração do mtree(8), adicione esta linha:

```
NO_MTREE=   yes
```

> Metaports devem usar `USES=metaport`. Ele configura padrões para ports que não baixam, criam ou instalam nada.

Staging é ativado pré-fixando a variável `STAGEDIR` para caminhos usados nos targets `pre-install`, `do-install` e `post-install` (veja os exemplos no livro). Normalmente, isso inclui as variáveis `PREFIX`, `ETCDIR`, `DATADIR`, `EXEMPLESDIR`, `MANPREFIX`, `DOCSDIR`, e assim por diante. Os diretórios devem ser criados como parte do target `post-install`. Evite usar caminhos absolutos sempre que possível.

> Ports que instalam módulos do kernel devem preceder a variável `STAGEDIR` em seus destinos, padrão /boot/modules.

6.1.1. Lidando com Links Simbólicos

Ao criar um link simbólico, os links relativos são fortemente recomendados. Use `${RLN}` para criar links simbólicos relativos. Ele usa o install(1) por baixo dos panos para descobrir automaticamente o link relativo a ser criado.

Exemplo 55. Crie Links Simbólicos Relativos Automaticamente

${RLN} usa o recurso simbólico relativo do install(1) que libera o mantenedor do port de computar o caminho relativo.

```
${RLN} ${STAGEDIR}${PREFIX}/lib/libfoo.so.42 ${STAGEDIR}${PREFIX}/lib/libfoo.so
${RLN} ${STAGEDIR}${PREFIX}/libexec/foo/bar ${STAGEDIR}${PREFIX}/bin/bar
${RLN} ${STAGEDIR}/var/cache/foo ${STAGEDIR}${PREFIX}/share/foo
```

Irá gerar:

```
% ls -lF ${STAGEDIR}${PREFIX}/lib
lrwxr-xr-x  1 nobody  nobody    181 Aug  3 11:27 libfoo.so@ -> libfoo.so.42
-rwxr-xr-x  1 nobody  nobody     15 Aug  3 11:24 libfoo.so.42*
% ls -lF ${STAGEDIR}${PREFIX}/bin
lrwxr-xr-x  1 nobody  nobody    181 Aug  3 11:27 bar@ -> ../libexec/foo/bar
% ls -lF ${STAGEDIRDIR}${PREFIX}/share
lrwxr-xr-x  1 nobody  nobody    181 Aug  3 11:27 foo@ -> ../../../var/cache/foo
```

6.2. Bibliotecas Empacotadas (Bundled)

Esta seção explica porque as dependências agrupadas(bundled) são consideradas ruins e o que fazer com elas.

6.2.1. Por Que as Bibliotecas Agrupadas(Bundled) São Ruins

Alguns softwares requerem que o mantenedor do port localize bibliotecas de terceiros e adicione as dependências necessárias ao port. Outros softwares agrupam todas as bibliotecas necessárias no arquivo de distribuição. A segunda abordagem parece mais fácil no começo, mas há algumas desvantagens sérias:

Esta lista é vagamente baseada nas wikis Fedora e Gentoo, ambas licenciadas sob CC-BY-SA 3.0.

Segurança

Se vulnerabilidades forem encontradas na biblioteca e arrumadas no upstream, elas podem não ser consertadas na biblioteca empacotada com o port. Uma razão pode ser que o autor não esteja ciente do problema. Isto significa que o mantenedor do port deve consertá-las, ou atualizar para uma versão não vulnerável e enviar um patch para o autor. Isso tudo leva tempo, o que resulta em software vulnerável por mais tempo do que o necessário. Isso, por sua vez, torna mais difícil coordenar uma correção sem vazamento desnecessário de informações sobre a vulnerabilidade.

Bugs

Esse problema é semelhante ao problema de segurança no último parágrafo, mas geralmente menos grave.

Forking

É mais fácil para o autor criar um fork da biblioteca depois que ela é empacotada. Embora seja conveniente à primeira vista, isso significa que o código diverge do upstream, dificultando o tratamento da segurança ou outros problemas com o software. A razão para isso é que o patching se torna mais difícil.

Outro problema de forking é que, como o código diverge do upstream, os bugs são resolvidos repetidamente em vez de apenas uma vez em um local central. Isso, em primeiro lugar, anula a ideia de software de código aberto.

Colisão de símbolo

Quando uma biblioteca é instalada no sistema, ela pode colidir com a versão empacotada. Isso pode causar erros imediatos no tempo de compilação ou link. Também pode causar erros ao executar o programa, o que pode ser mais difícil de rastrear. O último problema poderia ser causado porque as versões das duas bibliotecas são incompatíveis.

Licenciamento

Ao agrupar projetos de diferentes fontes, os problemas de licença podem surgir com mais facilidade, especialmente quando as licenças são incompatíveis.

Desperdício de recursos

Bibliotecas empacotadas desperdiçam recursos em vários níveis. Demora mais para compilar o aplicativo real, especialmente se essas bibliotecas já estiverem presentes no sistema. Em tempo de execução, elas podem ocupar memória desnecessária quando a biblioteca do sistema já está carregada por um programa e a biblioteca agrupada é carregada por outro programa.

Desperdício de esforço

Quando uma biblioteca precisa de patches para o FreeBSD, esses patches precisam ser duplicados novamente na biblioteca. Isso desperdiça tempo do desenvolvedor porque os patches podem não ser aplicados de forma limpa. Também pode ser difícil perceber que estes patches são necessários em primeiro lugar.

6.2.2. O Que Fazer em Relação às Bibliotecas Agrupadas

Sempre que possível, use a versão separada da biblioteca adicionando um `LIB_DEPENDS` para o port. Se esse port ainda não existir, considere criá-lo.

Use bibliotecas agrupadas somente se o upstream tiver um bom histórico de segurança e se o uso de versões não agrupadas originarem patches excessivamente complexos.

> Em alguns casos muito especiais, por exemplo, emuladores, como o Wine, um port tem que agrupar bibliotecas, porque elas estão em uma arquitetura diferente ou foram modificadas para se adequarem ao uso do software. Nesse caso, essas bibliotecas não devem ser expostas a outros ports para vinculação. Adicione `BUNDLE_LIBS=yes` no Makefile do port. Isso vai dizer ao pkg(8) para não computar as bibliotecas fornecidas. Pergunte sempre à equipe de gerenciamento do ports portmgr@FreeBSD.org antes de adicionar isso a um port.

6.3. Bibliotecas Compartilhadas

Se o port instalar uma ou mais bibliotecas compartilhadas, defina a variável `USE_LDCONFIG` para o make , a qual irá instruir o bsd.port.mk para executar o `${LDCONFIG} -m` no diretório onde a nova biblioteca está instalada (geralmente em PREFIX/lib) durante o target `post-install` para registrá-la no cache da biblioteca compartilhada. Esta variável, quando definida, também facilitará a adição do par `@exec /sbin/ldconfig -m` e `@unexec /sbin/ldconfig -R` no pkg-plist, para que o usuário que instalou o pacote possa começar a usar a biblioteca compartilhada imediatamente e para que a desinstalação não faça com que o sistema acredite que a biblioteca ainda está lá.

```
USE_LDCONFIG=   yes
```

O diretório padrão pode ser substituído configurando a variável `USE_LDCONFIG` para uma lista de diretórios nos quais as bibliotecas compartilhadas devem ser instaladas. Por exemplo, se o port instalar bibliotecas compartilhadas em PREFIX/lib/foo e PREFIXO/lib/bar utilize isso no Makefile:

```
USE_LDCONFIG=   ${PREFIX}/lib/foo ${PREFIX}/lib/bar
```

Por favor, verifique novamente, muitas vezes isso não é necessário ou é algo que pode ser evitado através do uso da opção `-rpath` ou da configuração da variável `LD_RUN_PATH` durante a fase de vinculação (consulte lang/mosml para um exemplo), ou através de um shell-wrapper que defina o `LD_LIBRARY_PATH` antes de executar o binário, como por exemplo o www/seamonkey faz.

Ao instalar bibliotecas de 32 bits em um sistema de 64 bits, use `USE_LDCONFIG32` como alternativa.

Se o software usa o autotools, e especificamente, o `libtool`, adicione `USES=libtool`.

Quando o número da versão da biblioteca principal aumenta na atualização para a nova versão do port, todos os outros ports que se vinculam à biblioteca afetada devem ter seu `PORTREVISION` incrementado, para forçar a recompilação com a nova versão da biblioteca.

6.4. Ports com Restrições de Distribuição ou Preocupações Legais

As licenças variam e algumas delas impõem restrições sobre como o aplicativo pode ser empacotado, se pode ser vendido com fins lucrativos e assim por diante.

> É de responsabilidade de um mantenedor de um port ler os termos de licenciamento do software e certificar-se de que o projeto do FreeBSD não será responsabilizado por violá-los, redistribuindo o código fonte ou os binários compilados via FTP/HTTP ou CD-ROM. Se estiver em dúvida, entre em contato com a Lista de discussão de ports do FreeBSD.

Em situações como esta, as variáveis descritas nas próximas seções podem ser definidas.

6.4.1. NO_PACKAGE

Esta variável indica que não podemos gerar um pacote binário da aplicação. Por exemplo, a licença pode proibir a redistribuição binária, ou pode proibir a distribuição de pacotes criados a partir de código adaptado.

No entanto, o DISTFILES do port pode ser livremente espelhado no FTP/HTTP. Eles também podem ser distribuídos em um CD-ROM (ou mídia similar), a menos que a variável NO_CDROM esteja definida também.

Se o pacote binário geralmente não é útil, e o aplicativo sempre deve ser compilado a partir do código-fonte, use o NO_PACKAGE. Por exemplo, se o aplicativo tiver informações de configuração específicas do site codificadas nele em tempo de compilação, defina o NO_PACKAGE.

Defina a variável NO_PACKAGE para uma string descrevendo o motivo pelo qual o pacote não pode ser gerado.

6.4.2. NO_CDROM

Esta variável sozinha indica que, embora tenhamos permissão para gerar pacotes binários, não podemos colocar nem esses pacotes nem o DISTFILES em um CD-ROM (ou mídia similar) para revenda. No entanto, os pacotes binários e os DISTFILES do ports ainda estarão disponíveis via FTP/HTTP.

Se esta variável for definida junto com NO_PACKAGE, então apenas o DISTFILES do port estará disponível e somente via FTP/HTTP.

Defina a variável NO_CDROM para uma string descrevendo o motivo pelo qual o port não pode ser redistribuído em CD-ROM. Por exemplo, use isto se a licença do port for somente para uso "não comercial".

6.4.3. NOFETCHFILES

Arquivos definidos em NOFETCHFILES não podem ser obtidos de nenhum dos MASTER_SITES. Um exemplo de tal tipo de arquivo é quando o arquivo é fornecido apenas em CD-ROM pelo fornecedor.

Ferramentas que verificam a disponibilidade desses arquivos nos MASTER_SITES devem ignorar estes arquivos e não informar nada sobre eles.

6.4.4. RESTRICTED

Defina esta variável sozinha, se a licença do aplicativo não permitir o espelhamento do DISTFILES e nem a distribuição do pacote binário de forma alguma.

Não defina as variáveis NO_CDROM ou NO_PACKAGE juntamente com a variável RESTRICT, uma vez que esta última variável implica as anteriores.

Defina a variável RESTRICTED para uma string que descreva o motivo pelo qual o port não pode ser redistribuído. Normalmente, isso indica que o port contém software proprietário e que o usuário precisará baixar manualmente o DISTFILES, possivelmente após se registrar para ter acesso ao

software ou após concordar em aceitar os termos de um EULA.

6.4.5. RESTRICTED_FILES

Quando a variável RESTRICT ou a NO_CDROM está definida, o valor padrão normalmente contém ${DISTFILES}${PATCHFILES} caso contrário, ela fica vazia. Se apenas alguns dos arquivos da distribuição forem restritos, defina essa variável para listá-los.

6.4.6. LEGAL_TEXT

Se o port tem preocupações legais as quais não foram abordadas pelas variáveis acima, defina a variável LEGAL_TEXT para uma string explicando a preocupação. Por exemplo, se o FreeBSD obteve uma permissão especial para redistribuir o binário, esta variável deve indicar isso.

6.4.7. /usr/ports/LEGAL e LEGAL

Um port que defina qualquer uma das variáveis acima também deverá ser adicionado ao /usr/ports/LEGAL. A primeira coluna é uma glob que corresponde aos distfiles restritos. A segunda coluna é a origem do port. A terceira coluna é a saída do comando make -VLEGAL.

6.4.8. Exemplos

A maneira preferida de declarar "os distfiles para este port devem ser obtidos manualmente" é a seguinte:

```
.if !exists(${DISTDIR}/${DISTNAME}${EXTRACT_SUFX})
IGNORE= may not be redistributed because of licensing reasons. Please visit some-
website to accept their license and download ${DISTFILES} into ${DISTDIR}
.endif
```

Isso tanto informa o usuário, quanto define os metadados apropriados na máquina do usuário para uso por programas automatizados.

Note que esta estrofe deve ser precedida por uma inclusão de bsd.port.pre.mk.

6.5. Mecanismos de Compilação

6.5.1. Compilando Ports em Paralelo

O framework de ports do FreeBSD suporta compilação paralela usando múltiplos subprocessos do comando make, o que permite que os sistemas SMP utilizem todo o poder disponível da CPU, permitindo que as compilações dos ports sejam mais rápidas e eficazes.

Isso é alcançado passando-se a flag -jX para o make(1) executando no código do fornecedor. Este é o comportamento de compilação padrão dos ports. Infelizmente, nem todos os ports lidam bem com compilações paralelas e pode ser necessário desabilitar explicitamente esse recurso adicionando a variável MAKE_JOBS_UNSAFE=yes. Ela é usada quando um port é conhecido por não funcionar com a opção -jX devido a race conditions e problemas de compilação intermitentes.

Ao definir a variável MAKE_JOBS_UNSAFE, é muito importante explicar com um comentário no Makefile, ou pelo menos na mensagem de commit, *porque* o port não pode ser compilado quando ela está ativa. Caso contrário, é quase impossível corrigir o problema ou testar se ele foi corrigido ao efetuar o commit de uma atualização em uma data posterior.

6.5.2. make, gmake, e imake

Existem várias implementações diferentes do make. O software portado geralmente requer uma implementação específica, como o GNU make, conhecido no FreeBSD como gmake.

Se o port usa o GNU make, adicione o gmake no USES.

A variável MAKE_CMD pode ser usada para referenciar o comando específico configurado pelo USES no Makefile do port. Use o MAKE_CMD apenas dentro dos Makefiles do aplicativo no WRKSRC para chamar o comando make para a implementação esperada pelo software portado.

Se o port é um aplicativo X que usa o Imake para criar o Makefile do Imakefile, defina USES=imake. Veja a seção sobre USES=imake no Usando Macros USES para mais detalhes.

Se o Makefile do port tem algo diferente de all como o target de compilação principal, defina a variável ALL_TARGET adequadamente. O mesmo vale para install e INSTALL_TARGET.

6.5.3. Script configure

Se o port usa o script configure para gerar Makefiles a partir do Makefile.in defina GNU_CONFIGURE=yes. Para dar argumentos extras ao script configure (o argumento padrão é --prefix=${PREFIX} --infodir=${PREFIX}/${INFO_PATH} --mandir = ${MANPREFIX}/man --build = ${CONFIGURE_TARGET}), defina estes argumentos extras em CONFIGURE_ARGS. Variáveis de ambiente extras podem ser passadas usando CONFIGURE_ENV.

Tabela 9. Variáveis para ports que usam o configure

Variável	Significa
GNU_CONFIGURE	O port usa o script configure para preparar a construção.
HAS_CONFIGURE	Igual a GNU_CONFIGURE, exceto que o destino de configuração padrão não é adicionado a CONFIGURE_ARGS.
CONFIGURE_ARGS	Argumentos adicionais passados para o script configure.
CONFIGURE_ENV	Variáveis de ambiente adicionais a serem definidas para execução de script configure.
CONFIGURE_TARGET	Substitui o target de configuração padrão. O valor padrão é ${MACHINE_ARCH}-portbld-freebsd${OSREL}.

6.5.4. Usando o cmake

Para ports que usam CMake, defina USES=cmake.

Tabela 10. Variáveis para ports que usam o cmake

Variável	Significa
CMAKE_ARGS	Flags do CMake especificas para o port a serem passadas para o binário do cmake.
CMAKE_ON	Para cada entrada em CMAKE_ON, um valor booleano ativado é adicionado ao CMAKE_ARGS. Veja CMAKE_ON and CMAKE_OFF.
CMAKE_OFF	Para cada entrada em CMAKE_OFF, um valor booleano desativado é adicionado ao CMAKE_ARGS. Veja CMAKE_ON and CMAKE_OFF.
CMAKE_BUILD_TYPE	Tipo de compilação (perfis de compilação predefinidos para o CMake). O padrão é Release ou Debug se a variável WITH_DEBUG estiver definida.
CMAKE_SOURCE_PATH	Caminho para o diretório do fonte. O padrão é ${WRKSRC}.
CONFIGURE_ENV	Variáveis de ambiente adicionais a serem definidas para o binário do cmake.

Tabela 11. Variáveis que os usuários podem definir para compilações com cmake

Variável	Significa
CMAKE_NOCOLOR	Desativa o output colorido na compilação. Não é definido por padrão, a menos que BATCH ou PACKAGE_BUILDING esteja definido.

CMake suporta estes perfis de construção: Debug, Release, RelWithDebInfo e MinSizeRel. Debug e Release sistema de respeito de perfis *FLAGS, RelWithDebInfo e MinSizeRel ajustará CFLAGS para -O2 -g e -Os -DNDEBUG correspondentemente. O valor do invólucro inferior CMAKE_BUILD_TYPE é exportado para PLIST_SUB e deve ser usado se o port for instalar *.cmake dependendo do tipo de compilação (veja devel/kf5-kcrash por um exemplo). Por favor, note que alguns projetos podem definir seus próprios perfis de compilação e/ou forçar um tipo específico de compilação CMAKE_BUILD_TYPE dentro de CMakeLists.txt. Para fazer um port para tal projeto respeite CFLAGS e WITH_DEBUG, as definições CMAKE_BUILD_TYPE devem ser removidas desses arquivos.

A maioria dos projetos baseados em CMake suportam um método de compilação out-of-source. A compilação out-of-source de um port é a configuração padrão. Uma compilação in-source pode ser executada usando-se o sufixo :insource. Em uma compilação out-of-source, CONFIGURE_WRKSRC, BUILD_WRKSRC e INSTALL_WRKSRC serão definidos como ${WRKDIR}/.Build e esse diretório será usado para manter todos os arquivos gerados durante os estágios de configuração e compilação, deixando o diretório de origem intacto.

Exemplo 56. Exemplo de `USES=cmake`

Este trecho demonstra o uso do CMake para um port. O `CMAKE_SOURCE_PATH` geralmente não é necessário, mas pode ser definido quando os fontes não estão localizados no diretório superior ou se apenas um subconjunto do projeto for compilado pelo port.

```
USES=           cmake
CMAKE_SOURCE_PATH=  ${WRKSRC}/subproject
```

Exemplo 57. `CMAKE_ON` *and* `CMAKE_OFF`

Ao adicionar valores booleanos a variável `CMAKE_ARGS`, será mais fácil usar as variáveis `CMAKE_ON` e `CMAKE_OFF` em vez disso. Desta forma:

```
CMAKE_ON=   VAR1 VAR2
CMAKE_OFF=  VAR3
```

É equivalente a:

```
CMAKE_ARGS= -DVAR1:BOOL=TRUE -DVAR2:BOOL=TRUE -DVAR3:BOOL=FALSE
```

Isto é apenas para os valores padrão desativados do `CMAKE_ARGS`. Os helpers descritos em `OPT_CMAKE_BOOL` e `OPT_CMAKE_BOOL_OFF` usam a mesma semântica, mas para valores opcionais.

6.5.5. Usando `scons`

Se o port usa SCons, definir `USES=scons`.

Para fazer os SConstruct de terceiros respeitarem tudo o que é passado para SCons no ambiente (isto é, o mais importante, `CC/CXX/CFLAGS/CXXFLAGS`), altere o SConstruct para que o `Evironment` de compilação fique da seguinte forma:

```
env = Environment(**ARGUMENTS)
```

Ele poderá então ser modificado com `env.Append` e `env.Replace`.

6.5.6. Compilando Aplicações Rust com `cargo`

Para ports que usam Cargo, defina `USES=cargo`.

Tabela 12. Variáveis que os Usuários Podem Configurar para Compilar `cargo`

Variável	Padrão	Descrição
`CARGO_CRATES`		Lista de crates que o port depende. Cada entrada precisa ter um formato como `cratename-semver` por exemplo, `libc-0.2.40`. Os mantenedores de ports podem gerar essa lista a partir do Cargo.lock usando o comando `make cargo-crates`. É possível alterar manualmente as versões dos crates, mas tenha em mente as dependências transitivas.
`CARGO_FEATURES`		Lista de recursos do aplicativo a serem compilados (lista separada por espaço). Para desativar todos os recursos padrão, adicione o token especial `--no-default-features` para `CARGO_FEATURES`. Passar manualmente para `CARGO_BUILD_ARGS`, `CARGO_INSTALL_ARGS`, e `CARGO_TEST_ARGS` não é necessário.
`CARGO_CARGOTOML`	`${WRKSRC}/Cargo.toml`	O caminho para o Cargo.toml que será usado.
`CARGO_CARGOLOCK`	`${WRKSRC}/Cargo.lock`	O caminho para o Cargo.lock que será utilizado para o `make cargo-crates`. É possível especificar mais de um arquivo de bloqueio quando necessário.
`CARGO_ENV`		Uma lista de variáveis de ambiente para passar para o Cargo semelhante a `MAKE_ENV`.
`RUSTFLAGS`		Flags para passar para o compilador Rust.
`CARGO_CONFIGURE`	`yes`	Use o padrão `do-configure`.
`CARGO_UPDATE_ARGS`		Argumentos extras para passar para o Cargo durante a fase de configuração. Os argumentos válidos podem ser consultados com `cargo update --help`.

Variável	Padrão	Descrição
CARGO_BUILDDEP	yes	Adiciona uma dependência de compilação em lang/rust.
CARGO_CARGO_BIN	${LOCALBASE}/bin/cargo	Localização do binário do cargo.
CARGO_BUILD	yes	Use o padrão do-build.
CARGO_BUILD_ARGS		Argumentos extras para passar para o Cargo durante a fase de compilação. Argumentos válidos podem ser consultados com cargo buil --help.
CARGO_INSTALL	yes	Use o padrão do-install.
CARGO_INSTALL_ARGS		Argumentos extras para passar para o Cargo durante a fase de instalação. Os argumentos válidos podem ser consultados com cargo isntall --help.
CARGO_INSTALL_PATH	.	Caminho para o crate instalar. Isto é passado para o cargo install via argumento --path. Quando múltiplos caminhos são informados, o cargo install é executado múltiplas vezes.
CARGO_TEST	yes	Use o padrão do-test.
CARGO_TEST_ARGS		Argumentos extras para passar para o Cargo durante a fase de teste. Os argumentos válidos podem ser consultados com cargo test --help.
CARGO_TARGET_DIR	${WRKDIR}/target	Localização do diretório de saída do cargo.
CARGO_DIST_SUBDIR	rust/crates	Diretório relativo a DISTDIR onde os arquivos de distribuição do crate serão armazenados.
CARGO_VENDOR_DIR	${WRKSRC}/cargo-crates	Localização do diretório do fornecedor onde todas os crates serão extraídos. Tente manter isto sob PATCH_WRKSRC, para que os patches possam ser aplicados facilmente.

Variável	Padrão	Descrição
CARGO_USE_GITHUB	no	Ativa a busca de crates bloqueadas para commits específicos do Git no GitHub via GH_TUPLE. Isso tentará modificar o Cargo.toml no WRKDIR para apontar para os fontes offline, em vez de buscá-los em um repositório Git durante a compilação.
CARGO_USE_GITLAB	no	O mesmo que CARGO_USE_GITHUB mas para instâncias GitLab e GL_TUPLE.

Exemplo 58. Criando um Port para uma Aplicação Simples em Rust

Criar um port baseado em cargo é um processo de três estágios. Primeiro, precisamos fornecer um modelo de port que busque o arquivo de distribuição do aplicativo:

```
PORTNAME=   tokei
DISTVERSIONPREFIX=  v
DISTVERSION=    7.0.2
CATEGORIES= devel

MAINTAINER= tobik@FreeBSD.org
COMMENT=    Display statistics about your code

USES=       cargo
USE_GITHUB= yes
GH_ACCOUNT= Aaronepower

.include <bsd.port.mk>
```

Gerar uma distinfo inicial:

```
% make makesum
=> Aaronepower-tokei-v7.0.2_GH0.tar.gz doesn't seem to exist in
/usr/ports/distfiles/.
=> Attempting to fetch
https://codeload.github.com/Aaronepower/tokei/tar.gz/v7.0.2?dummy=/Aaronepower-
tokei-v7.0.2_GH0.tar.gz
fetch:
https://codeload.github.com/Aaronepower/tokei/tar.gz/v7.0.2?dummy=/Aaronepower-
tokei-v7.0.2_GH0.tar.gz: size of remote file is not known
Aaronepower-tokei-v7.0.2_GH0.tar.gz                     45 kB  239 kBps 00m00s
```

Agora o arquivo de distribuição está pronto para uso e podemos ir em frente e extrair as dependências crate do pacote Cargo.lock:

```
% make cargo-crates
CARGO_CRATES=   aho-corasick-0.6.4 \
                ansi_term-0.11.0 \
                arrayvec-0.4.7 \
                atty-0.2.9 \
                bitflags-1.0.1 \
                byteorder-1.2.2 \
                [...]
```

A saída deste comando precisa ser colada diretamente no Makefile:

```
PORTNAME=   tokei
DISTVERSIONPREFIX=  v
DISTVERSION=    7.0.2
CATEGORIES= devel

MAINTAINER= tobik@FreeBSD.org
COMMENT=    Display statistics about your code

USES=       cargo
USE_GITHUB= yes
GH_ACCOUNT= Aaronepower

CARGO_CRATES=   aho-corasick-0.6.4 \
                ansi_term-0.11.0 \
                arrayvec-0.4.7 \
                atty-0.2.9 \
                bitflags-1.0.1 \
                byteorder-1.2.2 \
                [...]

.include <bsd.port.mk>
```

O distinfo precisa ser regenerado para conter todos os arquivos de distribuição dos crates:

```
% make makesum
=> rust/crates/aho-corasick-0.6.4.tar.gz doesn't seem to exist in
/usr/ports/distfiles/.
=> Attempting to fetch https://crates.io/api/v1/crates/aho-
corasick/0.6.4/download?dummy=/rust/crates/aho-corasick-0.6.4.tar.gz
rust/crates/aho-corasick-0.6.4.tar.gz         100% of   24 kB 6139 kBps 00m00s
=> rust/crates/ansi_term-0.11.0.tar.gz doesn't seem to exist in
/usr/ports/distfiles/.
=> Attempting to fetch
https://crates.io/api/v1/crates/ansi_term/0.11.0/download?dummy=/rust/crates/ansi_
```

```
term-0.11.0.tar.gz
rust/crates/ansi_term-0.11.0.tar.gz           100% of   16 kB   21 MBps 00m00s
=> rust/crates/arrayvec-0.4.7.tar.gz doesn't seem to exist in
/usr/ports/distfiles/.
=> Attempting to fetch
https://crates.io/api/v1/crates/arrayvec/0.4.7/download?dummy=/rust/crates/arrayve
c-0.4.7.tar.gz
rust/crates/arrayvec-0.4.7.tar.gz             100% of   22 kB 3237 kBps 00m00s
=> rust/crates/atty-0.2.9.tar.gz doesn't seem to exist in /usr/ports/distfiles/.
=> Attempting to fetch https://crates.io/api/v1/crates/atty/0.2.9/download?dummy
=/rust/crates/atty-0.2.9.tar.gz
rust/crates/atty-0.2.9.tar.gz                 100% of 5898  B   81 MBps 00m00s
=> rust/crates/bitflags-1.0.1.tar.gz doesn't seem to exist in
/usr/ports/distfiles/.
[...]
```

O port está agora pronto para uma compilação de teste e ajustes adicionais, como criar um plist, escrever uma descrição, adicionar informações de licença, opções, etc. como é normal.

Se você não estiver testando seu port em um ambiente limpo, como com o Poudriere, lembre-se de executar `make clean` antes de qualquer teste.

Exemplo 59. Ativando Recursos Adicionais do Aplicativo

Alguns aplicativos definem recursos adicionais em seus Cargo.toml. Eles podem ser compilados definindo a variável `CARGO_FEATURES` no port.

Aqui nós habilitamos as features Tokei's `json` e `yaml`:

```
CARGO_FEATURES= json yaml
```

Exemplo 60. Features de Codificação de Aplicativos como Opções de Port

Um exemplo de seção `[features]` no Cargo.toml pode parecer assim:

```
[features]
pulseaudio_backend = ["librespot-playback/pulseaudio-backend"]
portaudio_backend = ["librespot-playback/portaudio-backend"]
default = ["pulseaudio_backend"]
```

`pulseaudio_backend` é uma feature padrão. Ela está sempre ativada, a menos que desativemos explicitamente os recursos padrão adicionando `--no-default-features` para o `CARGO_FEATURES`. Aqui nós mudamos as features `portaudio_backend` e `pulseaudio_backend` em opções de port:

```
CARGO_FEATURES= --no-default-features
```

```
OPTIONS_DEFINE= PORTAUDIO PULSEAUDIO

PORTAUDIO_VARS=     CARGO_FEATURES+=portaudio_backend
PULSEAUDIO_VARS=    CARGO_FEATURES+=pulseaudio_backend
```

Exemplo 61. Listando Licenças Crate

Os crates têm suas próprias licenças. É importante saber o que elas são ao adicionar o bloco `LICENSE` para o port (ver Licenças). O target auxiliar `cargo-crates-licenses` tentará listar todas as licenças de todos os crates definidos no `CARGO_CRATES`.

```
% make cargo-crates-licenses
aho-corasick-0.6.4  Unlicense/MIT
ansi_term-0.11.0    MIT
arrayvec-0.4.7      MIT/Apache-2.0
atty-0.2.9          MIT
bitflags-1.0.1      MIT/Apache-2.0
byteorder-1.2.2     Unlicense/MIT
[...]
```

Os nomes das licenças geradas com `make cargo-create-licenses` são expressões de licenças do SPDX 2.1 que não correspondem aos nomes de licença definidos na estrutura de ports. Eles precisam ser traduzidos para os nomes de Lista de Licenças Predefinidas.

6.5.7. Usando `meson`

Para ports que usam Meson, defina `USES=meson`.

Tabela 13. Variáveis para ports que usam o `meson`

Variável	Descrição
`MESON_ARGS`	Flags do Meson especificas para o port a serem passadas para o binário do `meson`.
`MESON_BUILD_DIR`	Caminho para o diretório de compilação relativo ao `WRKSRC`. O padrão é `_build`.

Exemplo 62. Exemplo de `USES=meson`

Este trecho demonstra o uso do Meson para um port.

```
USES=       meson
MESON_ARGS= -Dfoo=enabled
```

6.5.8. Compilando Aplicações Go

Para ports que usam Go, defina `USES=go`. Consulte `go` para obter a lista de variáveis que podem ser configuradas para controlar o processo de compilação.

Exemplo 63. Criando um Port para uma Aplicação Baseada em Módulos Go

Criar um port baseado em Go é um processo de cinco estágios. Primeiro, precisamos fornecer um modelo de port que baixa o arquivo de distribuição do aplicativo:

```
PORTNAME=   ghq
DISTVERSIONPREFIX=  v
DISTVERSION=    0.12.5
CATEGORIES= devel

MAINTAINER= tobik@FreeBSD.org
COMMENT=    Remote repository management made easy

USES=       go:modules
USE_GITHUB= yes
GH_ACCOUNT= motemen

.include <bsd.port.mk>
```

Gerar uma distinfo inicial:

```
% make makesum
===>  License MIT accepted by the user
=> motemen-ghq-v0.12.5_GH0.tar.gz doesn't seem to exist in /usr/ports/distfiles/.
=> Attempting to fetch
https://codeload.github.com/motemen/ghq/tar.gz/v0.12.5?dummy=/motemen-ghq-
v0.12.5_GH0.tar.gz
fetch: https://codeload.github.com/motemen/ghq/tar.gz/v0.12.5?dummy=/motemen-ghq-
v0.12.5_GH0.tar.gz: size of remote file is not known
motemen-ghq-v0.12.5_GH0.tar.gz                          32 kB  177 kBps    00s
```

Agora o arquivo de distribuição está pronto para uso e podemos extrair as dependências necessárias de módulos Go. Esta etapa requer a instalação do ports-mgmt/modules2tuple:

```
% make gomod-vendor
[...]
GH_TUPLE=   \

Songmu:gitconfig:v0.0.2:songmu_gitconfig/vendor/github.com/Songmu/gitconfig \
        daviddengcn:go-
colortext:186a3d44e920:daviddengcn_go_colortext/vendor/github.com/daviddengcn/go-
colortext \
        go-yaml:yaml:v2.2.2:go_yaml_yaml/vendor/gopkg.in/yaml.v2 \
```

```
        golang:net:3ec191127204:golang_net/vendor/golang.org/x/net \
        golang:sync:112230192c58:golang_sync/vendor/golang.org/x/sync \
        golang:xerrors:3ee3066db522:golang_xerrors/vendor/golang.org/x/xerrors \
        motemen:go-
colorine:45d19169413a:motemen_go_colorine/vendor/github.com/motemen/go-colorine \
        urfave:cli:v1.20.0:urfave_cli/vendor/github.com/urfave/cli
```

A saída deste comando precisa ser colada diretamente no Makefile:

```
PORTNAME=   ghq
DISTVERSIONPREFIX=  v
DISTVERSION=    0.12.5
CATEGORIES= devel

MAINTAINER= tobik@FreeBSD.org
COMMENT=    Remote repository management made easy

USES=       go:modules
USE_GITHUB= yes
GH_ACCOUNT= motemen
GH_TUPLE=
Songmu:gitconfig:v0.0.2:songmu_gitconfig/vendor/github.com/Songmu/gitconfig \
        daviddengcn:go-
colortext:186a3d44e920:daviddengcn_go_colortext/vendor/github.com/daviddengcn/go-
colortext \
        go-yaml:yaml:v2.2.2:go_yaml_yaml/vendor/gopkg.in/yaml.v2 \
        golang:net:3ec191127204:golang_net/vendor/golang.org/x/net \
        golang:sync:112230192c58:golang_sync/vendor/golang.org/x/sync \
        golang:xerrors:3ee3066db522:golang_xerrors/vendor/golang.org/x/xerrors \
        motemen:go-
colorine:45d19169413a:motemen_go_colorine/vendor/github.com/motemen/go-colorine \
        urfave:cli:v1.20.0:urfave_cli/vendor/github.com/urfave/cli

.include <bsd.port.mk>
```

O distinfo precisa ser gerado novamente para conter todos os arquivos de distribuição:

```
% make makesum
=> Songmu-gitconfig-v0.0.2_GH0.tar.gz doesn't seem to exist in
/usr/ports/distfiles/.
=> Attempting to fetch
https://codeload.github.com/Songmu/gitconfig/tar.gz/v0.0.2?dummy=/Songmu-
gitconfig-v0.0.2_GH0.tar.gz
fetch: https://codeload.github.com/Songmu/gitconfig/tar.gz/v0.0.2?dummy=/Songmu-
gitconfig-v0.0.2_GH0.tar.gz: size of remote file is not known
Songmu-gitconfig-v0.0.2_GH0.tar.gz                    5662  B  936 kBps    00s
=> daviddengcn-go-colortext-186a3d44e920_GH0.tar.gz doesn't seem to exist in
/usr/ports/distfiles/.
=> Attempting to fetch https://codeload.github.com/daviddengcn/go-
```

```
colortext/tar.gz/186a3d44e920?dummy=/daviddengcn-go-colortext-
186a3d44e920_GH0.tar.gz
fetch: https://codeload.github.com/daviddengcn/go-
colortext/tar.gz/186a3d44e920?dummy=/daviddengcn-go-colortext-
186a3d44e920_GH0.tar.gz: size of remote file is not known
daviddengcn-go-colortext-186a3d44e920_GH0.tar.        4534  B 1098 kBps    00s
[...]
```

O port está agora pronto para uma compilação de teste e ajustes adicionais, como criar um plist, escrever uma descrição, adicionar informações de licença, opções, etc. como é normal.

Se você não estiver testando seu port em um ambiente limpo, como com o Poudriere, lembre-se de executar `make clean` antes de qualquer teste.

Exemplo 64. Definindo o Nome do Binário ou o Caminho da Instalação

Alguns ports precisam instalar o binário resultante com um nome diferente ou em um caminho diferente do padrão `${PREFIX}/bin`. Isso pode ser feito usando a sintaxe de tupla `GO_TARGET`, por exemplo:

```
GO_TARGET=  ./cmd/ipfs:ipfs-go
```

irá instalar o binário `ipfs` como `${PREFIX}/bin/ipfs-go` e

```
GO_TARGET=  ./dnscrypt-proxy:${PREFIX}/sbin/dnscrypt-proxy
```

irá instalar `dnscrypt-proxy` em `${PREFIX}/sbin`.

6.5.9. Compilando Aplicações Haskell com `cabal`

Para ports que usam Cabal, defina o sistema de compilação `USES=cabal`. Consulte `cabal` para obter a lista de variáveis que podem ser configuradas para controlar o processo de compilação.

Exemplo 65. Criando um Port para uma Aplicação Hackage-hosted Haskell

Ao preparar um port Haskell Cabal, o programa devel/hs-cabal-install é necessário, portanto, certifique-se de que esteja instalado previamente. Primeiro, precisamos definir variáveis de ports comuns que permitem ao cabal-install buscar o arquivo de distribuição de pacotes:

```
PORTNAME=   ShellCheck
DISTVERSION=    0.6.0
CATEGORIES= devel

MAINTAINER= haskell@FreeBSD.org
COMMENT=    Shell script analysis tool
```

```
USES=       cabal

.include <bsd.port.mk>
```

Esse Makefile mínimo nos permite baixar o arquivo de distribuição:

```
% make cabal-extract
[...]
Downloading the latest package list from hackage.haskell.org
cabal get ShellCheck-0.6.0
Downloading  ShellCheck-0.6.0
Downloaded   ShellCheck-0.6.0
Unpacking to ShellCheck-0.6.0/
```

Agora, temos o arquivo de descrição do pacote ShellCheck.cabal, que permite baixar todas as dependências do pacote, incluindo as transitivas:

```
% make cabal-extract-deps
[...]
Resolving dependencies...
Downloading  base-orphans-0.8.2
Downloaded   base-orphans-0.8.2
Downloading  primitive-0.7.0.0
Starting     base-orphans-0.8.2 (lib)
Building     base-orphans-0.8.2 (lib)
Downloaded   primitive-0.7.0.0
Downloading  dlist-0.8.0.7
[...]
```

Como efeito colateral, as dependências do pacote também são compiladas, portanto, o comando pode levar algum tempo. Uma vez feito, uma lista de dependências necessárias pode ser gerada:

```
% make make-use-cabal
USE_CABAL=QuickCheck-2.12.6.1 \
hashable-1.3.0.0 \
integer-logarithms-1.0.3 \
[...]
```

Pacotes Haskell podem conter revisões, assim como nos ports do FreeBSD. As revisões podem afetar apenas os arquivos .cabal, mas ainda é importante extraí-los. Para verificar os itens `USE_CABAL` quanto a atualizações de revisão disponíveis, execute o seguinte comando:

```
% make make-use-cabal-revs
USE_CABAL=QuickCheck-2.12.6.1_1 \
```

```
hashable-1.3.0.0 \
integer-logarithms-1.0.3_2 \
[...]
```

Observe os números de versão adicionais após o símbolo _. Coloque a lista USE_CABAL recém-gerada em vez de uma antiga.

Finalmente, o distinfo precisa ser gerado novamente para conter todos os arquivos de distribuição:

```
% make makesum
=> ShellCheck-0.6.0.tar.gz doesn't seem to exist in
/usr/local/poudriere/ports/git/distfiles/cabal.
=> Attempting to fetch https://hackage.haskell.org/package/ShellCheck-
0.6.0/ShellCheck-0.6.0.tar.gz
ShellCheck-0.6.0.tar.gz                                136 kB  642 kBps    00s
=> QuickCheck-2.12.6.1/QuickCheck-2.12.6.1.tar.gz doesn't seem to exist in
/usr/local/poudriere/ports/git/distfiles/cabal.
=> Attempting to fetch https://hackage.haskell.org/package/QuickCheck-
2.12.6.1/QuickCheck-2.12.6.1.tar.gz
QuickCheck-2.12.6.1/QuickCheck-2.12.6.1.tar.gz          65 kB  361 kBps    00s
[...]
```

O port está agora pronto para uma compilação de teste e ajustes adicionais, como criar um plist, escrever uma descrição, adicionar informações de licença, opções, etc. como é normal.

Se você não estiver testando seu port em um ambiente limpo, como com o Poudriere, lembre-se de executar `make clean` antes de qualquer teste.

6.6. Usando o GNU Autotools

Se um port precisar de algum software GNU Autotools, adicione `USES=autoreconf`. Veja `autoreconf` Para maiores informações.

6.7. Usando o GNU `gettext`

6.7.1. Uso Básico

Se o port requer o `gettext`, defina `USES=gettext`, e o port herdará a dependência libintl.so do devel/gettext. Outros valores para uso do `gettext` estão listados em `USES=gettext`.

Um caso bastante comum é um port que utilize o `gettext` e o `configure`. Geralmente, o GNU `configure` deve ser capaz de localizar o `gettext` automaticamente.

```
USES=   gettext
```

```
GNU_CONFIGURE=  yes
```

Se falhar, dicas da localização do `gettext` podem ser informados por meio do `CPPFLAGS` e LDFLAGS` utilizando `localbase` do seguinte modo:

```
USES=   gettext localbase:ldflags
GNU_CONFIGURE=  yes
```

6.7.2. Uso Opcional

Alguns softwares permitem desabilitar o NLS. Por exemplo, passando `--disable-nls` para o `configure`. Nesse caso, o port deve usar `gettext` condicionalmente, dependendo do status da opção `NLS`. Para ports de baixa a média complexidade, use este idioma:

```
GNU_CONFIGURE=      yes

OPTIONS_DEFINE=     NLS
OPTIONS_SUB=        yes

NLS_USES=       gettext
NLS_CONFIGURE_ENABLE=   nls

.include <bsd.port.mk>
```

Ou usando a maneira antiga de usar opções:

```
GNU_CONFIGURE=      yes

OPTIONS_DEFINE=     NLS

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MNLS}
USES+=          gettext
PLIST_SUB+=     NLS=""
.else
CONFIGURE_ARGS+=    --disable-nls
PLIST_SUB+=     NLS="@comment "
.endif

.include <bsd.port.mk>
```

O próximo item na lista de tarefas a fazer é organizar de forma condicional os arquivos do catálogo de mensagens na lista de pacotes. A parte do Makefile desta tarefa já é fornecida pela expressão idiomática. Isto é explicado na seção sobre práticas avançadas de pkg-plist. Em poucas palavras, cada ocorrência de `%%NLS%%` dentro de pkg-plist será substituído por “@comment” se o NLS estiver

desativado ou por uma cadeia nula se o NLS estiver ativado. Consequentemente, as linhas prefixadas por `%%NLS%%` se tornarão meros comentários na lista de empacotamento final se o NLS estiver desativado; caso contrário, o prefixo será deixado de fora. Em seguida, insira `%%NLS%%` antes de cada caminho para um arquivo de catálogo de mensagens em pkg-plist. Por exemplo:

```
%%NLS%%share/locale/fr/LC_MESSAGES/foobar.mo
%%NLS%%share/locale/no/LC_MESSAGES/foobar.mo
```

Em casos de alta complexidade, técnicas mais avançadas podem ser necessárias, como geração dinâmica de lista de empacotamento.

6.7.3. Manipulando Diretórios do Catálogo de Mensagens

Há um ponto a ser observado sobre a instalação de arquivos de catálogo de mensagens. Os diretórios de destino para eles, que residem em LOCALBASE/shared/locale, não devem ser criados e removidos por um port. Os idiomas mais populares têm seus respectivos diretórios listados em PORTSDIR/Templates/BSD.local.dist. Os diretórios para muitos outros idiomas são governados pelo port devel/gettext. Consulte o seu pkg-plist e veja se o port vai instalar um arquivo de catálogo de mensagens para um idioma exclusivo.

6.8. Usando Perl

E se o `MASTER_SITES` estiver configurado para `CPAN`, o subdiretório correto é geralmente selecionado automaticamente. Se o subdiretório padrão estiver errado, o `CPAN/Module` pode ser usado para alterá-lo. O `MASTER_SITES` também pode ser definido para o antigo `MASTER_SITE_PERL_CPAN`, então o valor preferido para o `MASTER_SITE_SUBDIR` é o nome da hierarquia de nível superior. Por exemplo, o valor recomendado para `p5-Module-Name` é `Module`. A hierarquia de nível superior pode ser examinada em cpan.org. Isso mantém o port funcionando quando o autor do módulo muda.

A exceção a essa regra é quando o diretório relevante não existe ou o distfile não existe neste diretório. Neste caso, é permitido usar o id do autor como `MASTER_SITE_SUBDIR`. A macro `CPAN: AUTOR` pode ser usada, a qual será traduzida para o diretório de autor com hash. Por exemplo,`CPAN: AUTOR` será convertido para `autores/id/A/AU/AUTOR`.

Quando um port precisa de suporte a Perl, ele deve definir `USES=perl5` com o opcional `USE_PERL5` descrito em descrição do USES no perl5.

Tabela 14. Variáveis Somente Leitura para Ports Que Usam Perl

Variáveis Somente de Leitura	Significa
`PERL`	O caminho completo do interpretador Perl 5, seja no sistema ou instalado a partir de um port, mas sem o número da versão. Use isso quando o software precisar do caminho para o interpretador Perl. Para substituir as linhas “#!” em scripts, use USES=shebangfix.

Variáveis Somente de Leitura	Significa
PERL_VERSION	A versão completa do Perl instalada (por exemplo, 5,8,9).
PERL_LEVEL	A versão do Perl instalada como um inteiro no formato MNNNPP (por exemplo,500809).
PERL_ARCH	Local no qual o Perl armazena as bibliotecas dependentes da arquitetura. O valor padrão aponta para ${ARCH}-freebsd.
PERL_PORT	Nome do port Perl instalado (por exemplo, perl5).
SITE_PERL	Nome do diretório para onde vão os pacotes Perl específicos do site. Esse valor é adicionado a PLIST_SUB.

Ports de Módulos Perl que não possuem um site oficial devem linkar para cpan.org na linha WWW do pkg-descr. O formato preferido para a URL é http://search.cpan.org/dist/Module-Name/ (incluindo a barra final).

Não use ${SITE_PERL} em declarações de dependência. Fazê-lo pressupõe que o perl5.mk foi incluído, o que nem sempre é verdade. Os ports que dependem desse port terão dependências incorretas se os arquivos desse port forem movidos posteriormente em uma atualização. O caminho certo para declarar as dependências do módulo Perl é mostrado no exemplo abaixo.

Exemplo 66. Exemplo de Dependência Perl

```
p5-IO-Tee>=0.64:devel/p5-IO-Tee
```

Para ports Perl que instalam páginas de manual, as macros PERL5_MAN3 e PERL5_MAN1 podem ser usadas dentro do pkg-plist. Por exemplo,

```
lib/perl5/5.14/man/man1/event.1.gz
lib/perl5/5.14/man/man3/AnyEvent::I3.3.gz
```

pode ser substituído por

```
%%PERL5_MAN1%%/event.1.gz
%%PERL5_MAN3%%/AnyEvent::I3.3.gz
```

Não existem macros PERL5_MANx para as outras seções (sendo *x* igual a 2 e de 4 até 9) porque estes são instalados nos diretórios comuns.

Exemplo 67. Um Port Que Requer Perl Apenas para Compilar

Como o valor padrão para USE_PERL5 é build e run, configure-o para:

```
USES=       perl5
USE_PERL5=  build
```

Exemplo 68. Um Port Que Também Requer Perl Para Patch

De tempos em tempos, o uso do sed(1) para patches se torna insuficiente. Quando usar perl(1) fica mais facil, para isso utilize:

```
USES=       perl5
USE_PERL5=  patch build run
```

Exemplo 69. Um Módulo Perl Que Precisa de `ExtUtils::MakeMaker` para Compilar

A maioria dos módulos Perl vêm com um script configure Makefile.PL. Neste caso, defina:

```
USES=       perl5
USE_PERL5=  configure
```

Exemplo 70. Um Módulo Perl Que Precisa `Módulo::Build` para Compilar

Quando um modulo Perl vem com um script configure Build.PL, pode exigir Module:Build, nesse caso, defina

```
USES=       perl5
USE_PERL5=  modbuild
```

Se for ao contrário, e exigir Module::Build::Tiny, defina

```
USES=       perl5
USE_PERL5=  modbuildtiny
```

6.9. Usando o X11

6.9.1. Componentes X.Org

A implementação do X11 disponível na Coleção de Ports é o X.Org. Se o aplicativo depender de

componentes X, adicione `USES= xorg` e defina `USE_XORG` na lista de componentes necessários. Uma lista completa pode ser encontrada em `xorg`.

O Projeto Mesa é um esforço para fornecer implementação gratuita do OpenGL. Para especificar uma dependência em vários componentes deste projeto, use a variável `USE_GL`. Veja `gl` para a lista completa dos componentes disponíveis. Para compatibilidade com versões anteriores, o valor `yes` direciona para `glu`.

Exemplo 71. Exemplo `USE_XORG`

```
USES=       gl xorg
USE_GL=     glu
USE_XORG=   xrender xft xkbfile xt xaw
```

Tabela 15. Variáveis para Ports Que Usam X

`USES= imake`	O port usa `imake`.
`XMKMF`	Definir o caminho de `xmkmf` se não no `PATH`. Padrão para `xmkmf -a`.

Exemplo 72. Usando Variáveis Relacionadas ao X11

```
# Use some X11 libraries
USES=       xorg
USE_XORG=   x11 xpm
```

6.9.2. Ports que Requerem Motif

Se o port requer uma biblioteca Motif, defina `USES=motif` no Makefile. A implementação padrão do Motif é x11-toolkits/open-motif. Os usuários podem escolher o x11-toolkits/lesstif em vez disso, definindo `WANT_LESSTIF` no seu make.conf.

O `MOTIFLIB` será definido por motif.mk para referenciar a biblioteca Motif apropriada. Por favor, corrija o fonte do port para usar `${MOTIFLIB}` onde quer que a biblioteca Motif seja referenciada no Makefile original ou no Imakefile.

Existem dois casos comuns:

- Se o port se referir à biblioteca Motif como `-lXm` em seu Makefile ou Imakefile, substitua `${MOTIFLIB}` por isso.
- Se o port usa `XmClientLibs` em seu Imakefile, mude para `${MOTIFLIB} ${XTOOLLIB} ${XLIB}`.

Observe que o `MOTIFLIB` (geralmente) se expande para `-L/usr/local/lib -lXm -lXp` ou `/usr/local/lib/libXm.a`, então não há necessidade de adicionar `-L` ou `-l` na frente.

6.9.3. Fontes X11

Se o port instalar fontes para o X Window System, coloque-as em LOCALBASE/lib/X11/fontes/local.

6.9.4. Obtendo um DISPLAY Falso com Xvfb

Algumas aplicações requerem uma tela X11 funcional para que a compilação seja bem-sucedida. Isso representa um problema para as máquinas que operam sem um monitor. Quando essa variável é usada, a infraestrutura de compilação iniciará o X virtual framebuffer. Um DISPLAY funcional é então passado para a compilação. Veja USES=exibição para os possíveis argumentos.

```
USES=   display
```

6.9.5. Entradas de Desktop

Entradas de desktop (um padrão Freedesktop) fornecem uma maneira de ajustar automaticamente os recursos do desktop quando um novo programa é instalado, sem a necessidade de intervenção do usuário. Por exemplo, programas recém-instalados aparecem automaticamente nos menus de aplicativos de ambientes de desktop compatíveis. Entradas de Desktop surgiram no ambiente de desktop GNOME, mas agora são um padrão e também funcionam com o KDE e o Xfce. Esta pitada de automação fornece um benefício real para o usuário, e as entradas de desktop são incentivadas para aplicativos que podem ser usados em um ambiente desktop.

Usando Arquivos .desktop Pré-definidos

Ports que incluem *.desktop pré-definidos devem incluir estes arquivos no pkg-plist e instalá-los no diretório $LOCALBASE/shared/applications. A macro INSTALL_DATA é útil para instalar esses arquivos.

Atualizando o Banco de Dados do Desktop

Se um port tiver uma entrada MimeType em seu portname.desktop, o banco de dados do desktop deve ser atualizado após a instalação e desinstalação. Para fazer isso, defina USES= desktop-file-utils.

Criando Entradas de Desktop com DESKTOP_ENTRIES

As entradas desktop podem ser facilmente criadas para aplicativos usando DESKTOP_ENTRIES. Um arquivo chamado name.desktop será criado, instalado e adicionado ao pkg-plist automaticamente. A sintaxe é:

```
DESKTOP_ENTRIES=    "NAME" "COMMENT" "ICON" "COMMAND" "CATEGORY" StartupNotify
```

A lista de possíveis categorias está disponível no Site Freedesktop. StartupNotify indica se a aplicação é compatível com *notificações de inicialização*. Estes são tipicamente um indicador gráfico como um relógio que aparece no ponteiro do mouse, menu ou painel para dar ao usuário uma indicação quando um programa está sendo iniciado. Um programa que seja compatível com as notificações de inicialização limpa o indicador depois de iniciado. Programas que não são

compatíveis com as notificações de inicialização nunca limpariam o indicador (possivelmente confundindo e enfurecendo o usuário) e devem ter `StartupNotify` definido como `false` então o indicador não é mostrado.

Exemplo:

```
DESKTOP_ENTRIES=    "ToME" "Roguelike game based on JRR Tolkien's work" \
            "${DATADIR}/xtra/graf/tome-128.png" \
            "tome -v -g" "Application;Game;RolePlaying;" \
            false
```

6.10. Usando o GNOME

6.10.1. Introdução

Este capítulo explica a estrutura do framework GNOME utilizado pelos ports. O framework pode ser dividido livremente nos componentes base, componentes desktop GNOME e algumas macros especiais que simplificam o trabalho dos mantenedores dos ports.

6.10.2. Usando `USE_GNOME`

Adicionar esta variável ao port permite o uso das macros e componentes definidos em bsd.gnome.mk. O código em bsd.gnome.mk adiciona as dependências de tempo de compilação, tempo de execução ou biblioteca necessárias ou o tratamento de arquivos especiais. Aplicativos GNOME sob o FreeBSD usam o framework `USE_GNOME`. Inclua todos os componentes necessários como uma lista separada por espaço. Os componentes `USE_GNOME` são divididos nessas listas virtuais: componentes básicos, componentes do GNOME 3 e componentes legados. Se o port precisa apenas de bibliotecas GTK3, este é o caminho mais curto para defini-lo:

```
USE_GNOME=  gtk30
```

Componentes `USE_GNOME` adicionam automaticamente as dependências de que precisam. Por favor, veja Componentes GNOMEpara uma lista exaustiva de todos componentes `USE_GNOME` e quais outros componentes eles implicam e suas dependências.

Aqui está um exemplo de Makefile para um port do GNOME que usa muitas das técnicas descritas neste documento. Por favor, use-o como um guia para criar novos ports.

```
# $FreeBSD$

PORTNAME=   regexxer
DISTVERSION=    0.10
CATEGORIES= devel textproc gnome
MASTER_SITES=   GNOME

MAINTAINER= kwm@FreeBSD.org
```

```
COMMENT=    Interactive tool for performing search and replace operations

USES=       gettext gmake localbase:ldflags pathfix pkgconfig tar:xz
GNU_CONFIGURE=  yes
USE_GNOME=  gnomeprefix intlhack gtksourceviewmm3
INSTALLS_ICONS= yes

GLIB_SCHEMAS=   org.regexxer.gschema.xml

.include <bsd.port.mk>
```

A macro USE_GNOME se utilizada sem nenhum argumento não irá adicionar nenhuma dependência ao port. O USE_GNOME não pode ser definido depois do bsd.port.pre.mk.

6.10.3. Variáveis

Esta seção explica quais macros estão disponíveis e como elas são usadas. Como elas são usadas no exemplo acima. A Componentes GNOME tem uma explicação mais detalhada. A variável USE_GNOME precisa ser definido para que essas macros sejam úteis.

INSTALLS_ICONS

Ports GTK+ que instalam ícones de estilo Freedesktop em ${LOCALBASE}/shared/icons deve usar essa macro para garantir que os ícones sejam armazenados em cache e exibidos corretamente. O arquivo de cache é nomeado icon-theme.cache. Não inclua esse arquivo em pkg-plist. Essa macro manipula isso automaticamente. Esta macro não é necessária para Qt, que usam um método interno.

GLIB_SCHEMAS

Lista de todos os arquivos de esquema de glib que o port instala. A macro adicionará os arquivos ao plist do port e manipulará o registro destes arquivos na instalação e desinstalação.

Os arquivos de esquema do glib são escritos em XML e terminam com a extensão gschema.xml. Eles estão instalados no diretório share/glib-2.0/schemas/. Esses arquivos de esquema contêm todos os valores de configuração do aplicativo com as configurações padrão. O banco de dados real usado pelos aplicativos é construído por glib-compile-schema, que é executado pela macro GLIB_SCHEMAS.

```
GLIB_SCHEMAS=foo.gschema.xml
```

Não adicione esquemas simplificados ao pkg-plist. Se eles estão listados em pkg-plist, eles não serão registrados e os aplicativos podem não funcionar corretamente.

GCONF_SCHEMAS

Liste todos os arquivos do esquema gconf. A macro adicionará os arquivos de esquema ao plist

do port e manipulará seu registro na instalação e desinstalação.

O GConf é o banco de dados baseado em XML que praticamente todos os aplicativos GNOME usam para armazenar suas configurações. Esses arquivos são instalados no banco de dados no diretório etc/gconf/schemas. Esse banco de dados é definido pelos arquivos de esquema instalados que são usados para gerar os arquivos chave %gconf.xml. Para cada arquivo de esquema instalado pelo port, deve existir uma entrada no Makefile:

```
GCONF_SCHEMAS=my_app.schemas my_app2.schemas my_app3.schemas
```

Os esquemas do Gconf estão listados na macro `GCONF_SCHEMAS` em vez do pkg-plist. Se eles estiverem listados em pkg-plist, eles não serão registrados e os aplicativos podem não funcionar corretamente.

`INSTALLS_OMF`

Os arquivos do Open Source Metadata Framework (OMF) são comumente usados pelos aplicativos GNOME 2. Esses arquivos contêm as informações do arquivo de ajuda do aplicativo e requerem processamento especial pelo ScrollKeeper/rarian. Para registrar adequadamente arquivos OMF ao instalar aplicativos GNOME a partir de pacotes, certifique-se de que os arquivos `omf` estão listados em `pkg-plist` e que o Makefile do port tem o `INSTALLS_OMF` definido:

```
INSTALLS_OMF=yes
```

Quando definido, bsd.gnome.mk digitaliza automaticamente o pkg-plist e adiciona diretivas `@exec` e `@unexec` para cada .omf para rastrear no banco de dados de registro do OMF.

6.11. Componentes GNOME

Para mais ajuda com um port GNOME, veja alguns dos ports existentes por exemplo. A pagina GNOME do FreeBSD tem informações de contato, se precisar de mais ajuda. Os componentes são divididos em componentes GNOME que estão atualmente em uso e componentes legados. Se o componente suportar argumento, eles serão listados entre parênteses na descrição. O primeiro é o padrão. "Ambos" são mostrados se o componente usar como padrão a adição às dependências de construção e execução.

Tabela 16. Componentes GNOME

Componente	Programa associado	Descrição
`atk`	accessibility/atk	Kit de ferramentas de acessibilidade (ATK)
`atkmm`	accessibility/atkmm	c++ bindings para atk
`cairo`	graphics/cairo	Biblioteca de gráficos vetoriais com suporte a saída entre dispositivos

Componente	Programa associado	Descrição
`cairomm`	graphics/cairomm	c++ bindings para o cairo
`dconf`	devel/dconf	Sistema de banco de dados de configuração (both, buil, run)
`evolutiondataserver3`	databases/evolution-data-server	Backends de dados para a suíte mail/PIM integrada do Evolution
`gdkpixbuf2`	graphics/gdk-pixbuf2	Biblioteca de gráficos para GTK+
`glib20`	devel/glib20	Biblioteca core do GNOME `glib20`
`glibmm`	devel/glibmm	c++ bindings para glib20
`gnomecontrolcenter3`	sysutils/gnome-control-center	Centro de Controle do GNOME 3
`gnomedesktop3`	x11/gnome-desktop	Biblioteca de interface do usuário do desktop GNOME 3
`gsound`	audio/gsound	Biblioteca GObject para reproduzir sons do sistema (both, build, run)
`gtk-update-icon-cache`	graphics/gtk-update-icon-cache	Utilitário Gtk-update-icon-cache do kit de ferramentas Gtk
`gtk20`	x11-toolkits/gtk20	Kit de ferramentas Gtk+ 2
`gtk30`	x11-toolkits/gtk30	Kit de ferramentas Gtk+ 3
`gtkmm20`	x11-toolkits/gtkmm20	c++ bindings 2.0 para o kit de ferramentas gtk20
`gtkmm24`	x11-toolkits/gtkmm24	c++ bindings 2.4 para o kit de ferramentas gtk20
`gtkmm30`	x11-toolkits/gtkmm30	c++ bindings 3.0 para o kit de ferramentas gtk30
`gtksourceview2`	x11-toolkits/gtksourceview2	Widget que adiciona destaque de sintaxe para o GtkTextView
`gtksourceview3`	x11-toolkits/gtksourceview3	Widget de texto que adiciona destaque de sintaxe ao widget GtkTextView
`gtksourceviewmm3`	x11-toolkits/gtksourceviewmm3	c++ bindings para a biblioteca gtksourceview3
`gvfs`	devel/gvfs	Sistema de arquivos virtual do GNOME
`intltool`	textproc/intltool	Ferramenta para Internacionalização (veja também intlhack)

Componente	Programa associado	Descrição
`introspection`	devel/gobject-introspection	Ligações de introspecção e ferramentas básicas para gerar ligações de introspecção. Na maioria das vezes: build é suficiente, e :both/:run só é necessário para aplicativos que usam ligações de introspecção. (both, build, run)
`libgda5`	databases/libgda5	Fornece acesso uniforme a diferentes tipos de fontes de dados
`libgda5-ui`	databases/libgda5-ui	Biblioteca de interface do usuário da biblioteca libgda5
`libgdamm5`	databases/libgdamm5	c++ bindings para a biblioteca libgda5
`libgsf`	devel/libgsf	Abstração extensível de I/O para lidar com formatos de arquivo estruturados
`librsvg2`	graphics/librsvg2	Biblioteca para analisar e renderizar arquivos gráficos vetoriais SVG
`libsigc++20`	devel/libsigc++20	Framework de Callback para C++
`libxml++26`	textproc/libxml++26	c++ bindings para a biblioteca libxml2
`libxml2`	textproc/libxml2	Biblioteca do parser XML (both, build, run)
`libxslt`	textproc/libxslt	Biblioteca XSLT C (both, build, run)
`metacity`	x11-wm/metacity	Gerenciador de janelas do GNOME
`nautilus3`	x11-fm/nautilus	Gerenciador de arquivos GNOME
`pango`	x11-toolkits/cave	Estrutura de código aberto para o layout e renderização do texto i18n
`pangomm`	x11-toolkits/pangomm	c++ bindings para a biblioteca pango
`py3gobject3`	devel/py3-gobject3	Python 3, GObject 3.0 bindings
`pygobject3`	devel/py-gobject3	Python 2, GObject 3.0 bindings

Componente	Programa associado	Descrição
`vte3`	x11-toolkits/vte3	Widget de terminal com melhor acessibilidade e suporte I18N

Tabela 17. Componentes Macro do GNOME

Componente	Descrição
`gnomeprefix`	Forneça `configure` com alguns locais padrão.
`intlhack`	O mesmo que intltool, porém com os patches necessários para garantir o share/locale/. Por favor, use somente quando `intltool` sozinho não for suficiente.
`referencehack`	Esta macro existe para ajudar a dividir a API ou a documentação de referência em seu próprio port.

Tabela 18. Componentes Legados do GNOME

Componente	Programa associado	Descrição
`atspi`	accessibility/at-spi	Interface do Provedor de Serviços de Tecnologia Assistiva
`esound`	audio/esound	Pacote de som do Enlightenment
`gal2`	x11-toolkits/gal2	Coleção de widgets obtidos do GNOME 2 gnumeric
`gconf2`	devel/gconf2	Sistema de banco de dados de configuração para o GNOME 2
`gconfmm26`	devel/gconfmm26	c++ bindings para o gconf2
`gdkpixbuf`	graphics/gdk-pixbuf	Biblioteca de gráficos para GTK+
`glib12`	devel/glib12	biblioteca principal glib 1.2
`gnomedocutils`	textproc/gnome-doc-utils	Utilitários de documentação para o GNOME
`gnomemimedata`	misc/gnome-mime-data	MIME e banco de dados de aplicativos para o GNOME 2
`gnomesharp20`	x11-toolkits/gnome-sharp20	Interfaces do GNOME 2 para o tempo de execução do .NET
`gnomespeech`	accessibility/gnome-speech	API de conversão de texto em voz do GNOME 2
`gnomevfs2`	devel/gnome-vfs	Sistema de Arquivos Virtual do GNOME 2
`gtk12`	x11-toolkits/gtk12	Kit de ferramentas Gtk+ 1.2

Componente	Programa associado	Descrição
gtkhtml3	www/gtkhtml3	Mecanismo leve de renderização/impressão/edição de HTML
gtkhtml4	www/gtkhtml4	Mecanismo leve de renderização/impressão/edição de HTML
gtksharp20	x11-toolkits/gtk-sharp20	Interfaces GTK+ e GNOME 2 para o runtime .NET
gtksourceview	x11-toolkits/gtksourceview	Widget que adiciona destaque de sintaxe para o GtkTextView
libartgpl2	graphics/libart_lgpl	Biblioteca para gráficos 2D de alto desempenho
libbonobo	devel/libbonobo	Componente e sistema de documentos compostos para o GNOME 2
libbonoboui	x11-toolkits/libbonoboui	GUI frontend para o componente libbonobo do GNOME 2
libgda4	databases/libgda4	Fornece acesso uniforme a diferentes tipos de fontes de dados
libglade2	devel/libglade2	Biblioteca glade do GNOME 2
libgnome	x11/libgnome	Bibliotecas para o GNOME 2, um ambiente de desktop GNU
libgnomecanvas	graphics/libgnomecanvas	Biblioteca Gráfica para o GNOME 2
libgnomekbd	x11/libgnomekbd	Biblioteca compartilhada de teclado do GNOME 2
libgnomeprint	print/libgnomeprint	Biblioteca de suporte de impressão do Gnome 2
libgnomeprintui	x11-toolkits/libgnomeprintui	Biblioteca de suporte de impressão do Gnome 2
libgnomeui	x11-toolkits/libgnomeui	Bibliotecas para a GUI do GNOME 2, um ambiente de desktop GNU
libgtkhtml	www/libgtkhtml	Mecanismo leve de renderização/impressão/edição de HTML
libgtksourceviewmm	x11-toolkits/libgtksourceviewmm	c++ binding do GtkSourceView

Componente	Programa associado	Descrição
`libidl`	devel/libIDL	Biblioteca para criação de árvores de arquivo do CORBA IDL
`libsigc++12`	devel/libsigc++12	Framework de Callback para C++
`libwnck`	x11-toolkits/libwnck	Biblioteca usada para escrever pagers e listas de tarefas
`libwnck3`	x11-toolkits/libwnck3	Biblioteca usada para escrever pagers e listas de tarefas
`orbit2`	devel/ORBit2	CORBA ORB de alto desempenho com suporte para a linguagem C
`pygnome2`	x11-toolkits/py-gnome2	Python bindings para GNOME 2
`pygobject`	devel/py-gobject	Python 2, GObject 2.0 bindings
`pygtk2`	x11-toolkits/py-gtk2	Conjunto de Python bindings para GTK+
`pygtksourceview`	x11-toolkits/py-gtksourceview	Python bindings para GtkSourceView 2
`vte`	x11-toolkits/vte	Widget de terminal com melhor acessibilidade e suporte I18N

Tabela 19. Componentes Obsoletos: Não Use

Componente	Descrição
`pangox-compat`	O pangox-compatfoi descontinuado e separado do pacote pango.

6.12. Usando o Qt

Para ports que fazem parte do Qt, veja `qt-dist`.

6.12.1. Ports que requerem o Qt

A coleção de ports fornece suporte para o Qt 5 com `USES+=qt:5`. Configure o `USE_QT` para a lista de componentes obrigatórios do Qt (bibliotecas, ferramentas, plugins).

O framework Qt exporta um número de variáveis que podem ser usadas por ports, algumas delas listadas abaixo:

Tabela 20. Variáveis Fornecidas aos Ports Que Usam o Qt

`QMAKE`	Caminho completo para o binário `qmake`.

LRELEASE	Caminho completo para utilitário Irelease.
MOC	Caminho completo para moc.
RCC	Caminho completo para rcc.
UIC	Caminho completo para uic.
QT_INCDIR	Diretório include Qt.
QT_LIBDIR	Caminho das bibliotecas Qt.
QT_PLUGINDIR	Caminho de plugins do Qt.

6.12.2. Seleção de Componentes

As dependências individuais das ferramentas e da biblioteca Qt devem ser especificadas em USE_QT. Todo componente pode ser sufixado com _build ou _run, o sufixo indica se a dependência no componente está no tempo de compilação ou no tempo de execução. Se um sufixo não for usado, a dependência do componente será tanto em tempo de compilação quanto em tempo de execução. Geralmente, os componentes da biblioteca são especificados como unsuffixed, os componentes das ferramentas são especificados com o sufixo _build e os componentes dos plugins são especificados com o sufixo _run. Os componentes mais comumente usados estão listados abaixo (todos os componentes disponíveis estão listados em _USE_QT_ALL e _USE_QT5_ONLY em /usr/ports/Mk/Uses/qt.mk):

Tabela 21. Componentes da Biblioteca Qt Disponíveis

Nome	Descrição
3d	Módulo Qt3D
assistant	Navegador de documentação do Qt 5
canvas3d	Módulo Qt canvas3d
charts	Módulo de gráficos Qt 5
concurrent	Módulo multi-threading Qt
connectivity	Módulo de conectividade Qt (Bluetooth/NFC)
core	Módulo não-gráfico do núcleo Qt
datavis3d	Módulo de visualização de dados 3D Qt 5
dbus	Módulo de comunicação entre processos Qt D-Bus
declarative	Framework declarativo Qt para interfaces dinâmicas de usuário
designer	Designer gráfico de interface de usuário do Qt 5
diag	Ferramenta para relatar informações de diagnóstico sobre o Qt e seu ambiente
doc	Documentação do Qt 5
examples	Código-fonte dos exemplos do Qt 5

Nome	Descrição
`gamepad`	Módulo de Gamepad Qt 5
`graphicaleffects`	Efeitos gráficos rápidos do Qt
`gui`	Módulo de interface gráfica do usuário do Qt
`help`	Módulo de integração de ajuda on-line do Qt
`l10n`	Mensagens localizadas do Qt
`linguist`	Ferramenta de tradução do Qt 5
`location`	Módulo de localização do Qt
`multimedia`	Módulo de suporte de áudio, vídeo, rádio e câmera do Qt
`network`	Módulo de rede do Qt
`networkauth`	Módulo de autenticação de rede do Qt
`opengl`	Módulo de suporte OpenGL compatível com o Qt 5
`paths`	Cliente de linha de comando para QStandardPaths
`phonon4`	Framework de multimídia do KDE
`pixeltool`	Lupa de tela do Qt 5
`plugininfo`	Dumper de metadados do plugin Qt5
`printsupport`	Módulo de suporte de impressão do Qt
`qdbus`	Interface de linha de comando do Qt para o D-Bus
`qdbusviewer`	Interface gráfica do Qt 5 para o D-Bus
`qdoc`	Gerador de documentação do Qt
`qdoc-data`	Arquivos de configuração do QDoc
`qev`	Ferramenta de introspecção de eventos Qt QWidget
`qmake`	Gerador de Makefile do Qt
`quickcontrols`	Conjunto de controles para construir interfaces completas no Qt Quick
`quickcontrols2`	Conjunto de controles para construir interfaces completas no Qt Quick
`remoteobjects`	Módulo SXCML Qt5
`script`	Módulo de script compatível com Qt 4
`scripttools`	Componentes adicionais do Qt Script
`scxml`	Módulo SXCML Qt5

Nome	Descrição
`sensors`	Módulo de sensores do Qt
`serialbus`	Funções do Qt para acessar sistemas de bus industriais
`serialport`	Funções do Qt para acessar portas seriais
`speech`	Recursos de acessibilidade para o Qt5
`sql`	Módulo de integração a banco de dados SQL do Qt
`sql-ibase`	Plugin de banco de dados InterBase/Firebird do Qt
`sql-mysql`	Plugin de banco de dados MySQL do Qt
`sql-odbc`	Plugin Qt para conectividade Open Database
`sql-pgsql`	Plugin de banco de dados do PostgreSQL do Qt
`sql-sqlite2`	Plugin de banco de dados SQLite 2 do Qt
`sql-sqlite3`	Plugin de banco de dados SQLite 3 do Qt
`sql-tds`	Plugin de conectividade ao banco de dados TDS do Qt
`svg`	Módulo de suporte SVT do Qt
`testlib`	Módulo de teste unitário do Qt
`uiplugin`	Interface de plug-in do Qt widget personalizado para o Qt Designer
`uitools`	Módulo de suporte a formulários de interface de usuário do Qt Designer
`virtualkeyboard`	Módulo de teclado virtual do Qt 5
`wayland`	Qt5 wrapper para o Wayland
`webchannel`	Biblioteca Qt 5 para integração de C++/QML com clientes HTML/js
`webengine`	Biblioteca Qt 5 para renderizar conteúdo da web
`webkit`	QtWebKit com uma base de código WebKit mais moderna
`websockets`	Implementação do protocolo WebSocket do Qt
`websockets-qml`	Implementação do protocolo WebSocket do Qt (QML bindings)
`webview`	Componente do Qt para exibir o conteúdo da web
`widgets`	Módulo de widgets C++ do Qt

Nome	Descrição
`x11extras`	Recursos específicos da plataforma Qt para sistemas baseados em X11
`xml`	Implementações SAX e DOM do Qt
`xmlpatterns`	Suporte do Qt para XPath, XQuery, XSLT e XML Schema

Para determinar as bibliotecas das quais um aplicativo depende, execute o `ldd` no executável principal após uma compilação bem sucedida.

Tabela 22. Componentes Disponíveis da Ferramenta Qt

Nome	Descrição
`buildtools`	Ferramentas de compilação (`moc`, `rcc`), necessária para quase todas as aplicações do Qt.
`linguisttools`	ferramentas de localização: `lrelease`, `lupdate`
`qmake`	Utilitário gerador/compilador de Makefile

Tabela 23. Componentes Disponíveis de Plugin Qt

Nome	Descrição
`imageformats`	plugins para formatos de imagem TGA, TIFF e MNG

Exemplo 73. Selecionando Componentes do Qt 5

Neste exemplo, o aplicativo portado usa a biblioteca de interface gráfica do usuário do Qt 5, a biblioteca principal do Qt 5, todas as ferramentas de geração de código do Qt 5 e o gerador de Makefile do Qt 5. Uma vez que a biblioteca `gui` implica na dependência da biblioteca principal, o `core` não precisa ser especificado. As ferramentas de geração de código do Qt 5 `moc`, `uic` e `rcc`, bem como o gerador de Makefile `qmake` são necessários apenas em tempo de compilação, assim eles são especificados com o sufixo `_build`:

```
USES=   qt:5
USE_QT= gui buildtools_build qmake_build
```

6.12.3. Usando `qmake`

Se o aplicativo fornecer um arquivo de projeto qmake (*.pro), defina `USES=qmake` junto com `USE_QTx`. Observe que `USES=qmake` já implica uma dependência de compilação no qmake, portanto, o componente qmake pode ser omitido de `USE_QT`. Igual ao CMake, o qmake suporta compilações out-of-source, que podem ser ativadas especificando o argumento `outsource` (ver`USES=qmake`exemplo) .

Tabela 24. Argumentos Possíveis para `USES= qmake`

Variável	Descrição
`no_configure`	Não adicione o target configure. Isso é implícito pelo `HAS_CONFIGURE=yes` e `GNU_CONFIGURE=yes`. Isso é requerido quando a compilação apenas precisa do ambiente de setup do `USES= qmake`, e dessa forma, executa-se o `qmake` por si próprio.
`no_env`	Suprime modificações dos ambientes configure e make. É necessário somente quando `qmake` é usado para configurar o software e a compilação falha em entender a configuração do ambiente pelo `USES= qmake`.
`norecursive`	Não passe o argumento `-recursive` para o `qmake`.
`outsource`	Realiza uma compilação out-of-source.

Tabela 25. Variáveis para Ports Que Usam o `qmake`

Variável	Descrição
`QMAKE_ARGS`	Flags específicas do port qmake a serem passadas para o binario do `qmake`.
`QMAKE_ENV`	Variáveis de ambiente a serem definidas para o binario `qmake`. O padrão é `${CONFIGURE_ENV}`.
`QMAKE_SOURCE_PATH`	Caminho para os arquivos de projeto do qmake (.pro). O padrão é `${WRKSRC}` se uma compilação out-of-source for solicitada, caso contrário, deixe em branco.

Ao usar `USES= qmake`, estas configurações são implementadas:

```
CONFIGURE_ARGS+=    --with-qt-includes=${QT_INCDIR} \
            --with-qt-libraries=${QT_LIBDIR} \
            --with-extra-libs=${LOCALBASE}/lib \
            --with-extra-includes=${LOCALBASE}/include

CONFIGURE_ENV+= QTDIR="${QT_PREFIX}" QMAKE="${QMAKE}" \
        MOC="${MOC}" RCC="${RCC}" UIC="${UIC}" \
        QMAKESPEC="${QMAKESPEC}"

PLIST_SUB+= QT_INCDIR=${QT_INCDIR_REL} \
        QT_LIBDIR=${QT_LIBDIR_REL} \
        QT_PLUGINDIR=${QT_PLUGINDIR_REL}
```

Alguns scripts de configuração não suportam os argumentos acima. Para suprimir a modificação de `CONFIGURE_ENV` e `CONFIGURE_ARGS` defina `USES= qmake:no_env`.

Exemplo 74. Exemplo `USES= qmake`

Este trecho demonstra o uso do qmake para um port Qt 5:

```
USES=   qmake:outsource qt:5
USE_QT= buildtools_build
```

Aplicações Qt são frequentemente escritas para serem multi-plataforma e muitas vezes o X11/Unix não é a plataforma em que são desenvolvidas, o que por sua vez leva a certas pontas soltas, como:

- *Faltam caminhos de inclusão adicionais.* Muitos aplicativos vêm com suporte ao ícone da bandeja do sistema, mas não buscam inclusões e/ou bibliotecas nos diretórios do X11. Para adicionar diretórios aos includes e bibliotecas de pesquisa do `qmake` através da linha de comando, use:

  ```
  QMAKE_ARGS+=    INCLUDEPATH+=${LOCALBASE}/include \
          LIBS+=-L${LOCALBASE}/lib
  ```

- *Caminhos falsos de instalação.* Às vezes, dados como ícones ou arquivos .desktop são instalados por padrão em diretórios que não são verificados por aplicativos compatíveis com XDG. O editors/texmaker é um exemplo disso - veja patch-texmaker.pro no diretório de arquivos desse port para um modelo sobre como remediar isso diretamente no arquivo de projeto `qmake`.

6.13. Usando o KDE

6.13.1. Definições de Variáveis do KDE

Se a aplicação depender do KDE, defina `USES+=kde:5` e defina `USE_KDE` com a lista de componentes necessários. Sufixos `_build` e `_run` podem ser usados para forçar o tipo de dependência de componentes (por exemplo, `baseapps_run`). Se nenhum sufixo for definido, o tipo padrão de dependência será usado. Para forçar os dois tipos, adicione o componente duas vezes com os dois sufixos (por exemplo, `ecm_build ecm_run`). Os componentes disponíveis estão listados abaixo (os componentes atualizados também estão listados em /usr/ports/Mk/Uses/kde.mk):

Tabela 26. Componentes Disponíveis do KDE

Nome	Descrição
`activities`	Biblioteca de tempo de execução do KF5 para organizar o trabalho em atividades separadas
`activities-stats`	Estatísticas do KF5 para atividades
`activitymanagerd`	Serviço do sistema para gerenciar atividades do usuário, rastrear os padrões de uso
`akonadi`	Servidor de armazenamento para o KDE-Pim
`akonadicalendar`	Integração de Calendário do Akonadi

Nome	Descrição
akonadiconsole	Console de gerenciamento e depuração do Akonadi
akonadicontacts	Bibliotecas e daemons para implementar o gerenciamento de contatos do Akonadi
akonadiimportwizard	Importa dados de outros clientes de email para o KMail
akonadimime	Bibliotecas e daemons para implementar o tratamento básico de email
akonadinotes	Biblioteca do KDE para acessar caixas postais no formato MBox
akonadisearch	Bibliotecas e daemons para implementar buscas no Akonadi
akregator	Um leitor de feeds do KDE
alarmcalendar	API do KDE para alarmes do KAlarm
apidox	Ferramentas de Documentação da API KF5
archive	Biblioteca KF5 que fornece classes para lidar com formatos de arquivo
attica	Biblioteca da API do Open Collaboration Services do KDE 5
attica5	Biblioteca da API do Open Collaboration Services do KDE 5
auth	Abstração do KF5 para funcionalidades de autenticação e políticas do sistema
baloo	KF5 Framework para pesquisar e gerenciar metadados do usuário
baloo-widgets	Biblioteca BalooWidgets
baloo5	KF5 Framework para pesquisar e gerenciar metadados do usuário
blog	API do KDE para acesso ao weblogging
bookmarks	Biblioteca KF5 para bookmarks e para o formato XBEL
breeze	Arte, estilos e recursos do Plasma5 para o estilo visual Breeze
breeze-gtk	Estilo visual do Plasma5 Breeze para Gtk
breeze-icons	Tema de ícones do Breeze para o KDE
calendarcore	Biblioteca de acesso ao calendário do KDE

Nome	Descrição
`calendarsupport`	Bibliotecas de suporte de calendário para o KDEPim
`calendarutils`	Utilitário KDE e funções da interface do usuário para acessar o calendário
`codecs`	Biblioteca KF5 para manipulação de string
`completion`	Assistentes e widgets de conclusão de texto do KF5
`config`	Widgets do KF5 para diálogos de configuração
`configwidgets`	Widgets do KF5 para diálogos de configuração
`contacts`	Api do KDE para gerenciar informações de contato
`coreaddons`	Complementos do KF5 para o QtCore
`crash`	Biblioteca KF5 para lidar com análise de falhas e relatório de erros de aplicativos
`dbusaddons`	Complementos do KF5 para o QtDBus
`decoration`	Biblioteca do Plasma5 para criar decorações de janelas
`designerplugin`	Integração do KF5 para widgets de Framework no Qt Designer/Creator
`discover`	Ferramentas de gerenciamento de pacotes do Plasma5
`dnssd`	Abstração do KF5 para os recursos do sistema DNSSD
`doctools`	Geração de documentação do KF5 a partir do docbook
`drkonqi`	Manipulador de falhas do Plasma5
`ecm`	Módulos e scripts extras para o CMake
`emoticons`	Biblioteca KF5 para converter emoticons
`eventviews`	Bibliotecas de visualização de eventos para o KDEPim
`filemetadata`	Biblioteca KF5 para extrair metadados de arquivos
`frameworkintegration`	Espaço de trabalho e plugins de integração entre estruturas KF5
`gapi`	Biblioteca baseada no KDE para acessar serviços do Google

Nome	Descrição
`globalaccel`	Biblioteca KF5 para incluir suporte para atalhos do espaço de trabalho global
`grantlee-editor`	Editor para os temas de Grantlee
`grantleetheme`	KDE PIM grantleetheme
`gravatar`	Biblioteca para suporte a gravatar
`guiaddons`	Complementos do KF5 para o QtGui
`holidays`	Biblioteca do KDE para feriados do calendário
`hotkeys`	Biblioteca do Plasma5 para teclas de atalho
`i18n`	Framework avançado de internacionalização do KF5
`iconthemes`	Biblioteca KF5 para manipular ícones em aplicativos
`identitymanagement`	Identidades do KDE pim
`idletime`	Biblioteca KF5 para monitorar a atividade do usuário
`imap`	API do KDE para suporte a IMAP
`incidenceeditor`	Bibliotecas do editor de incidências para o KDE Pim
`infocenter`	Utilidade do Plasma5 fornecendo informações do sistema
`init`	Iniciador de processos KF5 para acelerar o lançamento de aplicativos do KDE
`itemmodels`	Modelos KF5 para o sistema Qt Model / View
`itemviews`	KF5 widget addons para Qt Model/View
`jobwidgets`	Widgets do KF5 para rastrear a instância do KJob
`js`	Biblioteca KF5 que fornece um interpretador de script ECMA
`jsembed`	Biblioteca KF5 para ligar objetos JavaScript a QObjects
`kaddressbook`	Gerenciador de contatos do KDE
`kalarm`	Agendador de alarmes pessoal
`kalarm`	Agendador de alarmes pessoal
`kate`	Framework básico do editor para o sistema KDE
`kcmutils`	Utilitários KF5 para trabalhar com KCModules

Nome	Descrição
kde-cli-tools	Ferramentas não interativas do sistema do Plasma5
kde-gtk-config	Configurador Plasma5 GTK2 e GTK3
kdeclarative	Biblioteca KF5 que prove a integração dos frameworks do QML e do KDE
kded	Daemon extensível do KF5 para fornecer serviços a nível do sistema
kdelibs4support	KF5 porting aid from KDELibs4
kdepim-addons	Complementos do KDE PIM
kdepim-apps-libs	Bibliotecas do KDE PIM relacionadas ao correio
kdepim-runtime5	Ferramentas e serviços do KDE PIM
kdeplasma-addons	Complementos do Plasma 5 para melhorar a experiência do Plasma
kdesu	Integração do KF5 com o su para privilégios elevados
kdewebkit	Biblioteca KF5 que fornece a integração do QtWebKit
kgamma5	Configurações de gama do monitor Plasma5
khtml	Motor de renderização KF5 KTHML
kimageformats	Biblioteca KF5 que fornece suporte para formatos de imagem adicionais
kio	Recurso e abstração de acesso à rede do KF5
kirigami2	Conjunto de componentes baseados em QtQuick
kitinerary	Modelo de dados e sistema de extração para informações de reservas de viagens
kmail	Cliente de correio do KDE
kmail	Cliente de correio do KDE
kmail-account-wizard	Assistente de conta de e-mail do KDE
kmenuedit	Editor de menu do Plasma5
knotes	Notas pop-up
kontact	Gerenciador de Informações Pessoais do KDE
kontact	Gerenciador de Informações Pessoais do KDE
kontactinterface	Cola do KDE para incorporar KParts no Kontact
korganizer	Programa de calendário e agendamento
kpimdav	Uma implementação do protocolo DAV com KJobs

Nome	Descrição
kpkpass	Biblioteca para lidar com pass files da Apple Wallet
kross	Aplicação de scripting multi-language do KF5
kscreen	Biblioteca de gerenciamento de tela do Plasma5
kscreenlocker	Arquitetura de tela de bloqueio seguro do Plasma5
ksmtp	Biblioteca job-based para enviar email através de um servidor SMTP
ksshaskpass	Frontend ssh-add do Plasma5
ksysguard	Utilitário Plasma5 para rastrear e controlar os processos em execução
kwallet-pam	Integração PAM do Plasma5 KWallet
kwayland-integration	Plugins de integração para um desktop baseado em Wayland
kwin	Gerenciador de janelas do Plasma5
kwrited	Daemon do Plasma5 para ouvir paredes e escrever mensagens
ldap	API de acesso LDAP para o KDE
libkcddb	Biblioteca KDE CDDB
libkcompactdisc	Biblioteca do KDE para interfaceamento com CDs de áudio
libkdcraw	Interface LibRaw para o KDE
libkdegames	Bibliotecas usadas pelos jogos do KDE
libkdepim	Bibliotecas KDE PIM
libkeduvocdocument	Biblioteca para leitura e gravação de arquivos de vocabulário
libkexiv2	Interface da biblioteca Exiv2 para o KDE
libkipi	Interface de Plugin de Imagem do KDE
libkleo	Gerenciador de certificados para o KDE
libksane	Interface da biblioteca SANE para o KDE
libkscreen	Biblioteca de gerenciamento de tela do Plasma5
libksieve	Bibliotecas de inspeção para o KDEPim
libksysguard	Biblioteca do Plasma5 para rastrear e controlar processos em execução
mailcommon	Bibliotecas comuns para o KDEPim
mailimporter	Importar arquivos mbox para o KMail

Nome	Descrição
`mailtransport`	Biblioteca do KDE para gerenciar o transporte de correio
`marble`	Globo virtual e atlas mundial para o KDE
`mbox`	Biblioteca do KDE para acessar caixas postais no formato MBox
`mbox-importer`	Importar arquivos mbox para o KMail
`mediaplayer`	Interface de plug-in do KF5 para recursos do media player
`messagelib`	Biblioteca para manipular mensagens
`milou`	Plasma5 Plasmóide para pesquisa
`mime`	Biblioteca para manipular dados MIME
`newstuff`	Biblioteca KF5 para baixar aplicativos da rede
`notifications`	Abstração KF5 para notificações do sistema
`notifyconfig`	Sistema de configuração KF5 para o KNotify
`okular`	Visualizador universal de documentos do KDE
`oxygen`	Estilo Oxygen Plasma5
`oxygen-icons5`	O tema de ícones do Oxygen para o KDE
`package`	Biblioteca KF5 para carregar e instalar pacotes
`parts`	Sistema de plugin centrado em documentos KF5
`people`	Biblioteca KF5 para fornecer acesso a contatos
`pim-data-exporter`	Importar e exportar configurações do KDE PIM
`pimcommon`	Bibliotecas comuns para o KDEPim
`pimtextedit`	Biblioteca do KDE para utilitários de edição de texto específicos do PIM
`plasma-browser-integration`	Componentes do Plasma5 para integrar navegadores na área de trabalho
`plasma-desktop`	Área de trabalho plasma Plasma5
`plasma-framework`	UI runtime baseado no plugin KF5 usado para escrever interfaces de usuários
`plasma-integration`	Plugins de integração do Qt Platform Theme para os workspaces do Plasma
`plasma-pa`	Misturador de áudio de pulso do Plasma5 Plasma
`plasma-sdk`	Aplicações do Plasma5 úteis para o desenvolvimento Plasma
`plasma-workspace`	Workspace Plasma5 Plasma

Nome	Descrição
`plasma-workspace-wallpapers`	Plasma5 wallpapers
`plotting`	Framework de plotagem leve KF5
`polkit-kde-agent-1`	Daemon do Plasma5 que fornece uma interface de usuário de autenticação do polkit
`powerdevil`	Ferramenta Plasma5 para gerenciar as configurações de consumo de energia
`prison`	API para produzir códigos de barras
`pty`	Abstração KF5 pty
`purpose`	Oferece ações disponíveis para um propósito específico
`qqc2-desktop-style`	Estilo Qt QuickControl2 para o KDE
`runner`	Sistema de consulta paralelizado do KF5
`service`	Plugin KF5 avançado e serviço de introspecção
`solid`	Integração e detecção de hardware do KF5
`sonnet`	Biblioteca de verificação de ortografia baseada no plugin do KF5
`syndication`	Biblioteca de manipulação de feeds do KDE
`syntaxhighlighting`	Mecanismo de destaque de sintaxe KF5 para texto e código estruturados
`systemsettings`	Configurações do sistema Plasma5
`texteditor`	Editor avançado de texto embutido do KF5
`textwidgets`	Widgets avançados do KF5 para edição de texto
`threadweaver`	Complementos do KF5 para o QtDBus
`tnef`	API do KDE para o tratamento de dados TNEF
`unitconversion`	Biblioteca KF5 para conversão de unidade
`user-manager`	Gerenciador de usuários do Plasma5
`wallet`	Contêiner KF5 seguro e unificado para senhas de usuários
`wayland`	Wrapper da biblioteca KF5 Cliente e Servidor para as bibliotecas Wayland
`widgetsaddons`	Complementos do KF5 para o QtWidgets
`windowsystem`	Biblioteca KF5 para acesso ao sistema de janelas
`xmlgui`	Janelas principais configuráveis pelo usuário do KF5
`xmlrpcclient`	Interação KF5 com serviços XMLRPC

Exemplo 75. Exemplo `USE_KDE`

Este é um exemplo simples para um port do KDE. O `USES= cmake` instrui o port a utilizar o CMake, uma ferramenta de configuração amplamente usada pelos projetos do KDE (veja Usando o `cmake`para informações detalhadas sobre o uso). O `USE_KDE` informa a dependência das bibliotecas do KDE. Os componentes necessários do KDE e outras dependências podem ser determinadas através do log de configuração. O `USE_KDE` não implica no `USE_QT`. Se um port requer alguns componentes do Qt, especifique-os em `USE_QT`.

```
USES=       cmake kde:5 qt:5
USE_KDE=    ecm
USE_QT=     core buildtools_build qmake_build
```

6.14. Usando o LXQt

As aplicações que dependem do LXQt devem definir `USES+= lxqt` e definir a variável `USE_LXQT` para a lista de componentes necessários da tabela abaixo

Tabela 27. Componentes disponíveis do LXQt

Nome	Descrição
`buildtools`	Auxiliares para módulos CMake adicionais
`libfmqt`	Libfm Qt bindings
`lxqt`	LXQt core library
`qtxdg`	Implementação do Qt das especificações do XDG do freedesktop.org

Exemplo 76. Exemplo `USE_LXQT`

Este é um exemplo simples, `USE_LXQT` adiciona uma dependência em bibliotecas LXQt. Os componentes necessários do LXQt e outras dependências podem ser determinados a partir do log de configuração.

```
USES=   cmake lxqt qt:5 tar:xz
USE_QT=     core dbus widgets buildtools_build qmake_build
USE_LXQT=   buildtools libfmqt
```

6.15. Usando Java

6.15.1. Definições de Variáveis

Se o port precisar de um Java™ Development Kit (JDK) para compilar, executar ou até mesmo extrair o distfile, então defina `USE_JAVA`.

Existem vários JDKs na coleção de ports, de vários fornecedores e em várias versões. Se o port precisar usar uma versão específica, especifique-a usando a variável `JAVA_VERSION`. A versão mais atual é java/openjdk15, com java/openjdk14, java/openjdk13, java/openjdk12, java/openjdk11, java/openjdk8, e java/openjdk7 também disponíveis.

Tabela 28. Variáveis Que Podem ser Definidas por Ports Que Usam Java

Variável	Significa
`USE_JAVA`	Defina para as variáveis restantes para ter algum efeito.
`JAVA_VERSION`	Lista das versões Java adequadas separadas por espaço para o port. Um opcional `"""` permite especificar um intervalo de versões (valores permitidos: `7[] 8[] 11[] 12[] 13[] 14[] 15[]`).
`JAVA_OS`	Lista de sistemas operacionais adequados do port JDK separados por espaço para o port (valores permitidos: `native linux`).
`JAVA_VENDOR`	Lista de fornecedores adequados de ports JDK separados por espaços para o port (valores permitidos: `freebsd bsdjava sun openjdk`).
`JAVA_BUILD`	Quando definido, adiciona o port JDK selecionado às dependências de compilação.
`JAVA_RUN`	Quando definido, adicione o port JDK selecionado às dependências de execução.
`JAVA_EXTRACT`	Quando definido, adicione o port JDK selecionado às dependências de extração.

Abaixo está a lista de todas as configurações que um port receberá após a configuração de `USE_JAVA`:

Tabela 29. Variáveis Fornecidas para Ports que Usam Java

Variável	Valor
`JAVA_PORT`	O nome do port do JDK (por exemplo,`java/openjdk6`).
`JAVA_PORT_VERSION`	A versão completa do port do JDK (por exemplo,`1.6.0`). Somente os dois primeiros dígitos deste número de versão são necessários, use `${JAVA_PORT_VERSION:C/^([0-9])\.([0-9])(.*)$/\1.\2/}`.
`JAVA_PORT_OS`	O sistema operacional usado pelo port do JDK (por exemplo, `'native'`).
`JAVA_PORT_VENDOR`	O fornecedor do port JDK (por exemplo,`'openjdk'`).
`JAVA_PORT_OS_DESCRIPTION`	Descrição do sistema operacional usado pelo port JDK (por exemplo, `'Native'`).

Variável	Valor
`JAVA_PORT_VENDOR_DESCRIPTION`	Descrição do fornecedor do port JDK (por exemplo,`'OpenJDK BSD Porting Team'`).
`JAVA_HOME`	Caminho para o diretório de instalação do JDK (por exemplo, '/usr/local/openjdk6').
`JAVAC`	Caminho para o compilador Java (por exemplo, '/usr/local/openjdk6/bin/javac').
`JAR`	Caminho para ferramenta `jar` a ser usada (por exemplo, '/usr/local/openjdk6/bin/jar' ou '/usr/local/bin/fastjar').
`APPLETVIEWER`	Caminho para o utilitário `appletviewer` (por exemplo,'/usr/local/openjdk6/bin/appletviewer').
`JAVA`	Caminho para o executável `Java`. Use isto para executar programas Java (por exemplo, '/usr/local/openjdk6/bin/java').
`JAVADOC`	Caminho para o utilitário `javadoc`.
`JAVAH`	Caminho para o programa `javah`.
`JAVAP`	Caminho para o programa `javap`.
`JAVA_KEYTOOL`	Caminho para o utilitário `keytool`.
`JAVA_N2A`	Caminho para a ferramenta `native2ascii`.
`JAVA_POLICYTOOL`	Caminho para o programa `policytool`.
`JAVA_SERIALVER`	Caminho para o utilitário `serialver`.
`RMIC`	Caminho para o gerador de stub/skeleton RMI, `rmic`.
`RMIREGISTRY`	Caminho para o programa de registro RMI, `rmiregistry`.
`RMID`	Caminho para o daemon do RMI `rmid`.
`JAVA_CLASSES`	Caminho para o arquivo que contém os arquivos de classe do JDK, ${JAVA_HOME}/jre/lib/rt.jar.

Use o `java-debug` make target para obter informações para depurar o port. Ele exibirá o valor de muitas das variáveis listadas anteriormente.

Além disso, essas constantes são definidas para que todos os ports Java possam ser instalados de maneira consistente:

Tabela 30. Constantes definidas para os ports que usam Java

Constante	Valor
`JAVASHAREDIR`	O diretório base para tudo relacionado ao Java. Padrão: ${PREFIX}/shared/java.

Constante	Valor
JAVAJARDIR	O diretório onde os arquivos JAR são instalados. Padrão: ${JAVASHAREDIR}/classes.
JAVALIBDIR	O diretório onde os arquivos JAR instalados por outros ports estão localizados. Padrão: ${LOCALBASE}/shared/java/classes.

As entradas relacionadas são definidas em ambos PLIST_SUB (documentado em Alterando o pkg-plist Baseado em Variáveis Make) e SUB_LIST.

6.15.2. Compilando com Ant

Quando o port deve ser compilado usando o Apache Ant, ele deve definir USE_ANT. Ant é, portanto, considerado o comando sub-make. Quando nenhum target do-build é definido pelo port, será definido um padrão que execute Ant de acordo com MAKE_ENV, MAKE_ARGS e ALL_TARGET. Isso é semelhante ao mecanismo USES=gmake, documentado em Mecanismos de Compilação.

6.15.3. Melhores Práticas

Ao portar uma biblioteca Java, o port precisa instalar o(s) arquivo(s) JAR em ${JAVAJARDIR} e o resto em ${JAVASHAREDIR}/${PORTNAME} (exceto para a documentação, veja abaixo). Para reduzir o tamanho do arquivo de empacotamento, faça referência aos arquivos JAR diretamente no Makefile. Use esta declaração (onde myport.jar é o nome do arquivo JAR instalado como parte do port):

```
PLIST_FILES+=   ${JAVAJARDIR}/myport.jar
```

Ao portar um aplicativo Java, o port geralmente instala tudo em um único diretório (incluindo suas dependências JAR). O uso de ${JAVASHAREDIR}/${PORTNAME} é fortemente indicado neste caso. Cabe ao mantenedor do port decidir se o port instala as dependências JAR adicionais sob esse diretório ou utiliza as já instaladas (de ${JAVAJARDIR}).

Ao portar um aplicativo Java™ que requer um servidor de aplicação, como o www/tomcat7 para executar o serviço, é bastante comum que o fornecedor distribua um .war. Um .war é uma aplicação Web ARchive a qual é extraído quando chamado pelo aplicativo. Evite adicionar um .war no pkg-plist. Isto não é considerado a melhor prática. Um servidor de aplicação irá expandir o arquivo war mas não irá remove-lo se o port for desinstalado. Uma forma mais desejável de trabalhar com este arquivo é extrair o seu conteudo, depois instalar os arquivos e, por fim, adicionar esses arquivos ao pkg-plist.

```
TOMCATDIR=  ${LOCALBASE}/apache-tomcat-7.0
WEBAPPDIR=  myapplication

post-extract:
    @${MKDIR} ${WRKDIR}/${PORTDIRNAME}
    @${TAR} xf ${WRKDIR}/myapplication.war -C ${WRKDIR}/${PORTDIRNAME}
```

```
do-install:
    cd ${WRKDIR} && \
    ${INSTALL} -d -o ${WWWOWN} -g ${WWWGRP} ${TOMCATDIR}/webapps/${PORTDIRNAME}
    cd ${WRKDIR}/${PORTDIRNAME} && ${COPYTREE_SHARE} \* ${WEBAPPDIR}/${PORTDIRNAME}
```

Independentemente do tipo de port (biblioteca ou aplicativo), a documentação adicional é instalada na mesma localização como para qualquer outro port. A ferramenta Javadoc é conhecida por produzir um conjunto diferente de arquivos, dependendo da versão do JDK utilizado. Para ports que não impõem o uso de um determinado JDK, é uma tarefa complexa especificar a lista de empacotamento (pkg-plist). Esta é uma razão pela qual os mantenedores de ports são fortemente encorajados a usar `PORTDOCS`. Além disso, mesmo se o conjunto de arquivos que serão gerados pelo `javadoc` puder ser previsto, o tamanho do pkg-plist resultante irá encorajar o uso do `PORTDOCS`.

O valor padrão para `DATADIR` é ${PREFIX}/shared/${PORTNAME}. É uma boa ideia sobreescrever `DATADIR` para ${JAVASHAREDIR}/${PORTNAME} para ports Java. De fato, `DATADIR` é automaticamente adicionado a `PLIST_SUB` (documentado em Alterando o pkg-plist Baseado em Variáveis Make) então use `%%DATADIR%%` diretamente em pkg-plist.

Quanto à escolha de compilar ports Java a partir do código fonte ou instalar diretamente a partir de uma distribuição binária, não há política definida no momento da escrita deste livro. No entanto, os membros do Projeto Java do FreeBSD encorajam os mantenedores de ports a terem seus ports compilados a partir do código fonte sempre que for possível.

Todos os recursos que foram apresentados nesta seção são implementados em bsd.java.mk. Se o port precisar de suporte Java mais sofisticado, por favor, primeiro dê uma olhada no log do bsd.java.mk no Subversion pois normalmente leva algum tempo para documentar os recursos mais recentes. Então, se o suporte necessário que estiver faltando for benéfico para muitos outros ports Java, sinta-se à vontade para discuti-lo na Lista de discussão do FreeBSD sobre Linguagem Java.

Embora haja uma categoria `Java` para PRs, isso refere-se ao esforço de portabilidade do JDK do projeto Java do FreeBSD. Portanto, envie o port Java na categoria `ports` como para qualquer outro port, a menos que o problema esteja relacionado a uma implementação do JDK ou ao bsd.java.mk.

Da mesma forma, existe uma política definida sobre as `CATEGORIAS` de um port Java, que é detalhada em Categorização.

6.16. Aplicações Web, Apache e PHP

6.16.1. Apache

Tabela 31. Variáveis para Ports Que Usam o Apache

`USE_APACHE`	O port requer o Apache. Valores possíveis: `yes` (obtém qualquer versão), `22`, `24`, `22 a 24`, `22+`, etc. A versão padrão do APACHE é `22`. Mais detalhes estão disponíveis em ports/Mk/bsd.apache.mk e em wiki.freebsd.org/Apache/.

APXS	Caminho completo para o binário apxs. Pode ser modificado no port.
HTTPD	Caminho completo para o binário httpd. Pode ser modificado no port.
APACHE_VERSION	A versão da instalação atual do Apache (variável somente leitura). Esta variável só está disponível após a inclusão de bsd.port.pre.mk. Valores possíveis: 22, 24.
APACHEMODDIR	Diretório para módulos Apache. Esta variável é automaticamente expandida em pkg-plist.
APACHEINCLUDEDIR	Diretório para cabeçalhos Apache. Esta variável é automaticamente expandida em pkg-plist.
APACHEETCDIR	Diretório para arquivos de configuração do Apache. Esta variável é automaticamente expandida em pkg-plist.

Tabela 32. Variáveis Úteis para Portar Módulos do Apache

MODULENAME	Nome do módulo. O valor padrão é PORTNAME. Exemplo: mod_hello
SHORTMODNAME	Nome abreviado do módulo. Automaticamente derivado de MODULENAME, mas pode ser substituído. Exemplo: hello
AP_FAST_BUILD	Use o apxs para compilar e instalar o módulo.
AP_GENPLIST	Também cria automaticamente um pkg-plist.
AP_INC	Adiciona um diretório ao caminho de pesquisa de cabeçalhos durante a compilação.
AP_LIB	Adiciona um diretório ao caminho de pesquisa de bibliotecas durante a compilação.
AP_EXTRAS	Flags adicionais para passar para o apxs.

6.16.2. Aplicações Web

Aplicações web devem ser instaladas em PREFIX/www/appname. Este caminho está disponível tanto no Makefile quanto no pkg-plist como WWWDIR e o caminho relativo para PREFIX está disponível no Makefile como WWWDIR_REL.

O usuário e o grupo do processo do servidor web estão disponíveis como WWWOWN e WWWGRP, no caso de a propriedade de alguns arquivos precisar ser alterada. Os valores padrão de ambos são www. Use WWWOWN?=myuser e WWWGRP?=mygroup se o port precisar de valores diferentes. Isso permite ao usuário substituí-los facilmente.

Use WWWOWN e WWWGRP com moderação. Lembre-se de que todos os arquivos para os quais o servidor web tem permissão de escrita, são um risco de segurança

esperando para acontecer.

Não insira o Apache como dependência, a menos que o aplicativo web precise explicitamente do Apache. Respeite que os usuários podem desejar executar um aplicativo web em um servidor web diferente do Apache.

6.16.3. PHP

Aplicações PHP declaram sua dependência a ele com `USES=php`. Veja `php` para maiores informações.

6.16.4. Módulos PEAR

Portar módulos PEAR é um processo muito simples.

Adicione `USES=pear` ao Makefile do port. O framework instalará os arquivos relevantes nos lugares certos e gerará automaticamente a lista no momento da instalação.

Exemplo 77. Exemplo de Makefile para Classes PEAR

```
PORTNAME=       Date
DISTVERSION=    1.4.3
CATEGORIES= devel www pear

MAINTAINER= example@domain.com
COMMENT=    PEAR Date and Time Zone Classes

USES=   pear

.include <bsd.port.mk>
```

Os módulos PEAR serão automaticamente flavorizados usando PHPflavors.

Se um `PEAR_CHANNEL` não padrão for usado, as dependências de compilação e de tempo de execução serão automaticamente adicionadas.

Módulos PEAR não precisam definir `PKGNAMESUFFIX`, é preenchido automaticamente usando `PEAR_PKGNAMEPREFIX`. Se um port precisar adicionar `PKGNAMEPREFIX`, também deve usar `PEAR_PKGNAMEPREFIX` para diferenciar entre diferentes flavors.

Módulos Horde

Da mesma forma, portar módulos Horde é um processo simples.

Adicione `USES=horde` ao Makefile do port . O framework instalará os arquivos relevantes nos lugares certos e gerará automaticamente a lista no momento da instalação.

As variáveis `USE_HORDE_BUILD` e `USE_HORDE_RUN` podem ser usadas para adicionar dependências de

tempo de compilação e de tempo de execução em outros módulos Horde. Veja Mk/Uses/horde.mk para uma lista completa dos módulos disponíveis.

Exemplo 78. Exemplo de Makefile para Módulos Horde

```
PORTNAME=   Horde_Core
DISTVERSION=    2.14.0
CATEGORIES= devel www pear

MAINTAINER= horde@FreeBSD.org
COMMENT=    Horde Core Framework libraries

OPTIONS_DEFINE= KOLAB SOCKETS
KOLAB_DESC= Enable Kolab server support
SOCKETS_DESC=   Depend on sockets PHP extension

USES=   horde
USE_PHP=    session

USE_HORDE_BUILD=    Horde_Role
USE_HORDE_RUN=  Horde_Role Horde_History Horde_Pack \
        Horde_Text_Filter Horde_View

KOLAB_USE=  HORDE_RUN=Horde_Kolab_Server,Horde_Kolab_Session
SOCKETS_USE=    PHP=sockets

.include <bsd.port.mk>
```

Como os módulos Horde também são módulos PEAR, eles também serão automaticamente aromatizados usando PHP flavors.

6.17. Usando Python

A Coleção de Ports suporta a instalação paralela de várias versões do Python. Os ports devem usar um interpretador `python`, de acordo com a configuração do usuário de `PYTHON_VERSION`. Mais proeminentemente, isso significa substituir o caminho para o executável `python` em scripts com o valor de `PYTHON_CMD`.

Ports que instalam arquivos sob `PYTHON_SITELIBDIR` devem usar o prefixo `pyXY-` no prefixo do nome do pacote, assim o nome do pacote irá incorporar a versão do Python em que estão instalados.

```
PKGNAMEPREFIX=  ${PYTHON_PKGNAMEPREFIX}
```

Tabela 33. Variáveis úteis para Ports que usam Python

`USES=python`	O port precisa do Python. A versão mínima necessária pode ser especificada com valores como `2.7+`. Os intervalos de versão também podem ser especificados separando dois números de versão com um traço: `USES=python:3.2-3.3`
`USE_PYTHON=distutils`	Use o distutils do Python para configurar, compilar e instalar. Isso é necessário quando o port vem com setup.py. Isso sobrescreve os targets `do-build` e `do-install` e também pode substituir `do-configure` se o `GNU_CONFIGURE` não estiver definido. Além disso, isso implica em `USE_PYTHON=flavors`.
`USE_PYTHON=autoplist`	Crie a lista de empacotamento automaticamente. Isso também requer que `USE_PYTHON=distutils` seja definido.
`USE_PYTHON=concurrent`	O port usará um prefixo exclusivo, normalmente `PYTHON_PKGNAMEPREFIX` para determinados diretórios, como `EXAMPLESDIR` e `DOCSDIR` e também irá acrescentar um sufixo, a versão python de `PYTHON_VER`, para os binários e scripts que serão instalados. Isso permite que os ports sejam instalados para diferentes versões do Python ao mesmo tempo, o que de outra forma instalaria arquivos conflitantes.
`USE_PYTHON=flavors`	O port não usa distutils, mas ainda suporta várias versões do Python. `.FLAVORS` será definido para as versões suportadas do Python. Veja `USES=python` e Flavors para maiores informações.
`USE_PYTHON=optsuffix`	Se a versão atual do Python não for a versão padrão, o port receberá `PKGNAMESUFFIX=${PYTHON_PKGNAMESUFFIX}`. É útil apenas com flavors.
`PYTHON_PKGNAMEPREFIX`	Usado como um `PKGNAMEPREFIX` para distinguir pacotes para diferentes versões do Python. Exemplo: `py27-`
`PYTHON_SITELIBDIR`	Local da árvore site-packages, que contém o caminho de instalação do Python (geralmente `LOCALBASE`). A `PYTHON_SITELIBDIR` pode ser muito útil ao instalar módulos Python.

PYTHONPREFIX_SITELIBDIR	A variante PREFIX-clean do PYTHON_SITELIBDIR. Sempre use %%PYTHON_SITELIBDIR%% no pkg-plist quando possível. O valor padrão de %%PYTHON_SITELIBDIR%% é lib/python%%PYTHON_VERSION%%/site-packages
PYTHON_CMD	Linha de comando do interpretador Python, incluindo o número da versão.

Tabela 34. Assistentes do Módulo de Dependências do Python

PYNUMERIC	Linha de dependência para extensão numérica.
PYNUMPY	Linha de dependência para a nova extensão numérica, numpy. (PYNUMERIC foi descontinuado pelo fornecedor upstream).
PYXML	Linha de dependência para a extensão XML (não é necessária para o Python 2.0 e superior, pois também está na distribuição base).
PY_ENUM34	Dependência condicional do devel/py-enum34 dependendo da versão do Python.
PY_ENUM_COMPAT	Dependência condicional do devel/py-enum-compat dependendo da versão do Python.
PY_PATHLIB	Dependência condicional do devel/py-pathlib dependendo da versão do Python.
PY_IPADDRESS	Dependência condicional do net/py-ipaddress dependendo da versão do Python.
PY_FUTURES	Dependência condicional do devel/py-futures dependendo da versão do Python.

Uma lista completa das variáveis disponíveis pode ser encontrada em /usr/ports/Mk/Uses/python.mk.

Todas as dependências para ports Python usando Python flavors (quer com USE_PYTHON=distutils ou USE_PYTHON=flavors) deve ter o flavor Python anexado à sua origem usando @${PY_FLAVOR}. Veja Makefile para um Módulo Python Simples.

Exemplo 79. Makefile para um Módulo Python Simples

```
PORTNAME=   sample
DISTVERSION=    1.2.3
CATEGORIES= devel

MAINTAINER= john@doe.tld
COMMENT=    Python sample module
```

```
RUN_DEPENDS=    ${PYTHON_PKGNAMEPREFIX}six>0:devel/py-six@${PY_FLAVOR}

USES=       python
USE_PYTHON= autoplist distutils

.include <bsd.port.mk>
```

Algumas aplicações Python afirmam ter suporte a DESTDIR (que seria necessário para fazer o staging), mas ele está quebrado (Mailman até a versão 2.1.16, por exemplo). Isso pode ser contornado, recompilando os scripts. Isso pode ser feito, por exemplo, no target post-build. Assumindo que os scripts Python devem estar em PYTHONPREFIX_SITELIBDIR após a instalação, esta solução pode ser aplicada:

```
(cd ${STAGEDIR}${PREFIX} \
  && ${PYTHON_CMD} ${PYTHON_LIBDIR}/compileall.py \
   -d ${PREFIX} -f ${PYTHONPREFIX_SITELIBDIR:S;${PREFIX}/;;})
```

Isso recompila os fontes com um caminho relativo ao diretório de stage e acrescenta o valor de PREFIX para o nome do arquivo gravado no arquivo bytecode de saída por -d. O -f é necessário para forçar a recompilação e o :S;${PREFIX}/;; remove prefixos do valor de PYTHONPREFIX_SITELIBDIR para torná-lo relativo ao PREFIX.

6.18. Usando Tcl/Tk

A Coleção de Ports suporta a instalação paralela de múltiplas versões do Tcl/Tk. Ports devem tentar suportar pelo menos a versão padrão do Tcl/Tk e superior com USES=tcl. É possível especificar a versão desejada do tcl anexando :xx, por exemplo, USES=tcl:85.

Tabela 35. As variáveis read only muito úteis para Ports que usam Tcl/Tk

TCL_VER	versão major.minor escolhida do Tcl
TCLSH	caminho completo do interpretador Tcl
TCL_LIBDIR	caminho das bibliotecas Tcl
TCL_INCLUDEDIR	caminho dos arquivos de cabeçalho C do Tcl
TK_VER	versão major.minor escolhida do Tk
WISH	caminho completo do interpretador Tk
TK_LIBDIR	caminho das bibliotecas Tk
TK_INCLUDEDIR	caminho dos arquivos de cabeçalho C do Tk

Veja o USES=tcl e USES=tk do Usando Macros USES para uma descrição completa dessas variáveis. Uma lista completa dessas variáveis está disponível em /usr/ports/Mk/Uses/tcl.mk.

6.19. Usando Ruby

Tabela 36. Variáveis Úteis para Ports Que Usam Ruby

Variável	Descrição
`USE_RUBY`	Adiciona dependências de build e run no Ruby.
`USE_RUBY_EXTCONF`	O port utiliza extconf.rb para configurar.
`USE_RUBY_SETUP`	O port utiliza setup.rb para configurar.
`RUBY_SETUP`	Substitui o nome do script de configuração do setup.rb. Outro valor comum é install.rb.

Esta tabela mostra as variáveis selecionadas disponíveis para os autores dos ports através da infraestrutura de ports. Essas variáveis são usadas para instalar arquivos em seus locais apropriados. Use-os em pkg-plist tanto quanto possível. Não redefina essas variáveis no port.

Tabela 37. Variáveis Somente Leitura Selecionadas para Ports Que Usam Ruby

Variável	Descrição	Exemplo de valor
`RUBY_PKGNAMEPREFIX`	Usado como um `PKGNAMEPREFIX` para distinguir pacotes para diferentes versões do Ruby.	`ruby19-`
`RUBY_VERSION`	Versão completa do Ruby na forma de `x.y.z[.p]`.	`1.9.3.484`
`RUBY_SITELIBDIR`	Caminho de instalação de bibliotecas independentes de arquitetura.	`/usr/local/lib/ruby/site_ruby/1.9`
`RUBY_SITEARCHLIBDIR`	Caminho de instalação das bibliotecas dependentes de arquitetura.	`/usr/local/lib/ruby/site_ruby/1.9/amd64-freebsd10`
`RUBY_MODDOCDIR`	Caminho de instalação da documentação do módulo.	`/usr/local/shared/doc/ruby19/patsy`
`RUBY_MODEXAMPLESDIR`	Caminho de instalação dos exemplos do módulo.	`/usr/local/shared/examples/ruby19/patsy`

Uma lista completa das variáveis disponíveis pode ser encontrada em /usr/ports/Mk/bsd.ruby.mk.

6.20. Usando SDL

O `USE_SDL` é usado para auto configurar as dependências para os ports que usam uma biblioteca baseada em SDL como o devel/sdl12 e o graphics/sdl_image.

Estas bibliotecas SDL para a versão 1.2 são reconhecidas:

- sdl: devel/sdl12
- console: devel/sdl_console

- gfx: graphics/sdl_gfx
- image: graphics/sdl_image
- mixer: audio/sdl_mixer
- mm: devel/sdlmm
- net: net/sdl_net
- pango: x11-toolkits/sdl_pango
- sound: audio/sdl_sound
- ttf: graphics/sdl_ttf

Estas são as bibliotecas SDL para a versão 2.0 reconhecidas:

- sdl: devel/sdl20
- gfx: graphics/sdl2_gfx
- image: graphics/sdl2_image
- mixer: audio/sdl2_mixer
- net: net/sdl2_net
- ttf: graphics/sdl2_ttf

Portanto, se um port tiver uma dependência do net/sdl_net e do audio/sdl_mixer, a sintaxe será:

```
USE_SDL=    net mixer
```

A dependência devel/sdl12, a qual é exigida por net/sdl_net e audio/sdl_mixer, é automaticamente adicionada também.

Usar `USE_SDL` com entradas para o SDL 1.2, irá automaticamente:

- Adicionar uma dependência em sdl12-config para `BUILD_DEPENDS`
- Adicionar a variável `SDL_CONFIG` em `CONFIGURE_ENV`
- Adicionar as dependências das bibliotecas selecionadas ao `LIB_DEPENDS`

Usar `USE_SDL` com entradas para o SDL 2.0, irá automaticamente:

- Adicionar uma dependência em sdl2-config para `BUILD_DEPENDS`
- Adicionar a variável `SDL2_CONFIG` ao `CONFIGURE_ENV`
- Adicionar as dependências das bibliotecas selecionadas ao `LIB_DEPENDS`

6.21. Usando wxWidgets

Esta seção descreve o status das bibliotecas wxWidgets na árvore de ports e sua integração com o sistema de ports.

6.21.1. Introdução

Existem muitas versões das bibliotecas do wxWidgets que entram em conflito entre elas (instalam arquivos com o mesmo nome). Na árvore de ports este problema foi resolvido instalando cada versão sob um nome diferente usando sufixos de número de versão.

A desvantagem óbvia disso é que cada aplicativo precisa ser modificado para encontrar a versão esperada. Felizmente, a maioria dos aplicativos chama o script `wx-config` para determinar os sinalizadores necessários para o compilador e o vinculador. O script é nomeado de maneira diferente para cada versão disponível. A maioria dos aplicativos respeita uma variável de ambiente ou aceita um argumento de configuração para especificar o script `wx-config` que deve ser chamado. Caso contrário, eles têm que ser corrigidos.

6.21.2. Seleção de Versão

Para fazer o port usar uma versão específica do wxWidgets existem duas variáveis disponíveis para definir (se apenas uma for definida, a outra será definida para um valor padrão):

Tabela 38. Variáveis para Selecionar as Versões do wxWidgets

Variável	Descrição	Valor padrão
`USE_WX`	Lista de versões que o port pode usar	Todas as versões disponíveis
`USE_WX_NOT`	Lista de versões que o port não pode usar	Nenhum

As versões disponíveis do wxWidgets e os ports correspondentes na árvore são:

Tabela 39. Versões Disponíveis do wxWidgets

Versão	Port
`2.8`	x11-toolkits/wxgtk28
`3.0`	x11-toolkits/wxgtk30

As variáveis em Variáveis para Selecionar as Versões do wxWidgets podem ser definidas para uma ou mais dessas combinações separadas por espaços:

Tabela 40. Especificações de Versão do wxWidgets

Descrição	Exemplo
Versão única	`2.8`
Range ascendente	`2.8+`
Range descendente	`3.0-`
Range total (deve ser crescente)	`2.8-3.0`

Também existem algumas variáveis para selecionar as versões preferidas entre as disponíveis. Elas podem ser configuradas para uma lista de versões, as primeiras terão maior prioridade.

Tabela 41. Variáveis para selecionar as versões preferidas do wxWidgets

Nome	Desenhado para
`WANT_WX_VER`	o port
`WITH_WX_VER`	o usuário

6.21.3. Seleção de Componentes

Existem outras aplicações que, apesar de não serem bibliotecas wxWidgets, estão relacionadas a eles. Estas aplicações podem ser especificadas em `WX_COMPS`. Estes componentes estão disponíveis:

Tabela 42. Componentes wxWidgets Disponíveis

Nome	Descrição	Restrição de versão
`wx`	biblioteca principal	nenhum
`contrib`	bibliotecas contribuídas	`none`
`python`	wxPython(ligações Python)	`2.8-3.0`

O tipo de dependência pode ser selecionado para cada componente, adicionando-se um sufixo separado por um ponto-e-vírgula. Se não estiver presente, será usado um tipo padrão (veja Tipos de Dependência Padrão do wxWidgets). Estes tipos estão disponíveis:

Tabela 43. Tipos de Dependências wxWidgets Disponíveis

Nome	Descrição
`build`	Componente é necessário para a compilação, equivalente a `BUILD_DEPENDS`
`run`	O componente é necessário para execução, equivalente a `RUN_DEPENDS`
`lib`	O componente é necessário para a compilação e execução, equivalente a `LIB_DEPENDS`

Os valores padrão para os componentes estão detalhados nesta tabela:

Tabela 44. Tipos de Dependência Padrão do wxWidgets

Componente	Tipo de dependência
`wx`	`lib`
`contrib`	`lib`
`python`	`run`
`mozilla`	`lib`
`svg`	`lib`

Exemplo 80. Selecionando Componentes wxWidgets

Este fragmento corresponde a um port que usa wxWidgets versão `2.4` e suas bibliotecas contribuídas.

```
USE_WX=     2.8
WX_COMPS=   wx contrib
```

6.21.4. Detectando Versões Instaladas

Para detectar uma versão instalada, defina `WANT_WX`. Se não estiver definido para uma versão específica, os componentes terão um sufixo de versão. O `HAVE_WX` será preenchido após a detecção.

Exemplo 81. Detectando as versões instaladas wxWidgets e seus componentes

Este fragmento pode ser usado em um port que usa wxWidgets se estiver instalado ou uma opção estiver selecionada.

```
WANT_WX=    yes

.include <bsd.port.pre.mk>

.if defined(WITH_WX) || !empty(PORT_OPTIONS:MWX) || !empty(HAVE_WX:Mwx-2.8)
USE_WX=         2.8
CONFIGURE_ARGS+=    --enable-wx
.endif
```

Este fragmento pode ser usado em um port que permite suporte ao wxPython se ele estiver instalado ou se uma opção for selecionada, em adição ao wxWidgets, ambas nas versões `2.8`.

```
USE_WX=     2.8
WX_COMPS=   wx
WANT_WX=    2.8

.include <bsd.port.pre.mk>

.if defined(WITH_WXPYTHON) || !empty(PORT_OPTIONS:MWXPYTHON) ||
!empty(HAVE_WX:Mpython)
WX_COMPS+=      python
CONFIGURE_ARGS+=    --enable-wxpython
.endif
```

6.21.5. Variáveis Definidas

Estas variáveis estão disponíveis no port (depois de definir uma de Variáveis para Selecionar as Versões do wxWidgets).

Tabela 45. Variáveis definidas para ports que usam wxWidgets

Nome	Descrição
WX_CONFIG	O caminho para o script wxWidgets wx-config (com nome diferente)
WXRC_CMD	O caminho para o programa wxWidgets wxrc (com nome diferente)
WX_VERSION	A versão do wxWidgets que será usada (por exemplo,2.6)

6.21.6. Processando em bsd.port.pre.mk

Defina WX_PREMK para ser capaz de usar as variáveis logo após a inclusão do bsd.port.pre.mk.

Ao definir WX_PREMK, a versão, dependências, componentes e variáveis definidas não serão alteradas mesmo se alterado as variáveis do port wxWidgets *depois de* incluir o bsd.port.pre.mk.

Exemplo 82. Usando Variáveis nos Comandos wxWidgets

Este fragmento ilustra o uso de WX_PREMK executando o script wx-config para obter a string de versão completa, atribuí-lo a uma variável e passá-lo para o programa.

```
USE_WX=     2.8
WX_PREMK=   yes

.include <bsd.port.pre.mk>

.if exists(${WX_CONFIG})
VER_STR!=   ${WX_CONFIG} --release

PLIST_SUB+= VERSION="${VER_STR}"
.endif
```

As variaveis wxWidgets podem ser usadas com segurança em comandos quando estão dentro de targets sem a necessidade de WX_PREMK.

6.21.7. Argumentos Adicionais do configure

Alguns scripts GNU configure não podem encontrar wxWidgets com apenas o conjunto de variáveis de ambiente WX_CONFIG, exigindo argumentos adicionais. WX_CONF_ARGS pode ser usado para fornecê-los.

Tabela 46. Valores Legais para WX_CONF_ARGS

Valor possível	Argumento resultante
absolute	--with-wx-config=${WX_CONFIG}

Valor possível	Argumento resultante
`relative`	`--with-wx=${LOCALBASE} --with-wx-config=${WX_CONFIG:T}`

6.22. Usando Lua

Esta seção descreve o status das bibliotecas Lua na árvore de ports e sua integração com o sistema de ports.

6.22.1. Introdução

Existem muitas versões das bibliotecas Lua e interpretadores correspondentes, que entram em conflito entre eles (instalam arquivos com o mesmo nome). Na árvore de ports este problema foi resolvido instalando cada versão sob um nome diferente usando sufixos de número de versão.

A desvantagem óbvia disso é que cada aplicativo precisa ser modificado para encontrar a versão esperada. Mas isto pode ser resolvido adicionando alguns sinalizadores adicionais ao compilador e ao linker.

Aplicativos que usam Lua normalmente devem ser compilados para apenas uma versão. No entanto, os módulos carregáveis para Lua são compilados em flavor separado para cada versão Lua que eles suportam, e as dependências de tais módulos devem especificar o flavor usando o sufixo `@${LUA_FLAVOR}` no caminho do port.

6.22.2. Seleção de Versão

Um port usando Lua deve ter uma linha dessa forma:

```
USES=   lua
```

Se uma versão específica de Lua, ou intervalo de versões for necessária, ela pode ser especificada como um parâmetro na forma `XY` (que pode ser usado várias vezes), `XY+`, `-XY`, ou `XY-ZA`. A versão padrão do Lua definida por meio do `DEFAULT_VERSIONS` será usada se cair no intervalo solicitado, caso contrário, a versão solicitada mais próxima do padrão será usada. Por exemplo:

```
USES=   lua:52-53
```

Observe que nenhuma tentativa é feita para ajustar a seleção da versão com base na presença de qualquer versão Lua já instalada.

> A forma `XY+` de especificação de versão não deve ser usada sem consideração cuidadosa; a API Lua muda consideravelmente em todas as versões, e ferramentas de configuração como CMake ou Autoconf frequentemente não funcionarão em versões futuras do Lua até ser atualizado para isso.

6.22.3. Flags de Configuração e Compilador

Software that uses Lua may have been written to auto-detect the Lua version in use. In general ports should override this assumption, and force the use of the specific Lua version selected as described above. Depending on the software being ported, this might require any or all of:

- Usando `LUA_VER` como parte de um parâmetro para o script de configuração do software via `CONFIGURE_ARGS` ou `CONFIGURE_ENV` (ou equivalente para outros sistemas de compilação);
- Adicionando `-I${LUA_INCDIR}`, `-L${LUA_LIBDIR}`, e `-llua-${LUA_VER}` para `CFLAGS`, `LDFLAGS`, `LIBS` respectivamente, conforme apropriado;
- Altere a configuração do software ou arquivos de compilação para selecionar a versão correta.

6.22.4. Flavors de Versão

Um port que instala um módulo Lua (em vez de um aplicativo que simplesmente faz uso do Lua) deve compilar um flavor separado para cada versão do Lua suportada . Isso é feito adicionando o parâmetro `module`:

```
USES=   lua:module
```

Um número de versão ou intervalo de versões também pode ser especificado; use uma vírgula para separar os parâmetros.

Uma vez que cada flavor deve ter um nome de pacote diferente, a variável `LUA_PKGNAMEPREFIX` é fornecida e será definida com um valor apropriado; o uso pretendido é:

```
PKGNAMEPREFIX=  ${LUA_PKGNAMEPREFIX}
```

Ports de módulo normalmente devem instalar arquivos apenas em `LUA_MODLIBDIR`, `LUA_MODSHAREDIR`, `LUA_DOCSDIR`, e `LUA_EXAMPLESDIR`>, todos os quais estão definidos para se referir a subdiretórios específicos da versão. A instalação de quaisquer outros arquivos deve ser feita com cuidado para evitar conflitos entre as versões.

Um port (diferente de um módulo Lua) que deseja compilar um pacote separado para cada versão Lua deve usar o parâmetro `flavors`:

```
USES=   lua:flavors
```

Isso funciona da mesma maneira que o parâmetro `module` descrito acima, mas sem a suposição de que o pacote deve ser documentado como um módulo Lua (então `LUA_DOCSDIR` e `LUA_EXAMPLESDIR` não são definidos por padrão). No entanto, o port pode escolher definir `LUA_DOCSUBDIR` como um nome de subdiretório adequado (geralmente o `PORTNAME` do port, desde que não entre em conflito com o `PORTNAME` de qualquer módulo), caso em que a estrutura definirá `LUA_DOCSDIR` e `LUA_EXAMPLESDIR`.

Tal como acontece com os ports de módulo, um port com flavor deve evitar a instalação de arquivos que entrariam em conflito entre as versões. Normalmente, isso é feito adicionando `LUA_VER_STR`

como um sufixo para nomes de programas (por exemplo, usando USES=uniquefiles), e de outra forma usando LUA_VER ou LUA_VER_STR como parte de quaisquer outros arquivos ou subdiretórios usados fora de LUA_MODLIBDIR e LUA_MODSHAREDIR.

6.22.5. Variáveis Definidas

Essas variáveis estão disponíveis no port.

Tabela 47. Variáveis Definidas para Ports Que Usam Lua

Nome	Descrição
LUA_VER	A versão Lua que será usada (por exemplo,5,1)
LUA_VER_STR	A versão Lua sem os pontos (por exemplo,51)
LUA_FLAVOR	O nome do flavor correspondente à versão selecionada Lua, a ser usado para especificar dependências
LUA_BASE	O prefixo que deve ser usado para localizar o Lua (e componentes) que já estão instalados
LUA_PREFIX	O prefixo onde o Lua (e os seus componentes) são instalados por este port
LUA_INCDIR	O diretório onde os arquivos header do Lua estão instalados
LUA_LIBDIR	O diretório onde as bibliotecas Lua são instaladas
LUA_REFMODLIBDIR	O diretório no qual as bibliotecas dos módulos Lua (.so) que já estão instalados podem ser encontrados
LUA_REFMODSHAREDIR	O diretório no qual os módulos Lua (.lua) que já estão instalados podem ser encontrados
LUA_MODLIBDIR	O diretório no qual as bibliotecas dos módulos Lua (.so) serão instalados por este port
LUA_MODSHAREDIR	O diretório no qual os módulos Lua (.lua) serão instalados por este port
LUA_PKGNAMEPREFIX	O prefixo do nome do pacote usado por módulos Lua
LUA_CMD	O nome do interpretador Lua (exemplo lua53)
LUAC_CMD	O nome do compilador Lua (exemplo luac53)

Essas variáveis adicionais estão disponíveis para ports que especificaram o parâmetro module:

Tabela 48. Variáveis Definidas para Ports de Módulos Lua

Nome	Descrição
LUA_DOCSDIR	o diretório no qual a documentação do módulo deve ser instalada.
LUA_EXAMPLESDIR	o diretório no qual os arquivos de exemplo do módulo devem ser instalados.

6.22.6. Exemplos

Exemplo 83. Makefile para uma aplicação que utiliza Lua

Este exemplo mostra como fazer referência a um módulo Lua necessário em tempo de execução. Observe que a referência deve especificar um flavor.

```
PORTNAME=   sample
DISTVERSION=    1.2.3
CATEGORIES= whatever

MAINTAINER= john@doe.tld
COMMENT=    Sample

RUN_DEPENDS=    ${LUA_REFMODLIBDIR}/lpeg.so:devel/lua-lpeg@${LUA_FLAVOR}

USES=       lua

.include <bsd.port.mk>
```

Exemplo 84. Makefile para módulo simples de Lua

```
PORTNAME=   sample
DISTVERSION=    1.2.3
CATEGORIES= whatever
PKGNAMEPREFIX=  ${LUA_PKGNAMEPREFIX}

MAINTAINER= john@doe.tld
COMMENT=    Sample

USES=       lua:module

DOCSDIR=    ${LUA_DOCSDIR}

.include <bsd.port.mk>
```

6.23. Usando iconv

Após 2013-10-08 (r254273), o FreeBSD 10-CURRENT e as versões mais recentes têm um iconv nativo no sistema operacional. Em versões anteriores, o converters/libiconv era usado como dependência.

Para softwares que precisam do iconv, defina USES=iconv. As versões do FreeBSD antes do 10-CURRENT em 2013-08-13 (r254273) não tem um iconv nativo. Nestas versões anteriores, uma dependência do converters/libiconv será adicionada automaticamente.

Quando um port define USES=iconv, estas variáveis estarão disponíveis:

Nome da variável	Propósito	Valor antes do FreeBSD 10-CURRENT 254273(2013-08-13)	Valor após o FreeBSD 10-CURRENT 254273(2013-08-13)
ICONV_CMD	Diretório onde o binário iconv reside	${LOCALBASE}/bin/iconv	/usr/bin/iconv
ICONV_LIB	argumento do ld para vincular ao libiconv (se necessário)	-liconv	(vazio)
ICONV_PREFIX	Diretório onde a implementação do iconv reside (útil para configurar scripts)	${LOCALBASE}	/usr
ICONV_CONFIGURE_ARG	Argumento de configuração pré-configurado para scripts de configuração	--with-libiconv -prefix=${LOCALBASE}	(vazio)
ICONV_CONFIGURE_BASE	Argumento de configuração pré-configurado para scripts de configuração	--with -libiconv=${LOCALBASE}	(vazio)

Esses dois exemplos preenchem automaticamente as variáveis com o valor correto para sistemas usando respectivamente o converters/libiconv ou o iconv nativo:

Exemplo 85. Simples uso do iconv

```
USES=       iconv
LDFLAGS+=   -L${LOCALBASE}/lib ${ICONV_LIB}
```

Exemplo 86. Uso do iconv com configure

```
USES=       iconv
```

```
CONFIGURE_ARGS+=${ICONV_CONFIGURE_ARG}
```

Como mostrado acima, a variável `ICONV_LIB` estará vazia quando um `iconv` nativo estiver presente. Isso pode ser usado para detectar o `iconv` nativo e responder adequadamente.

Às vezes um programa tem um argumento `ld` ou caminho de pesquisa codificado em um Makefile ou no script configure. Essa abordagem pode ser usada para resolver esse problema:

Exemplo 87. Corrigindo Hardcoded `-liconv`

```
USES=       iconv

post-patch:
    @${REINPLACE_CMD} -e 's/-liconv/${ICONV_LIB}/' ${WRKSRC}/Makefile
```

Em alguns casos, é necessário definir valores alternativos ou executar operações dependendo se há um `iconv` nativo. O bsd.port.pre.mk deve ser incluído antes de testar o valor de `ICONV_LIB`:

Exemplo 88. Verificando Disponibilidade do `iconv` Nativo

```
USES=       iconv

.include <bsd.port.pre.mk>

post-patch:
.if empty(ICONV_LIB)
    # native iconv detected
    @${REINPLACE_CMD} -e 's|iconv||' ${WRKSRC}/Config.sh
.endif

.include <bsd.port.post.mk>
```

6.24. Usando o Xfce

Ports que precisam de bibliotecas ou aplicações Xfce, utilizam `USES=xfce`.

Dependencias específicas de bibliotecas e aplicativos Xfce são definidas com valores atribuídos a `USE_XFCE`. Eles são definidos em /usr/ports/Mk/Uses/xfce.mk. Os valores possíveis são:

Valores de `USE_XFCE`

garcon

sysutils/garcon

libexo

x11/libexo

libgui

x11-toolkits/libxfce4gui

libmenu

x11/libxfce4menu

libutil

x11/libxfce4util

painel

x11-wm/xfce4-panel

thunar

x11-fm/thunar

xfconf

x11/xfce4-conf

Exemplo 89. Exemplo de `USES=xfce`

```
USES=       xfce
USE_XFCE=   libmenu
```

Exemplo 90. Usando os Próprios Widgets GTK2 do Xfce

Neste exemplo, o aplicativo portado usa os widgets específicos do GTK2, o x11/libxfce4menu e o x11/xfce4-conf.

```
USES=       xfce:gtk2
USE_XFCE=   libmenu xfconf
```

Os componentes Xfce incluídos dessa maneira incluirão automaticamente todas as dependências necessárias. Não é mais necessário especificar a lista inteira. Se o port precisar apenas de x11-wm/xfce4-panel, use:

```
USES=       xfce
USE_XFCE=   panel
```

Não há necessidade de listar os componentes que o x11-wm/xfce4-panel precisa para ele mesmo, desta forma:

```
USES=       xfce
USE_XFCE=   libexo libmenu libutil panel
```

Contudo, os componentes Xfce e as dependências do port que não dependem do Xfce devem ser incluídas explicitamente. Não conte com um componente Xfce para fornecer uma sub-dependência diferente de si para o port principal.

6.25. Usando Bancos de Dados

Utilize uma das macros `USES` de Banco de Dados de Macros `USES` para adicionar a dependência de um banco de dados.

Tabela 49. Banco de Dados de Macros `USES`

Base de Dados	Macro USES
Berkeley DB	`bdb`
MariaDB, MySQL, Percona	`mysql`
PostgreSQL	`pgsql`
SQLite	`sqlite`

Exemplo 91. Usando o Berkeley DB 6

```
USES=   bdb:6
```

Veja `bdb` para maiores informações.

Exemplo 92. Usando MySQL

Quando um port precisa da biblioteca cliente do MySQL, adicione

```
USES=   mysql
```

Veja `mysql` para mais informações.

Exemplo 93. Usando PostgreSQL

Quando um port precisar do servidor PostgreSQL versão 9.6 ou posterior, adicione

```
USES=       pgsql:9.6+
WANT_PGSQL= server
```

Veja `pgsql` para mais informações.

Exemplo 94. Usando SQLite 3

```
USES=   sqlite:3
```

Veja `sqlite` para mais informações.

6.26. Iniciando e Parando Serviços (com scripts `rc`)

Os scripts rc.d são usados para iniciar serviços na inicialização do sistema e para fornecer aos administradores uma maneira padrão de parar, iniciar e reiniciar o serviço. Ports se integram ao sistema de estrutura do rc.d. Detalhes sobre seu uso podem ser encontrados no capitulo sobre rc.d do handbook. A explicação detalhada dos comandos disponíveis é fornecida em rc(8) e rc.sub(8). Finalmente, existe um artigo sobre aspectos práticos do sistema de scripts do rc.d.

Com um port mítico chamado *doorman*, o qual precisa iniciar um daemon *doormand*. Adicione o seguinte ao Makefile:

```
USE_RC_SUBR=    doormand
```

Vários scripts podem ser listados e serão instalados. Os scripts devem ser colocados no subdiretório files e um sufixo `.in` deve ser adicionado ao nome do arquivo. Expansões padrões `SUB_LIST` serão executadas neste arquivo. Usar as expansões `%%PREFIX%%` e `%%LOCALBASE%%` também é fortemente encorajado. Veja mais sobre a `SUB_LIST` na seção relevante.

A partir do FreeBSD 6.1-RELEASE, scripts locais rc.d (incluindo aqueles instalados pelos ports) estão incluídos no rcorder(8) do sistema base.

Um exemplo simples de script rc.d para iniciar o daemon doormand:

```
#!/bin/sh

# $FreeBSD: head/pt_BR.ISO8859-1/books/porters-handbook/book.xml 54410 2020-08-05
22:13:01Z dbaio $
#
# PROVIDE: doormand
# REQUIRE: LOGIN
# KEYWORD: shutdown
#
# Add these lines to /etc/rc.conf.local or /etc/rc.conf
# to enable this service:
#
# doormand_enable (bool):   Set to NO by default.
#                Set it to YES to enable doormand.
```

```
# doormand_config (path):   Set to %%PREFIX%%/etc/doormand/doormand.cf
#                 by default.

. /etc/rc.subr

name=doormand
rcvar=doormand_enable

load_rc_config $name

: ${doormand_enable:="NO"}
: ${doormand_config="%%PREFIX%%/etc/doormand/doormand.cf"}

command=%%PREFIX%%/sbin/${name}
pidfile=/var/run/${name}.pid

command_args="-p $pidfile -f $doormand_config"

run_rc_command "$1"
```

A menos que haja uma boa razão para iniciar o serviço mais cedo, ou ele seja executado como um usuário específico (diferente de root), todos os scripts de ports devem usar:

```
REQUIRE: LOGIN
```

Se o script de inicialização iniciar um daemon que deve ser desligado, o seguinte acionará uma parada do serviço no desligamento do sistema:

```
KEYWORD: shutdown
```

Se o script não está iniciando um serviço persistente, isso não é necessário.

Para os elementos de configuração opcional o estilo "=" de atribuição de variável padrão é preferível ao estilo ":=", já que o primeiro define um valor padrão apenas se a variável não estiver definida, e o segundo define um se a variável não está definida *ou* se ela é nula. Um usuário pode muito bem incluir algo como:

```
doormand_flags=""
```

no seu rc.conf.local, e uma substituição de variável usando ":=" substituirá inadequadamente a intenção do usuário. A variável `_enable` não é opcional e deve usar o ":" por padrão.

Os Ports *não devem* iniciar e parar seus serviços durante a instalação e desinstalação. Não abuse das keywords plist descritas em @preexec command, @postexec command, @preunexec command, @postunexec command executando comandos que modificam o sistema em execução, incluindo iniciar ou

interromper serviços.

6.26.1. Pre-Commit Checklist

Antes de contribuir um port com um script rc.d, e mais importante, antes de realizar o commit de um, por favor consulte esta lista de verificação para ter certeza de que ele está pronto.

O port devel/rclint pode verificar a maioria destes itens, mas não substitui uma revisão adequada.

1. Se este é um novo arquivo, ele tem uma extensão .sh? Se assim for, isso deve ser mudado para apenas file.in uma vez que os arquivos rc.d não podem terminar com essa extensão.
2. O arquivo tem uma tag `$FreeBSD: head/pt_BR.ISO8859-1/books/porters-handbook/book.xml 54410 2020-08-05 22:13:01Z dbaio $`?
3. O nome do arquivo (menos .in), a linha `PROVIDE` e `$` *name* são as mesmas? O nome do arquivo ao corresponder com o `PROVIDE` irá facilitar a depuração, especialmente para problemas de rcorder(8). Combinar o nome do arquivo e o `$` *name* torna mais fácil descobrir quais variáveis são relevantes no rc.conf[.local]. Isso também é uma política para todos os novos scripts, incluindo aqueles no sistema base.
4. A linha `REQUIRE` está definida para `LOGIN`? Isso é obrigatório para scripts que são executados como um usuário não root. Se ele for executado como root, há uma boa razão para ele ser executado antes de `LOGIN`? Caso contrário, ele deve ser executado depois para que os scripts locais possam ser agrupados em um ponto no rcorder(8) depois que quase tudo no sistema base já estiver rodando.
5. O script inicia um serviço persistente? Em caso afirmativo, ele deve ter o `KEYWORD: shutdown`.
6. Certifique-se de que não há um `KEYWORD: FreeBSD` presente. Isto não foi necessário nem desejável durante anos. Isto também é uma indicação de que o novo script foi copiado/colado de um script antigo, portanto, um cuidado extra deve ser dado à revisão.
7. Se o script usa uma linguagem interpretada como o `perl`, o `python` ou o `ruby`, certifique-se de que o `command_interpreter` está definido adequadamente, por exemplo, para o Perl, adicione `PERL=${PERL}` para a `SUB_LIST` e utilize `%%PERL%%`. De outra forma,

   ```
   # service name stop
   ```

   provavelmente não funcionará corretamente. Consulte service(8) para maiores informações.
8. Todas as ocorrências de /usr/local foram substituídas por `%%PREFIX%%`?
9. As atribuições das variáveis padrão vêm depois de `load_rc_config`?
10. Existem atribuições padrões para sequências vazias? Elas devem ser removidas, mas verifique se a opção está documentada nos comentários na parte superior do arquivo.
11. As variáveis definidas estão realmente sendo utilizadas no script?
12. As opções listadas no padrão *name* `_flags` são realmente obrigatórias? Se assim for, elas devem estar em `command_args`. A opção `-d` é uma flag vermelha (com o perdão do trocadilho)

aqui, já que geralmente é a opção de "daemonizar" o processo e, portanto, é realmente obrigatório.

13. O `name_flags` nunca deve ser incluído em `command_args` (e vice-versa, embora esse erro seja menos comum).
14. O script executa qualquer código incondicionalmente? Isso é desaprovado. Normalmente, essas coisas devem ser tratadas através de um `start_precmd`.
15. Todos os testes booleanos devem usar a função `checkyesno`. Nenhum teste deve usar `[Yy][Ee][Ss]`, etc.
16. Se houver um loop (por exemplo, esperando que algo inicie), ele tem um contador para terminar o loop? Não queremos que a inicialização seja bloqueada para sempre se houver um erro.
17. O script cria arquivos ou diretórios que precisam de permissões específicas, por exemplo, um pid que precisa ser de propriedade do usuário que executa o processo? Em vez da rotina tradicional touch(1)/chown(8)/chmod(1), considere usar install(1) com os argumentos de linha de comando apropriados para fazer todo o procedimento com um passo.

6.27. Adicionando Usuários e Grupos

Alguns ports exigem que uma conta de usuário específica esteja presente, geralmente para daemons executados como esse usuário. Para esses ports, escolha um UID *único* entre 50 a 999 e registre-o em ports/UIDs (para usuários) e em ports/GIDs(para grupos). A identificação única deve ser a mesma para usuários e grupos.

Por favor, inclua um patch para estes dois arquivos quando for necessário que um novo usuário ou grupo seja criado para o port.

Então use `USERS` e `GROUPS` dentro do Makefile e o usuário será criado automaticamente ao instalar o port.

```
USERS=  pulse
GROUPS= pulse pulse-access pulse-rt
```

A lista atual de UIDs e GIDs reservados pode ser encontrada em ports/UIDs e ports/GIDs.

6.28. Ports que Dependem dos Fontes do kernel

Alguns ports (como módulos carregáveis do kernel) precisam dos arquivos fonte do kernel para que o port possa ser compilado. Aqui está a maneira correta de determinar se o usuário os instalou:

```
USES=   kmod
```

Além desta verificação, o recurso `kmod` cuida da maioria dos itens que esses ports precisam levar em

consideração.

6.29. Bibliotecas Go

Os ports não devem empacotar ou instalar bibliotecas Go ou código-fonte. Os ports Go devem baixar as dependências na hora da compilação e devem instalar apenas programas que os usuários precisam, e não o que os desenvolvedores Go precisam.

Ports devem (por ordem de preferência):

- Usar as dependências fornecidas no código fonte do pacote.
- Baixar as versões das dependências especificadas pelo upstream (no caso do go.mod, vendor.json ou similar).
- Como um último recurso (as dependências não estão incluídas e nem as versões foram especificadas exatamente) busque as versões das dependências disponíveis no momento do desenvolvimento/release.

6.30. Bibliotecas Haskell

Assim como na linguagem Go, Ports não devem empacotar ou instalar as bibliotecas Haskell. Os ports Haskell devem vincular estaticamente a suas dependências e buscar todos os arquivos de distribuição no estágio fetch.

6.31. Arquivos Shell Completion

Muitos shells modernos (incluindo bash, tcsh e zsh) suportam parâmetro e/ou opção de tab-completion. Esse suporte geralmente vem de arquivos completion, os quais contêm as definições de como as tabs completion funcionarão para um determinado comando. As vezes ports vem com seus arquivos completion, ou os mantenedores de ports podem ter criado um eles mesmos.

Quando disponível, os arquivos de completion devem sempre ser instalados. Não é necessário fazer uma opção para eles. Apesar que se uma opção for usada, sempre habilite-a em `OPTIONS_DEFAULT`.

Tabela 50. Caminhos dos arquivos shell completion

`bash`	${PREFIX}/etc/bash_completion.d
`zsh`	${PREFIX}/shared/zsh/site-functions

Não registre nenhuma dependência nos próprios shells.

Capítulo 7. Flavors

7.1. Uma Introdução aos Flavors

Os flavors são uma maneira de ter várias variações de um port. O port é construído várias vezes, com variações.

Por exemplo, um port pode ter uma versão normal com muitos recursos e algumas dependências, e uma versão leve "lite" com apenas recursos básicos e dependências mínimas.

Outro exemplo poderia ser, um port pode ter um flavor GTK e um QT, dependendo de qual kit de ferramentas ele usa.

7.2. Usando FLAVORS

Para declarar um port com vários flavors, adicione `FLAVORS` no seu Makefile. O primeiro flavor em `FLAVORS` é o flavor padrão.

Isso pode ajudar a simplificar a lógica do Makefile para também definir um `FLAVOR` como:

```
FLAVOR?=    ${FLAVORS:[1]}
```

Para distinguir os flavors das opções, que são sempre letras maiúsculas, os nomes dos flavors podem conter *apenas* letras minúsculas, números e underline _.

Exemplo 95. Uso Básico de Flavors

Se um port tiver um port slave "lite", o port slave pode ser removido, e o port pode ser convertido em flavors com:

```
FLAVORS=    default lite
lite_PKGNAMESUFFIX= -lite
[...]
.if ${FLAVOR:U} != lite
[enable non lite features]
.endif
```

O primeiro flavor é o padrão, e é chamado aqui de `default`. Não é uma obrigação e, se possível, use um nome de flavor mais específico, como em Outro Uso Básico de Flavors.

Exemplo 96. Outro Uso Básico de Flavors

Se um port tiver um port slave `-nox11`, o port slave pode ser removido, e o port pode ser convertido em flavors com:

```
FLAVORS=    x11 nox11
FLAVOR?=    ${FLAVORS:[1]}
nox11_PKGNAMESUFFIX=    -nox11
[...]
.if ${FLAVOR} == x11
[enable x11 features]
.endif
```

Exemplo 97. Uso Mais Complexo de Flavors

Aqui está um excerto ligeiramente editado do que está presente em devel/libpeas, um port que usa os flavors Python. Com as versões padrões do Python 2 e 3 sendo 2.7 e 3.6, ele irá automaticamente mudar para `FLAVORS=py27 py36`

```
USES=       gnome python
USE_PYTHON= flavors ①

.if ${FLAVOR:Upy27:Mpy2*} ②
USE_GNOME=  pygobject3 ③

CONFIGURE_ARGS+=    --enable-python2 --disable-python3

BUILD_WRKSRC=   ${WRKSRC}/loaders/python ④
INSTALL_WRKSRC= ${WRKSRC}/loaders/python ⑤
.else # py3*
USE_GNOME+= py3gobject3 ⑥

CONFIGURE_ARGS+=    --disable-python2 --enable-python3 \
            ac_cv_path_PYTHON3_CONFIG=${LOCALBASE}/bin/python${PYTHON_VER}-config
⑦

BUILD_WRKSRC=   ${WRKSRC}/loaders/python3 ⑧
INSTALL_WRKSRC= ${WRKSRC}/loaders/python3 ⑨
.endif

py34_PLIST= ${.CURDIR}/pkg-plist-py3 ⑩
py35_PLIST= ${.CURDIR}/pkg-plist-py3 ⑪
py36_PLIST= ${.CURDIR}/pkg-plist-py3 ⑫
```

① Este port não usa o `USE_PYTHON=distutils` mas precisa do flavor Python de qualquer maneira.

② Para proteger contra o `FLAVOR` estar vazio, o que causaria um erro no make(1), use

`${FLAVOR:U}` em comparações de strings em vez de `${FLAVOR}`.

③ As ligações gobject3 doGnome Python têm dois nomes diferentes, um para Python2, pygobject3 e um para Python3, py3gobject3.

④ O script `configure` tem que ser executado em ${WRKSRC}, mas estamos interessados apenas em compilar e instalar as partes Python 2 ou Python 3 do software, então configure os diretórios base de compilação e instalação apropriadamente.

⑤ Sugestão sobre o nome correto do caminho do script de configuração do Python 3.

⑥ A lista de empacotamento é diferente quando compilada com Python 3. Como existem três possíveis versões do Python3 , defina `PLIST` para todos os três usando o helper.

7.2.1. Flavors Helpers

Para tornar o Makefile mais fácil de ser escrito, existem alguns flavors helpers.

Esta lista de helpers definirá sua variável:

- `flavor_PKGNAMEPREFIX`
- `flavor_PKGNAMESUFFIX`
- `flavor_PLIST`
- `flavor_DESCR`

Esta lista de helpers será anexada à sua variável:

- `flavor_CONFLICTS`
- `flavor_CONFLICTS_BUILD`
- `flavor_CONFLICTS_INSTALL`
- `flavor_PKG_DEPENDS`
- `flavor_EXTRACT_DEPENDS`
- `flavor_PATCH_DEPENDS`
- `flavor_FETCH_DEPENDS`
- `flavor_BUILD_DEPENDS`
- `flavor_LIB_DEPENDS`
- `flavor_RUN_DEPENDS`
- `flavor_TEST_DEPENDS`

Exemplo 98. Flavor Específico `PKGNAME`

Como todos os pacotes devem ter um nome de pacote diferente, os flavors devem mudar os seus, usando `flavor_PKGNAMEPREFIX` e o `flavor_PKGNAMESUFFIX` torna isso fácil:

```
FLAVORS=    normal lite
```

```
lite_PKGNAMESUFFIX= -lite
```

7.3. USES=php e Flavors

Ao usar o USES=php com um destes argumentos, phpize, ext, zend ou pecl, o port terá automaticamente o FLAVORS preenchido com a versão PHP que ele suporta.

Todos os exemplos assumem que as versões PHP suportadas atualmente são 5.6, 7.0, 7.1 e 7.2.

Exemplo 99. Extensão Simples USES=php

Isso irá gerar o pacote para todas as versões suportadas:

```
PORTNAME=   some-ext
PORTVERSION=    0.0.1
PKGNAMEPREFIX=  ${PHP_PKGNAMEPREFIX}

USES=       php:ext
```

Isto irá gerar pacotes para todas as versões suportadas, menos a 7.2:

```
PORTNAME=   some-ext
PORTVERSION=    0.0.1
PKGNAMEPREFIX=  ${PHP_PKGNAMEPREFIX}

USES=       php:ext
IGNORE_WITH_PHP=    72
```

7.3.1. Flavors PHP com Aplicações PHP

Aplicações PHP também podem ter flavors.

Isso permite gerar pacotes para todas as versões do PHP, para que os usuários possam usá-los com qualquer versão que precisarem em seus servidores.

Aplicações PHP que são acrescidas de flavors *devem* acrescentar PHP_PKGNAMESUFFIX aos nomes dos pacotes.

Exemplo 100. Adicionando Flavors em uma Aplicação PHP

Incluir o suporte de Flavors em uma aplicação PHP é simples:

```
PKGNAMESUFFIX=  ${PHP_PKGNAMESUFFIX}
```

```
USES=   php:flavors
```

Ao adicionar uma dependência em um port com flavors PHP, use `@${PHP_FLAVOR}`. *Nunca* use `FLAVOR` diretamente.

7.4. `USES=python` e Flavors

Ao usar `USES=python` e `USE_PYTHON=distutils`, o port irá automaticamente preencher `FLAVORS` com a versão Python que suporta.

Exemplo 101. Simples `USES=python`

Supondo que as versões suportadas do Python são 2.7, 3.4, 3.5 e 3.6, e a versão padrão do Python 2 e 3 são 2.7 e 3.6, um port com:

```
USES=   python
USE_PYTHON= distutils
```

Receberá esses flavors: `py27` e `py36`.

```
USES=   python
USE_PYTHON= distutils allflavors
```

Receberá esses flavors: `py27`, `py34`, `py35` e `py36`.

Exemplo 102. `USES=python` *com Requisitos de Versão*

Supondo que as versões suportadas do Python são 2.7, 3.4, 3.5 e 3.6, e a versão padrão do Python 2 e 3 são 2.7 e 3.6, um port com:

```
USES=   python:-3.5
USE_PYTHON= distutils
```

Vai ter esse flavor: `py27`.

```
USES=   python:-3.5
USE_PYTHON= distutils allflavors
```

Receberá esses flavors: `py27`, `py34` e `py35`.

```
USES=   python:3.4+
```

```
USE_PYTHON= distutils
```

Vai ter esse flavor: `py36`.

```
USES=   python:3.4+
USE_PYTHON= distutils allflavors
```

Receberá esses flavors: `py34`, `py35` e `py36`.

A variável `PY_FLAVOR` é disponibilizada para depender da versão correta dos módulos Python. Todas as dependências em ports Python com flavors devem usar `PY_FLAVOR`, e não `FLAVOR` diretamente.

Exemplo 103. Para um port que não usa `distutils`

Se a versão padrão do Python3 é 3.6, o seguinte irá definir a variável `PY_FLAVOR` para `py36`:

```
RUN_DEPENDS=    ${PYTHON_PKGNAMEPREFIX}mutagen>0:audio/py-mutagen@${PY_FLAVOR}

USES=   python:3.5+
```

7.5. `USES=lua` e Flavors

Ao usar `lua:module` ou `lua:flavors`, o port terá automaticamente `FLAVORS` preenchidos com as versões Lua que suporta. No entanto, não se espera que aplicativos comuns (em vez de módulos Lua) usem este recurso; a maioria das aplicações que incorporam ou usam Lua simplesmente devem usar `USES=lua`.

`LUA_FLAVOR` está disponível (e deve ser usado) para depender da versão correta das dependências, independentemente do port usar os parâmetros `flavors` ou `module`.

Veja Usando Lua para maiores informações.

Capítulo 8. Prácticas Avançadas de pkg-plist

8.1. Alterando o pkg-plist Baseado em Variáveis Make

Alguns ports, particularmente os `p5-` ports, precisam mudar seus pkg-plist dependendo de quais opções eles são configurados com (ou versão de `perl`, no caso de `p5-` ports). Para tornar isso fácil, todas as instâncias pkg-plist de `%%OSREL%%`, `%%PERL_VER%%` e `%%PERL_VERSION%%` serão substituídas apropriadamente. O valor de `%%OSREL%%` é a revisão numérica do sistema operacional (por exemplo,`4.9`). `%%PERL_VERSION%%` e `%%PERL_VER%%` é o número completo da versão `perl` (por exemplo,`5.8.9`). Muitos outros `%%VARS%%` relacionados aos arquivos de documentação do port são descritos na seção relevante.

Para fazer outras substituições, defina `PLIST_SUB` com uma lista de pares `VAR=VALOR` e as instâncias de `%%VAR%%` serão substituídas por *VALOR* no pkg-plist.

Por exemplo, se um port instalar muitos arquivos em um subdiretório específico da versão, use um placeholder para a versão de modo que o pkg-plist não precise ser gerado novamente toda vez que o port é atualizado. Por exemplo:

```
OCTAVE_VERSION= ${PORTREVISION}
PLIST_SUB=  OCTAVE_VERSION=${OCTAVE_VERSION}
```

no Makefile e use `%%OCTAVE_VERSION%%` onde quer que a versão apareça em pkg-plist. Quando o port é atualizado, não será necessário editar dezenas (ou em alguns casos, centenas) de linhas no pkg-plist.

Se os arquivos são instalados condicionalmente pelas opções definidas no port, a maneira usual de lidar com isso é prefixando as linhas pkg-plist com `%%OPT%%` para linhas necessárias quando a opção está ativada ou `%%NO_OPT%%` quando a opção está desativada e adicionando `OPTIONS_SUB=yes` ao Makefile. Veja `OPTIONS_SUB` para mais informações.

Por exemplo, se houver arquivos que são instalados apenas quando a opção `X11` está ativada, e o Makefile tem:

```
OPTIONS_DEFINE= X11
OPTIONS_SUB=    yes
```

No pkg-plist, insira `%%X11%%` no início das linhas que serão instaladas apenas quando a opção estiver habilitada, assim:

```
%%X11%%bin/foo-gui
```

Esta substituição será feita entre os targets `pre-install` e `do-install`, lendo a partir do PLIST e escrevendo em TMPPLIST (padrão:WRKDIR/.PLIST.mktmp). Então, se o port gera o PLIST na hora da compilação, faça isso em ou antes do `pre-install`. Além disso, se o port precisar editar o arquivo

resultante, faça-o em `post-install` em um arquivo chamado TMPPLIST.

Outra maneira de modificar a lista de empacotamento de um port é baseada na configuração das variáveis `PLIST_FILES` e `PLIST_DIRS`. O valor de cada variável é considerado como uma lista de nomes de caminho para gravar no TMPPLIST junto com conteúdo do PLIST. Enquanto os nomes listados no `PLIST_FILES` e `PLIST_DIRS` estão sujeitos a substituição do `%%VAR%%` conforme descrito acima, é melhor usar o `${VAR}` diretamente. Exceto por isso, os nomes contidos no `PLIST_FILES` aparecerão inalterados na lista final de packing, enquanto o `@dir` será anexado aos nomes do `PLIST_DIRS`. Para fazer efeito, o `PLIST_FILES` e o `PLIST_DIRS` devem ser definidos antes que o TMPPLIST seja escrito, isto é, no `pre-install` ou antes.

De vez em quando, usar `OPTIONS_SUB` não é o suficiente. Nesses casos, adicionar uma `TAG` para `PLIST_SUB` dentro do Makefile com um valor especial `@comment`, faz as ferramentas de pacote ignorar a linha. Por exemplo, se alguns arquivos são instalados apenas quando a opção `X11` está habilitada e a arquitetura é `i386`:

```
.include <bsd.port.pre.mk>

.if ${PORT_OPTIONS:MX11} && ${ARCH} == "i386"
PLIST_SUB+= X11I386=""
.else
PLIST_SUB+= X11I386="@comment "
.endif
```

8.2. Diretórios Vazios

8.2.1. Limpando Diretórios Vazios

Ao ser desinstalado, um port deve remover os diretórios vazios que ele criou. A maioria desses diretórios são removidos automaticamente pelo pkg(8), mas para os diretórios criados fora do ${PREFIX}, ou diretórios vazios, mais alguns passos precisam ser feitos. Isso geralmente é realizando adicionando entradas `@dir` para esses diretórios.Os subdiretórios devem ser excluídos antes de excluir os diretórios pai.

```
[...]
@dir /var/games/oneko/saved-games
@dir /var/games/oneko
```

8.2.2. Criando Diretórios Vazios

Os diretórios vazios criados durante a instalação do port precisam de atenção especial. Eles devem estar presentes quando o pacote é criado. Se eles não forem criados pelo código do port, crie-os no Makefile:

```
post-install:
```

```
    ${MKDIR} ${STAGEDIR}${PREFIX}/some/directory
```

Adicione o diretório ao pkg-plist como qualquer outro. Por exemplo:

```
@dir some/directory
```

8.3. Arquivos de Configuração

Se o port instalar arquivos de configuração em PREFIX/etc (ou em outro lugar) *não* liste-os em pkg-plist. Isso fará com que `pkg delete` remova os arquivos que foram cuidadosamente editados pelo usuário, e uma reinstalação irá eliminá-los.

Em vez disso, instale arquivos de exemplo com uma extensão filename.sample. A macro `@sample` automatiza isso, consulte `@sample` *file*[*file*] para entender o que ela faz exatamente. Para cada arquivo de exemplo, adicione uma entrada no pkg-plist:

```
@sample etc/orbit.conf.sample
```

Se houver uma boa razão para não instalar um arquivo de configuração por padrão, liste apenas o nome do arquivo de exemplo em pkg-plist, sem o `@sample` seguido por um espaço e adicione uma mensagem ressaltando que o usuário deve copiar e editar o arquivo antes que o software seja executado.

Quando um port instala sua configuração em um subdiretório de ${PREFIX}/etc, usar `ETCDIR`, cujo padrão é `${PREFIX}/etc/${PORTNAME}`, pode ser substituído nos Makefile dos ports se houver uma convenção para o port usar algum outro diretório. A macro `%%ETCDIR%%` será usado em seu lugar em pkg-plist.

Os arquivos de configuração de exemplo devem sempre ter o sufixo .sample. Se, por algum motivo histórico, o uso do sufixo padrão não for possível ou se os arquivos de exemplo vierem de algum outro diretório, use esta construção:

```
@sample etc/orbit.conf-dist etc/orbit.conf
```

ou

```
@sample %%EXAMPLESDIR%%/orbit.conf etc/orbit.conf
```

O formato é `@sample sample-file actual-config-file`.

8.4. Lista de Pacotes Estática versus Dinâmica

Uma *lista de pacotes estáticos* é uma lista de pacotes que está disponível na Coleção de Ports ou como pkg-plist (com ou sem substituição de variável), ou embutido no Makefile através do `PLIST_FILES` e do `PLIST_DIRS`. Mesmo se o conteúdo for gerado automaticamente por uma ferramenta ou um taget no Makefile *antes* da inclusão na Coleção de Ports por um committer (por exemplo, usando `make makeplist`), isso ainda é considerado uma lista estática, já que é possível examiná-la sem ter que baixar ou compilar o distfile.

Uma *lista de pacotes dinâmicos* é uma lista de pacotes que é gerada no momento em que o port é compilado com base nos arquivos e diretórios que estão instalados. Não é possível examiná-lo antes que o código-fonte do aplicativo portado seja baixado e compilado, ou após executar um `make clean`.

Embora o uso de listas de pacotes dinâmicos não seja proibido, os mantenedores devem usar listas de pacotes estáticos sempre que possível, já que isso permite aos usuários utilizar grep(1) nos de ports disponíveis para descobrir, por exemplo, qual port instala um determinado arquivo. Listas dinâmicas devem ser usadas principalmente para ports complexos onde a lista de pacotes muda drasticamente com base nos recursos opcionais do port (e assim manter uma lista de pacotes estática é impraticável), ou ports que alteram a lista de pacotes com base na versão do software dependente usado. Por exemplo, ports que geram documentos com Javadoc.

8.5. Criação Automatizada da Lista de Pacotes

Primeiro, verifique se o port está quase completo, faltando apenas o pkg-plist. Executar o comando `make makeplist` irá mostrar um exemplo para o pkg-plist. A saída do `makeplist` deve ser checada duas vezes quanto à correção, pois ela tenta adivinhar automaticamente algumas coisas e pode errar.

Os arquivos de configuração do usuário devem ser instalados como filename.sample, como é descrito em Arquivos de Configuração. O info/dir não deve ser listado e entradas apropriadas install-info devem ser adicionadas conforme a seção arquivos de informação. Quaisquer bibliotecas instaladas pelo port devem ser listadas conforme especificado na seção bibliotecas compartilhadas.

8.5.1. Expansão do `PLIST_SUB` com Expressões Regulares

As strings a serem substituídas às vezes precisam ser muito específicas para evitar substituições indesejadas. Esse é um problema comum com valores mais curtos.

Para resolver este problema, para cada `PLACEHOLDER=value`, um `PLACEHOLDER_regex =regex` pode ser definido, com o *regex* do `value` correspondendo mais precisamente.

Exemplo 104. Usando PLIST_SUB com Expressões Regulares

Os ports Perl podem instalar arquivos dependentes da arquitetura em uma árvore específica. No FreeBSD para facilitar a portabilidade, esta árvore é chamada de `mach`. Por exemplo, um port que instala um arquivo cujo caminho contém `mach` poderia ter essa parte da sequência do caminho substituída pelos valores incorretos. Considere este Makefile:

```
PORTNAME=   Machine-Build
DISTVERSION=    1
CATEGORIES= devel perl5
MASTER_SITES=   CPAN
PKGNAMEPREFIX=  p5-

MAINTAINER= perl@FreeBSD.org
COMMENT=    Building machine

USES=       perl5
USE_PERL5=  configure

PLIST_SUB=  PERL_ARCH=mach
```

Os arquivos instalados pelo port são:

```
/usr/local/bin/machine-build
/usr/local/lib/perl5/site_perl/man/man1/machine-build.1.gz
/usr/local/lib/perl5/site_perl/man/man3/Machine::Build.3.gz
/usr/local/lib/perl5/site_perl/Machine/Build.pm
/usr/local/lib/perl5/site_perl/mach/5.20/Machine/Build/Build.so
```

Executar o `make makeplist` gera incorretamente:

```
bin/%%PERL_ARCH%%ine-build
%%PERL5_MAN1%%/%%PERL_ARCH%%ine-build.1.gz
%%PERL5_MAN3%%/Machine::Build.3.gz
%%SITE_PERL%%/Machine/Build.pm
%%SITE_PERL%%/%%PERL_ARCH%%/%%PERL_VER%%/Machine/Build/Build.so
```

Altere a linha `PLIST_SUB` do Makefile para:

```
PLIST_SUB=  PERL_ARCH=mach \
        PERL_ARCH_regex=\bmach\b
```

Agora o `make makeplist` gera corretamente:

```
bin/machine-build
%%PERL5_MAN1%%/machine-build.1.gz
%%PERL5_MAN3%%/Machine::Build.3.gz
%%SITE_PERL%%/Machine/Build.pm
%%SITE_PERL%%/%%PERL_ARCH%%/%%PERL_VER%%/Machine/Build/Build.so
```

8.6. Expandindo a Lista de Pacotes com Keywords

Todas as keywords também podem ter argumentos opcionais entre parênteses. Os argumentos são owner, group e mode. Esse argumento é usado no arquivo ou diretório referenciado. Para alterar o dono, o grupo e o modo de um arquivo de configuração, use:

```
@sample(games,games,640) etc/config.sample
```

Os argumentos são opcionais. Se apenas o grupo e o modo precisarem ser alterados, use:

```
@sample(,games,660) etc/config.sample
```

Se uma keyword for utilizada em uma entrada de opção, ela precisa ser adicionada após o assistente:

```
%%FOO%%@sample etc/orbit.conf.sample
```

Isso é porque os assistentes plist das opções são utilizados para comentar as linhas, e por isso eles precisam ser inseridos no início. Veja `OPTIONS_SUB` para maiores informações.

8.6.1. `@desktop-file-utils`

Irá executar o `update-desktop-database -q` após a instalação e desinstalação. *Nunca* use diretamente, adicione `USES=desktop-file-utils` ao Makefile.

8.6.2. `@fc` *directory*

Adiciona uma entrada `@dir` para o diretório passado como um argumento, e executa `fc-cache -fs` nesse diretório após a instalação e desinstalação.

8.6.3. `@fcfontsdir` *directory*

Adiciona uma entrada `@dir` para o diretório passado como um argumento, e executa `fc-cache -fs`, `mkfontscale` e `mkfontdir` nesse diretório após a instalação e desinstalação. Além disso, na desinstalação, ele remove os arquivos de cache fonts.scale e fonts.dir, se estiverem vazios. Esta keyword é equivalente a adicionar o diretório `@fc` e o diretório `@fontsdir`.

8.6.4. `@fontsdir` *directory*

Adiciona um entrada `@dir` para o diretório passado como um argumento, e executa `mkfontscale` e `mkfontdir` nesse diretório após a instalação e desinstalação. Além disso, na desinstalação, ele remove os arquivos de cache fonts.scale e fonts.dir, se estiverem vazios.

8.6.5. `@glib-schemas`

Executa `glib-compile-schemas` na instalação e desinstalação.

8.6.6. `@info` *file*

Adiciona o arquivo passado como argumento ao plist e atualiza o índice do documento info na instalação e desinstalação. Além disso, ele remove o índice se estiver vazio na desinstalação. Isso nunca deve ser usado manualmente, mas sempre `INFO`. Veja Arquivos de Informação para maiores informações.

8.6.7. `@kld` *directory*

Executa o `kldxref` no diretório na instalação e desinstalação. Além disso, na desinstalação, ele removerá o diretório se estiver vazio.

8.6.8. `@rmtry` *file*

O arquivo será removido na desinstalação, e não dará um erro se o arquivo não estiver lá.

8.6.9. `@sample` *file*[*file*]

Isso é usado para lidar com a instalação de arquivos de configuração, através de arquivos de exemplo empacotados com o pacote. O arquivo "atual", não-amostra, ou é o segundo nome do arquivo, se presente, ou o primeiro nome de arquivo sem a extensão .sample.

Isso faz três coisas. Primeiro, adiciona o primeiro arquivo passado como argumento, o arquivo de exemplo, ao plist. Então, na instalação, se o arquivo real não for encontrado, copia o arquivo de exemplo para o arquivo real. E, finalmente, na desinstalação, remove o arquivo atual se ele não tiver sido modificado. Veja Arquivos de Configuração para maiores informações.

8.6.10. `@shared-mime-info` *directory*

Executa `update-mime-database` no diretório na instalação e desinstalação.

8.6.11. `@shell` *file*

Adiciona o arquivo passado como argumento ao plist.

Na instalação, adiciona o caminho completo do *file* em /etc/shells, certificando-se que não é adicionado duas vezes. Na desinstalação, remove-o de /etc/shells.

8.6.12. `@terminfo`

Não use sozinho. Se o port for instalar arquivos *.terminfo, adicione USES=terminfo no seu Makefile.

Na instalação e desinstalação, se o `tic` estiver presente, atualize o ${PREFIX}/shared/misc/terminfo.db a partir dos arquivos *.terminfo disponíveis em

${PREFIX}/shared/misc.

8.6.13. Keywords Básicas

Existem algumas keywords que são codificadas e documentadas em pkg-create(8). Por uma questão de completude, elas também estão documentadas aqui.

`@` [*file*]

A keyword vazia é um espaço reservado para ser usado quando o proprietário, grupo ou modo do arquivo precisam ser alterados. Por exemplo, para definir o grupo de um arquivo como `games` e adicionar o bit setgid, adicione:

```
@(,games,2755) sbin/daemon
```

`@preexec` *command*, `@postexec` *command*, `@preunexec` *command*, `@postunexec` *command*

Executa o *command* como parte do processo de instalação ou desinstalação.

`@preexec` *command*
: Executar o *command* como parte dos scripts pre-install.

`@postexec` *command*
: Executar o *command* como parte dos scripts post-install.

`@preunexec` *command*
: Executar o *command* como parte dos scripts pre-deinstall.

`@postunexec` *command*
: Executar o *command* como parte dos scripts post-deinstall.

E se o *command* contém qualquer uma dessas sequências em algum lugar, elas são expandidas em linha. Para estes exemplos, assuma que `@cwd` está configurado para /usr/local e o último arquivo extraído foi bin/emacs.

`%F`
: Expandir para o último nome de arquivo extraído (conforme especificado). No caso do exemplo bin/emacs.

`%D`
: Expandir para o prefixo do diretório atual, como definido no `@cwd`. No caso do exemplo /usr/local.

`%B`
: Expandir para o nome de base do nome completo do arquivo, ou seja, o prefixo do diretório atual mais o último arquivo, menos o nome do arquivo final. No exemplo, isso seria /usr/local/bin.

`%f`

Expandir para a parte do nome do arquivo do nome totalmente qualificado, ou o inverso de `%B`. No caso do exemplo, emacs.

> Essas keywords estão aqui para ajudá-lo a configurar o pacote para que ele esteja tão pronto quanto possível. Elas *não devem* ser abusadas para iniciar serviços, interromper serviços ou executar quaisquer outros comandos que modificarão o sistema em execução.

`@mode` ***mode***

Define a permissão padrão para todos os arquivos extraídos posteriormente para *mode*. O formato é o mesmo usado por chmod(1). Use sem um argumento para voltar às permissões padrão (modo do arquivo enquanto estava sendo empacotado).

> Este deve ser um modo numérico, como `644`, `4755` ou `600`. Não pode ser um modo relativo como `u+s`.

`@owner` ***user***

Define a propriedade padrão para todos os arquivos subsequentes para *user*. Use sem um argumento para voltar à propriedade padrão (`root`).

`@group` ***group***

Define a propriedade de grupo padrão para todos os arquivos subsequentes para *group*. Use sem um argumento para retornar à propriedade do grupo padrão (`wheel`).

`@comment` ***string***

Esta linha é ignorada no momento de empacotar.

`@dir` ***directory***

Declara o nome do diretório. Por padrão, os diretórios criados sob `PREFIX` por uma instalação de pacote são automaticamente removidos. Use isto quando um diretório vazio sob `PREFIX` precisa ser criado ou quando o diretório precisa ter proprietário, grupo ou modo não padrão. Diretórios fora de `PREFIX` precisam ser registrados. Por exemplo, /var/db/${PORTNAME} precisa ter uma entrada `@dir` enquanto ${PREFIX}/shared/${PORTNAME} não, se contiver arquivos ou usar o proprietário, grupo e modo padrão.

`@exec` ***command***, `@unexec` ***command*** **(Descontinuado)**

Executa o *command* como parte do processo de instalação ou desinstalação. Por favor, use `@preexec` *command*, `@postexec` *command*, `@preunexec` *command*, `@postunexec` *command* no lugar.

`@dirrm` ***directory*** **(Descontinuado)**

Declara o nome do diretório a ser excluído na desinstalação. Por padrão, os diretórios criados sob

`PREFIX` por uma instalação de pacote são excluídos quando o pacote é desinstalado.

`@dirrmtry` *directory* (Descontinuado)

Declara o nome do diretório a ser removido, como para a keyword `@dirrm`, mas não emite um aviso se o diretório não puder ser removido.

8.6.14. Criando Novas Keywords

Os arquivos da lista de pacotes podem ser estendidos por keywords definidas no diretório ${PORTSDIR}/Keywords. As configurações de cada keyword são armazenadas em um arquivo UCL chamado keyword.ucl. O arquivo deve conter pelo menos uma destas seções:

- `attributes`
- `action`
- `pre-install`
- `post-install`
- `pre-deinstall`
- `post-deinstall`
- `pre-upgrade`
- `post-upgrade`

`attributes`

Altera o dono, grupo ou modo usado pela keyword. Contém uma matriz associativa em que as chaves possíveis são `owner`, `group` e `mode`. Os valores são, respectivamente, um nome de usuário, um nome de grupo e um modo de arquivo. Por exemplo:

```
attributes: { owner: "games", group: "games", mode: 0555 }
```

`action`

Define o que acontece com o parâmetro da keyword. Contém uma matriz onde os valores possíveis são:

`setprefix`
: Define o prefixo para as próximas entradas do plist.

`dir`
: Registra um diretório para ser criado na instalação e removido na desinstalação.

`dirrm`
: Registra um diretório a ser excluído na desinstalação. Descontinuado.

`dirrmtry`
: Registra um diretório para tentar deletar na desinstalação. Descontinuado.

`file`

Registra um arquivo.

`setmode`

Define o modo para as próximas entradas do plist.

`setowner`

Define o dono para as próximas entradas do plist.

`setgroup`

Define o grupo para as próximas entradas do plist.

`comment`

Não faz nada, é o equivalente a não entrar em uma seção `action`.

`ignore_next`

Ignora a próxima entrada no plist.

`arguments`

Se definido para `true`, adiciona manipulação de argumentos, dividindo toda a linha, `%@`, em argumentos numerados,`%1`, `%2`, e assim por diante. Por exemplo, para esta linha:

```
@foo some.content other.content
```

`%1` e `%2` irão conter:

```
some.content
other.content
```

Também afeta como a entrada `action` funciona. Quando há mais de um argumento, o número do argumento deve ser especificado. Por exemplo:

```
actions: [file(1)]
```

`pre-install`, `post-install`, `pre-deinstall`, `post-deinstall`, `pre-upgrade`, `post-upgrade`

Essas keywords contêm um script sh(1) a ser executado antes ou depois da instalação, desinstalação, ou atualização do pacote. Além do habitual placeholder `@exec %foo` descrito em `@preexec` *command*, `@postexec` *command*, `@preunexec` *command*, `@postunexec` *command*, há um novo `%@`, que representa o argumento da keyword.

Exemplos de Keywords Customizadas

Exemplo 105. Exemplo de uma Keyword `@dirrmtryecho`

Esta keyword faz duas coisas, adiciona uma linha `@dirrmtry directory` na lista de empacotamento, e escreve no log quando o diretório é removido ao desinstalar o pacote.

```
actions: [dirrmtry]
post-deinstall: <<EOD
  echo "Directory %D/%@ removed."
EOD
```

Exemplo 106. Exemplo na vida real, como o `@sample` é implementado

Esta keyword faz três coisas. Ela adiciona o primeiro *filename* passado como um argumento para `@sample` na lista de empacotamento, ele adiciona instruções ao script de `post-install` para copiar o exemplo para o arquivo de configuração real, se ele ainda não existir, e adiciona instruções ao script `post-deinstall` para remover o arquivo de configuração se ele não tiver sido modificado.

```
actions: [file(1)]
arguments: true
post-install: <<EOD
  case "%1" in
  /*) sample_file="%1" ;;
  *) sample_file="%D/%1" ;;
  esac
  target_file="${sample_file%.sample}"
  set -- %@
  if [ $# -eq 2 ]; then
      target_file=${2}
  fi
  case "${target_file}" in
  /*) target_file="${target_file}" ;;
  *) target_file="%D/${target_file}" ;;
  esac
  if ! [ -f "${target_file}" ]; then
    /bin/cp -p "${sample_file}" "${target_file}" && \
      /bin/chmod u+w "${target_file}"
  fi
EOD
pre-deinstall: <<EOD
  case "%1" in
  /*) sample_file="%1" ;;
  *) sample_file="%D/%1" ;;
  esac
  target_file="${sample_file%.sample}"
  set -- %@
  if [ $# -eq 2 ]; then
      set -- %@
```

```
      target_file=${2}
  fi
  case "${target_file}" in
  /*) target_file="${target_file}" ;;
  *) target_file="%D/${target_file}" ;;
  esac
  if cmp -s "${target_file}" "${sample_file}"; then
    rm -f "${target_file}"
  else
    echo "You may need to manually remove ${target_file} if it is no longer
needed."
  fi
EOD
```

Capítulo 9. pkg-*

Existem alguns truques que ainda não foram mencionamos sobre os arquivos pkg-* que são úteis às vezes.

9.1. pkg-message

Para exibir uma mensagem quando o pacote é instalado, coloque a mensagem no pkg-message. Esse recurso é geralmente útil para exibir etapas adicionais de instalação a serem executadas após o `pkg install` ou `pkg upgrade`.

- pkg-message deve conter apenas informações *vitais* de setup e operação no FreeBSD, e isso é único para o port em questão.
- As informações de configuração devem ser mostradas apenas na instalação inicial. As instruções de atualização devem ser exibidas apenas ao atualizar a versão relevante.
- Não coloque as mensagens entre espaços em branco ou linhas de símbolos (como `----------`, , ou `==========`). Deixe a formatação com o pkg(8).
- Os committers têm aprovação implícita para restringir as mensagens existentes na hora da instalação ou em intervalos de atualização, usando as especificações do formato UCL.

pkg-message suporta dois formatos:

raw

Um arquivo de texto simples comum. Sua mensagem é exibida apenas na instalação.

UCL

Se o arquivo começar com "[" será considerado como um arquivo UCL. O formato UCL é descrito na página libucl no GitHub.

Não adicione uma entrada para o pkg-message ao pkg-plist.

9.1.1. UCL no pkg-message

O formato é o seguinte. Deve ser uma matriz de objetos. Os objetos em si podem ter essas palavras-chave:

`message`

A mensagem atual a ser exibida. Esta palavra-chave é obrigatória.

`type`

Quando a mensagem deve ser exibida.

`maximum_version`

Somente se `type` for `upgrade`. Exibe se estiver atualizando de uma versão inferior que a versão

especificada.

`minimum_version`

Somente se `type` for `upgrade`. Exibe se estiver atualizando de uma versão maior que a versão especificada.

As palavras-chave `maximum_version` e `minimum_version` podem ser combinadas.

A palavra-chave `type` pode ter três valores:

`install`

A mensagem só deve ser exibida quando o pacote é instalado.

`remove`

A mensagem só deve ser exibida quando o pacote é removido.

`upgrade`

a mensagem só deve ser exibida durante uma atualização do pacote.

Para preservar a compatibilidade com arquivos pkg-message não UCL, a primeira linha de um arquivo pkg-message UCL DEVE ter um simples “[”, e a última linha DEVE ter um simples “]”.

Exemplo 107. Strings Curtas UCL

A mensagem é delimitada por aspas duplas `"`, isto é utilizado em strings simples de linha única:

```
[
{ type: install
  message: "Simple message"
}
]
```

Exemplo 108. Strings de Múltiplas Linhas UCL

Strings de múltiplas linhas utiliza o padrão here de documento de notação. O delimitador de múltiplas linhas *deve* iniciar logo após os símbolos `<<` sem espaço em branco, e ele *deve* ser apenas em letras maiúsculas. Para finalizar uma sequência de múltiplas linhas, adicione o delimitador em uma linha única, sem nenhum espaço em branco. A mensagem de Strings Curtas UCL pode ser escrita como:

```
[
{ type: install
  message: <<EOM
Simple message
EOM
}
```

```
]
```

Exemplo 109. Exibir uma Mensagem na Instalação/Desinstalação

Quando uma mensagem precisa ser exibida apenas na instalação ou na desinstalação, defina o tipo:

```
[
{
  type: remove
  message: "package being removed."
}
{ type: install, message: "package being installed."}
]
```

Exemplo 110. Exibir uma Mensagem na Atualização

Quando um port é atualizado, a mensagem exibida pode ser ainda mais adaptada às necessidades do port.

```
[
{
  type: upgrade
  message: "Package is being upgraded."
}
{
  type: upgrade
  maximum_version: "1.0"
  message: "Upgrading from before 1.0 need to do this."
}
{
  type: upgrade
  minimum_version: "1.0"
  message: "Upgrading from after 1.0 should do that."
}
{
  type: upgrade
  maximum_version: "3.0"
  minimum_version: "1.0"
  message: "Upgrading from > 1.0 and < 3.0 remove that file."
}
]
```

Ao exibir uma mensagem na atualização, é importante limitar até quando ela será mostrada ao usuário. Na maioria das vezes, é usado o `maximum_version` para limitar seu uso a atualizações anteriores a uma certa versão, quando

algo específico precisa ser feito.

9.2. pkg-install

Se o port precisa executar comandos quando o pacote binário é instalado com o `pkg add` ou com o `pkg install`, use o pkg-install. Este script será automaticamente adicionado ao pacote. Será executado duas vezes pelo `pkg`, a primeira vez como `${SH} pkg-install ${PKGNAME} PRE-INSTALL` antes que o pacote seja instalado e uma segunda vez como `${SH} pkg-install ${PKGNAME} POST-INSTALL` depois dele ter sido instalado. O valor de `$2` pode ser testado para determinar em que modo o script está sendo executado. A variável de ambiente `PKG_PREFIX` será definida para o diretório de instalação do pacote.

> Este script está aqui para ajudá-lo a configurar o pacote para que ele esteja tão pronto quanto possível para ser usado. Ele *não deve* ser abusado para iniciar serviços, interromper serviços ou executar quaisquer outros comandos que modificarão o sistema em execução no momento.

9.3. pkg-deinstall

Este script é executado quando um pacote é removido.

Este script será executado duas vezes pelo `pkg delete`. A primeira vez como `${SH} pkg-deinstall ${PKGNAME} DEINSTALL` antes que o port seja desinstalado e a segunda vez como `${SH} pkg-deinstall ${PKGNAME} POST-DEINSTALL` após o port ter sido desinstalado. O valor de `$2` pode ser testado para determinar em que modo o script está sendo executado. A variável de ambiente `PKG_PREFIX` será definida para o diretório de instalação do pacote

> Este script está aqui para ajudá-lo a configurar o pacote para que ele esteja tão pronto quanto possível para ser usado. Ele *não deve* ser abusado para iniciar serviços, interromper serviços ou executar quaisquer outros comandos que modificarão o sistema em execução no momento.

9.4. Mudando os nomes dos pkg-*

Todos os nomes de pkg- são definidos usando variáveis que podem ser alteradas no Makefile se necessário. Isso é especialmente útil ao compartilhar os mesmos arquivos pkg- entre vários ports ou quando é necessário gravar em um desses arquivos. Veja escrevendo em lugares que não o `WRKDIR` para entender por que é uma má ideia escrever diretamente no diretório que contém os arquivos pkg-*.

Aqui está uma lista de nomes de variáveis e seus valores padrão. (O valor padrão do `PKGDIR` é `${MASTERDIR}`.)

Variável	Valor padrão
`DESCR`	`${PKGDIR}/pkg-descr`

Variável	Valor padrão
PLIST	${PKGDIR}/pkg-plist
PKGINSTALL	${PKGDIR}/pkg-install
PKGDEINSTALL	${PKGDIR}/pkg-deinstall
PKGMESSAGE	${PKGDIR}/pkg-message

9.5. Fazendo uso de SUB_FILES e SUB_LIST

O SUB_FILES e o SUB_LIST são úteis para valores dinâmicos em arquivos do port, como o PREFIX de instalação dentro do pkg-message.

A SUB_FILES especifica uma lista de arquivos a serem modificados automaticamente. Cada arquivo na lista SUB_FILES deve ter um arquivo.in correspondente presente no FILESDIR. Uma versão modificada será criada como ${WRKDIR}/arquivo. Os arquivos definidos como um valor de USE_RC_SUBR são automaticamente adicionados ao SUB_FILES. Para os arquivos pkg-message, pkg-install e pkg-deinstall, a variável Makefile correspondente é automaticamente definida para apontar para a versão processada.

A SUB_LIST é uma lista de pares VAR=VALUE. Para cada par, %%VAR%% será substituído por VALUE em cada arquivo listado em SUB_FILES. Vários pares comuns são definidos automaticamente: PREFIX, LOCALBASE, DATADIR, DOCSDIR, EXEMPLESDIR, WWWDIR e ETCDIR. Qualquer linha que comece com @Comment seguido por um espaço, será excluído dos arquivos resultantes após uma substituição de variável.

Este exemplo substitui %%ARCH%% com a arquitetura do sistema em um pkg-message:

```
SUB_FILES=  pkg-message
SUB_LIST=   ARCH=${ARCH}
```

Note que para este exemplo, o pkg-message.in deve existir no FILESDIR.

Exemplo de um bom pkg-message.in:

```
Now it is time to configure this package.
Copy %%PREFIX%%/shared/examples/putsy/%%ARCH%%.conf into your home directory
as .putsy.conf and edit it.
```

Capítulo 10. Testando o Port

10.1. Executando `make describe`

Várias das ferramentas de manutenção de ports do FreeBSD, tal como o portupgrade(1), conta com um banco de dados chamado /usr/ports/INDEX o qual mantém um registro de itens tais como as dependências do port. O INDEX é criado pelo ports/Makefile de nível superior através do comando `make index`, que desce em cada subdiretório dos ports e executa o comando `make describe` lá. Desta forma, se o `make describe` falhar em qualquer port, ninguém poderá gerar o INDEX e muitas pessoas rapidamente se tornarão infelizes.

É importante poder gerar este arquivo independentemente das opções presentes no make.conf então evite fazer coisas como usar declarações `.error` quando (por exemplo) uma dependência não estiver satisfeita. (Veja Evite o Uso do Construtor `.error`.)

E se o `make describe` produzir uma string em vez de uma mensagem de erro, provavelmente está tudo certo. Veja o bsd.port.mk para saber o significado da string gerada.

Note também que rodar uma versão recente do `portlint` (conforme especificado na próxima seção) executará o `make describe` automaticamente.

10.2. Portlint

Verifique o port com `portlint` antes de submete-lo ou de fazer o seu commit. O `portlint` alerta sobre muitos erros comuns, tanto funcionais quanto de estilo. Para um novo (ou um repocopiado) port , `portlint -A` é o uso mais completo; para um port existente, `portlint -C` é suficiente.

O `portlint` usa uma técnica heurística para tentar descobrir erros, pode produzir avisos falso-positivos. Além disso, ocasionalmente, algo que é sinalizado como um problema, pode não ter uma outra forma de ser realizado por limitações no framework dos ports. Em caso de dúvida, a melhor coisa a fazer é perguntar na Lista de discussão de ports do FreeBSD.

10.3. Ferramentas do Ports

O programa ports-mgmt/porttools faz parte da Coleção de Ports.

O `port` é o script front-end, que pode ajudar a simplificar o trabalho de teste. Sempre que um novo port ou uma atualização de um já existente precisar de teste, use `port test` para testar o port, incluindo a verificação `portlint`. Este comando também detecta e lista todos os arquivos que não estão listados no pkg-plist. Por exemplo:

```
# port test /usr/ports/net/csup
```

10.4. PREFIX e DESTDIR

PREFIX determina onde o port será instalado. O padrão é /usr/local, mas pode ser definido pelo usuário para um caminho personalizado como /opt. O port deve respeitar o valor dessa variável.

O DESTDIR, se definido pelo usuário, determina o ambiente alternativo completo, geralmente uma jail ou um sistema instalado montado em outro local que não seja o /. Um port será realmente instalado no DESTDIR/PREFIX, e registrado no banco de dados de pacotes em DESTDIR/var/db/pkg. Como o DESTDIR é tratado automaticamente pela infraestrutura de ports com o chroot(8). Não há necessidade de modificações ou qualquer cuidado extra para escrever ports compatíveis com o DESTDIR.

O valor de PREFIX será definido para LOCALBASE (o valor padrão é /usr/local). E se USE_LINUX_PREFIX estiver definido o PREFIX será LINUXBASE (o valor padrão é /compat/linux).

Evitar o uso do caminho /usr/local codificado no fonte tornam o port muito mais flexível e capaz de atender às necessidades de outros sites. Muitas vezes, isso pode ser feito substituindo as ocorrências de /usr/local nos vários Makefiles dos ports por ${PREFIX}. Essa variável é transmitida automaticamente para todos os estágios dos processos de compilação e instalação.

Verifique se o aplicativo não está instalando arquivos em /usr/local ao invés de PREFIX. Um teste rápido para esses caminhos codificados é:

```
% make clean; make package PREFIX=/var/tmp/`make -V PORTNAME`
```

Se alguma coisa for instalada fora do PREFIX, o processo de criação de pacotes irá reclamar que não pode encontrar os arquivos.

Além disso, vale a pena verificar o mesmo em relação ao suporte a diretórios stage (veja Staging):

```
% make stage && make check-plist && make stage-qa && make package
```

- O check-plist verifica arquivos ausentes do plist e arquivos no plist que não são instalados pelo port.
- O stage-qa verifica problemas comuns como shebang incorretas, links simbólicos apontando para fora do diretório de stage, arquivos setuid e bibliotecas não removidas...

Esses testes não encontrarão caminhos codificados dentro dos arquivos do port, nem verificarão se o LOCALBASE está sendo usado para se referir corretamente a arquivos de outros ports. O port instalado temporariamente em /var/tmp/make -V PORTNAME deve ser testado quanto à operação correta para garantir que não haja problemas com os caminhos.

O PREFIX não deve ser definido explicitamente em um Makefile do port. Usuários instalando o port podem ter definido a variável PREFIX para um local personalizado e o port deve respeitar essa configuração.

Referencie programas e arquivos de outros ports com as variáveis mencionadas acima, não com

nomes de caminho explícitos. Por exemplo, se o port exigir uma macro `PAGER` para ter o nome de caminho completo para o `less`, não use um caminho literal para /usr/local/bin/less. Em vez disso, use `${LOCALBASE}`:

```
-DPAGER=\"${LOCALBASE}/bin/less\"
```

O caminho com `LOCALBASE` é muito provável que ainda funcione se o administrador do sistema mudou toda a arvore /usr/local para algum outro lugar.

Todos esses testes são feitos automaticamente ao executar `poudriere testport` ou `poudriere bulk -t`. É altamente recomendável que cada contribuidor de ports instale e teste seus ports com ele. Veja Poudriere para maiores informações.

10.5. Poudriere

Para um contribuidor de ports, o Poudriere é uma das mais importantes e úteis ferramentas de teste e compilação. Suas principais características incluem:

- Compilação em massa de toda a árvore de ports, subconjuntos específicos da árvore de ports, ou um único port incluindo suas dependências
- Empacotamento automático do resultados de compilação
- Geração de arquivos de log de compilação por port
- Fornecer um repositório pkg(8) assinado
- Testar a compilação do port antes de enviar um patch para o rastreador de bugs do FreeBSD ou antes de fazer o commit para a árvore de ports
- Testar a compilação bem-sucedida de ports usando opções diferentes

Porque o Poudriere realiza a sua compilação em um ambiente de jail(8) limpo e usa características do zfs(8), ele tem várias vantagens sobre os testes tradicionais no sistema host:

- Ele não polui o ambiente do host: sem arquivos sobrando, sem remoções acidentais, sem alterações nos arquivos de configuração existentes.
- Ele verifica o pkg-plist para entradas ausentes ou supérfluas
- Committers de ports às vezes pedem um log do Poudriere juntamente com a apresentação de um patch para avaliar se o patch está pronto para integração na árvore de ports

Ele também é muito simples de configurar e usar, não tem dependências e será executado em qualquer versão suportada do FreeBSD. Esta seção mostra como instalar, configurar e executar o Poudriere como parte do fluxo de trabalho normal de um contribuidor de ports.

Os exemplos nesta seção mostram um layout de arquivo padrão, como padrão no FreeBSD. Substitua quaisquer alterações locais de acordo. A árvore de ports, representada por `${PORTSDIR}`, está localizada em /usr/ports. Ambos `${LOCALBASE}` e `${PREFIX}` são /usr/local por padrão.

10.5.1. Instalando o Poudriere

O Poudriere está disponível na árvore de ports em ports-mgmt/poudriere. Ele pode ser instalado usando o pkg(8) ou a partir do ports:

```
# pkg install poudriere
```

ou

```
# make -C /usr/ports/ports-mgmt/poudriere install clean
```

Há também uma versão de trabalho em andamento do Poudriere que acabará por se tornar o próximo release. Ele está disponível em ports-mgmt/poudriere-devel. Esta versão de desenvolvimento é usada para as compilações oficiais de pacotes do FreeBSD, então é bem testada. Muitas vezes tem novos recursos interessantes. Um committer de ports desejará usar a versão de desenvolvimento porque é o que é usado na produção e possui todos os novos recursos que farão com que tudo esteja exatamente correto. Um colaborador não precisará necessariamente deles, pois as correções mais importantes são sempre incorporadas na versão release. A principal razão para o uso da versão de desenvolvimento para compilar os pacotes oficiais é porque é mais rápido, de uma forma que encurtará uma compilação completa de 18 horas para 17 horas ao usar um servidor de 32 CPUs high-end com 128GB de RAM. Essas otimizações não terão muita importância ao compilar ports em uma máquina desktop.

10.5.2. Configurando o Poudriere

O port instala um arquivo de configuração padrão, o /usr/local/etc/poudriere.conf. Cada parâmetro é documentado no arquivo de configuração em poudriere(8). Aqui está um arquivo de configuração mínimo de exemplo:

```
ZPOOL=tank
ZROOTFS=/poudriere
BASEFS=/poudriere
DISTFILES_CACHE=/usr/ports/distfiles
RESOLV_CONF=/etc/resolv.conf
FREEBSD_HOST=ftp://ftp.freebsd.org
SVN_HOST=svn.FreeBSD.org
```

`ZPOOL`

O nome do pool de armazenamento do ZFS que o Poudriere deve usar. Deve ser listado na saída de `zpool status`.

`ZROOTFS`

A raiz dos sistemas de arquivos gerenciados do Poudriere. Esta entrada fará com que o Poudriere crie o sistema de arquivo zfs(8) sob `tank/poudriere`.

`BASEFS`

O ponto de montagem da raiz do sistema de arquivo Poudriere. Esta entrada fará com que o Poudriere monte o `tank/poudriere` no `/poudriere`.

`DISTFILES_CACHE`

Define onde os distfiles são armazenados. Neste exemplo, o Poudriere e o host compartilham o diretório de armazenamento dos distfiles. Isso evita o download de tarballs que já estão presentes no sistema. Por favor, crie este diretório se ele ainda não existir, para que o Poudriere possa encontrá-lo.

`RESOLV_CONF`

Utiliza o /etc/resolv.conf do host dentro do jails para a resolução de DNS. Isso é necessário para que as jails possam resolver as URLs dos distfiles durante o download. Não é necessário ao usar um proxy. Consulte o arquivo de configuração padrão para a configuração de proxy.

`FREEBSD_HOST`

O servidor FTP/HTTP a ser usado quando as jails são instaladas a partir de versões do FreeBSD e atualizadas com o freebsd-update(8). Escolha um servidor cuja localização esteja próxima, por exemplo, se a máquina estiver localizada na Austrália, use `ftp.au.freebsd.org`.

`SVN_HOST`

O servidor de onde as jails são instaladas e atualizadas ao usar o Subversion. Também usado para a árvore de ports quando não estiver usando o portsnap(8). Mais uma vez, escolha um local próximo. Uma lista de espelhos oficiais do Subversion podem ser encontrados na seção sobre Subversion do Handbook do FreeBSD.

10.5.3. Criando Poudriere Jails

Crie as jails de base que serão usadas pelo Poudriere para as compilações:

```
# poudriere jail -c -j 113Ramd64 -v 11.3-RELEASE -a amd64
```

Baixe a versão `11.3-RELEASE` para `amd64` do servidor FTP dado por `FREEBSD_HOST` dentro do poudriere.conf, crie o sistema de arquivos com zfs em `tank/poudriere/jails/113Ramd64`, monte-o em /poudriere/jails/113Ramd64 e extrai os tarballs `11.3-RELEASE` neste sistema de arquivos.

```
# poudriere jail -c -j 11i386 -v stable/11 -a i386 -m svn+https
```

Criado o `tank/poudriere/jaulas/11i386` monte-o em /poudriere/jails/11i386, então confira a dica do Subversion branch do `FreeBSD-11-STABLE` a partir do `SVN_HOST` dentro do poudriere.conf para dentro de /poudriere/jails/11i386/usr/src, e então complete um `buildworld` e instale-o em /poudriere/jails/11i386.

Se uma determinada revisão do Subversion é necessária, anexe ela à string de versão. Por exemplo:

```
# poudriere jail -c -j 11i386 -v stable/11@123456 -a i386 -m svn+https
```

Embora seja possível compilar uma versão mais nova do FreeBSD em uma versão mais antiga, na maioria das vezes ela não irá executar. Por exemplo, se uma jail `stable/11` é necessária, o host terá que rodar `stable/11` também. Rodar `11.3-RELEASE` não é o suficiente.

Para criar uma jail Poudriere para o `13.0-CURRENT`:

```
# poudriere jail -c -j 13amd64 -v head -a amd64 -m svn+https
```

Para executar uma jail `13.0-CURRENT` no Poudriere você deve estar rodando o `13.0-CURRENT`. Em geral, novos kernels podem ser compilados e executar jails mais antigas. Por exemplo, um kernel `13.0-CURRENT` pode compilar e executar uma jail `11.3-STABLE` no Poudriere se a opção de kernel `COMPAT_FREEBSD11` tiver sido compilada (habilitada por padrão na configuração do kernel GENERIC do `13.0-CURRENT`).

O protocolo padrão `svn` funciona normalmente, mas não é muito seguro. Usar `svn+https` juntamente com a verificação do fingerpprint SSL do servidor remoto é aconselhável. Isso garantirá que os arquivos usados para compilar a jail sejam de uma fonte confiável.

Uma lista de jails atualmente conhecidas pelo Poudriere podem ser mostradas com `poudriere jail -l`:

```
# poudriere jail -l
JAILNAME            VERSION             ARCH    METHOD
113Ramd64           11.3-RELEASE        amd64   ftp
11i386              11.3-STABLE         i386    svn+https
```

10.5.4. Mantendo as Jails do Poudriere Atualizadas

Gerenciar atualizações é muito simples. O comando:

```
# poudriere jail -u -j JAILNAME
```

atualiza a jail especificada para a versão mais recente disponível. Para releases do FreeBSD, atualiza para o patchlevel mais recente com o freebsd-update(8). Para versões do FreeBSD compiladas a partir do código fonte, atualiza para a revisão mais recente na branch do Subversion.

Para jails que empregam um método `svn+*`, é útil adicionar `-J NumberOfParallelBuildJobs` para acelerar a compilação aumentando o número de

trabalhos de compilação paralelos utilizados. Por exemplo, se a máquina de compilação tiver 6 CPUs, use:

```
# poudriere jail -u -J 6 -j JAILNAME
```

10.5.5. Configurando a Árvores de Ports para Uso com o Poudriere

Existem várias maneiras de usar árvores de ports no Poudriere. A maneira mais direta é o Poudriere criar uma árvore de ports padrão para si mesmo usando portsnap(8) (se estiver executando FreeBSD 12.1 ou 11.4) ou Subversion (se estiver executando FreeBSD-CURRENT):

```
# poudriere ports -c -m portsnap
```

ou

```
# poudriere ports -c -m svn+https
```

Estes comandos criam `tank/poudriere/ports/default`, monta-o em /poudriere/ports/default e o povoa usando o portsnap(8) ou Subversion. Depois disso, ele é incluído na lista de árvores de ports conhecidas:

```
# poudriere ports -l
PORTSTREE METHOD    TIMESTAMP           PATH
default   svn+https 2020-07-20 04:23:56 /poudriere/ports/default
```

Note que a árvore de ports "default" é especial. Cada um dos comandos de compilação explicados posteriormente usará implicitamente essa árvore de ports, a menos que seja especificamente especificado de outra forma. Para usar outra árvore, adicione `-p treename` aos comandos.

Embora seja útil para compilações em massa regulares, ter esta árvore de ports padrão com o método portsnap(8) pode não ser a melhor maneira de lidar com modificações locais para um contribuidor de ports. Assim como na criação dos jails, é possível usar um método diferente para criar a árvore de ports. Para adicionar uma árvore de ports adicional para testar modificações locais e para o desenvolvimento de ports, baixar a árvore via Subversion (como descrito acima) é preferido:

Os métodos http e https precisam que o devel/subversion seja compilado com a opção `SERF` ativada. Ela vem habilitada por padrão.

O método `svn` permite qualificadores extras para dizer ao Subversion exatamente como buscar os dados. Isso é explicado em poudriere(8). Por exemplo, `poudriere ports -c -m svn+ssh -p subversive` usa o SSH para o checkout.

10.5.6. Usando Árvores de Ports Gerenciadas Manualmente com o Poudriere

Dependendo do fluxo de trabalho, pode ser extremamente útil usar árvores de ports que são mantidas manualmente. Por exemplo, se houver uma cópia local da árvore de ports em /work/ports, aponte o Poudriere para o local:

- Para o Poudriere anterior à versão 3.1.20:

```
# poudriere ports -c -F -f none -M /work/ports -p development
```

- Para o Poudriere versão 3.1.20 e posterior:

```
# poudriere ports -c -m null -M /work/ports -p development
```

Isto será listado na tabela de árvores conhecidas:

```
# poudriere ports -l
PORTSTREE   METHOD    TIMESTAMP           PATH
development null      2020-07-20 05:06:33 /work/ports
```

> O traço ou `null` na coluna `METHOD` significa que o Poudriere nunca irá atualizar ou alterar esta árvore de ports. É de responsabilidade total do usuário a manutenção desta árvore, incluindo todas as modificações locais que podem ser usadas para testar novos ports e enviar patches.

10.5.7. Mantendo as Árvores de Ports do Poudriere Atualizadas

Tão simples quanto com as jails descritas anteriormente:

```
# poudriere ports -u -p PORTSTREE
```

Vai atualizar a *PORTSTREE*, uma árvore dada pela saída de `poudriere -l`, para a última revisão disponível nos servidores oficiais.

> As arvores de ports sem um método, veja Usando Árvores de Ports Gerenciadas Manualmente com o Poudriere, não podem ser atualizadas assim. Elas devem ser atualizadas manualmente pelo mantenedor de ports.

10.5.8. Testando Ports

Depois que as jails e as árvores de ports foram configuradas, o resultado das modificações de um colaborador na árvore de ports pode ser testado.

Por exemplo, modificações locais no port www/firefox localizado em /work/ports/www/firefox pode

ser testado na jail 11.3-RELEASE criada anteriormente:

```
# poudriere testport -j 113Ramd64 -p development -o www/firefox
```

Isso irá compilar todas as dependências do firefox. Se uma dependência foi criada anteriormente e ainda está atualizada, o pacote pré-criado é instalado. Se uma dependência não tiver um pacote atualizado, ela será compilada com opções padrão em uma jail. Depois disso o firefox será compilado.

A compilação completa de cada port será registrada em /poudriere/data/logs/bulk/113Ri386-development/build-time/logs.

O nome do diretório `113Ri386-development` é derivado dos argumentos para `-j` e `-p`, respectivamente. Por conveniência, um link simbólico /poudriere/data/logs/bulk/113Ri386-development/latest também é mantido. O link aponta para o mais recente diretório *build-time*. Neste diretório também se encontra um index.html para que possa ser possível observar o processo de compilação com um navegador web.

Por padrão, o Poudriere limpa as jails e deixa os arquivos de log nos diretórios mencionados acima. Para facilitar a investigação, as jails podem ser mantidas em execução após a compilação, adicionando a opção `-i` ao `testport`:

```
# poudriere testport -j 113Ramd64 -p development -i -o www/firefox
```

Depois que a compilação é concluída, e independentemente de ter sido bem-sucedida, um shell é fornecido dentro da jail. O shell é usado para investigações adicionais. O Poudriere pode ser dito para deixar a jail em execução após a conclusão da compilação com `-i`. O Poudriere mostrará o comando para ser executado quando a jail não for mais necessária. E então é possível fazer um jexec(8) para dentro dele:

```
# poudriere testport -j 113Ramd64 -p development -I -o www/firefox
[...]
====>> Installing local Pkg repository to /usr/local/etc/pkg/repos
====>> Leaving jail 113Ramd64-development-n running, mounted at
/poudriere/data/.m/113Ramd64-development/ref for interactive run testing
====>> To enter jail: jexec 113Ramd64-development-n env -i TERM=$TERM /usr/bin/login
-fp root
====>> To stop jail: poudriere jail -k -j 113Ramd64 -p development
# jexec 113Ramd64-development-n env -i TERM=$TERM /usr/bin/login -fp root
# [do some stuff in the jail]
# exit
# poudriere jail -k -j 113Ramd64 -p development
====>> Umounting file systems
```

Uma parte integral da infraestrutura de compilação de ports do FreeBSD é a capacidade de ajustar os ports as preferências pessoais por meio de opções. Elas podem ser testadas com o Poudriere também. Adicionando a opção `-c`:

```
# poudriere testport -c -o www/firefox
```

Apresenta o diálogo de configuração do port antes que o port seja compilado. Os ports informados após o `-o` no formato `category/portname` usará as opções especificadas, todas as dependências usarão as opções padrão. O teste de ports dependentes com opções não padrão pode ser realizado usando conjuntos, consulte Usando Conjuntos.

Ao testar ports nos quais o pkg-plist é alterado durante a compilação, dependendo das opções selecionadas, é recomendável executar um teste com todas as opções selecionadas *e* um com todas as opções desmarcadas.

10.5.9. Usando Conjuntos

Para todas as ações envolvendo builds, um então chamado *conjunto* pode ser especificado usando `-z setname`. Um conjunto se refere a uma compilação totalmente independente. Isso permite, por exemplo, o uso de `testport` com opções não padrão para os ports dependentes.

Para usar sets, o Poudriere espera uma estrutura de diretórios existente semelhante a `PORT_DBDIR`, o padrão é /var/db/ports no seu diretório de configuração. Este diretório é então nullfs(5)-montado nas jails onde os ports e suas dependências são compilados. Normalmente, um ponto de partida adequado pode ser obtido copiando de forma recursiva o `PORT_DBDIR` para /usr/local/etc/poudriere.d/jailname-portname-setname-options. Isso é descrito em detalhes em poudriere(8). Por exemplo, para testar o www/firefox em um conjunto específico chamado `devset`, adicione o parâmetro `-z devset` ao comando testport:

```
# poudriere testport -j 113Ramd64 -p development -z devset -o www/firefox
```

Isso irá procurar pela existência desses diretórios nesta ordem:

- /usr/local/etc/poudriere.d/113Ramd64-development-devset-options
- /usr/local/etc/poudriere.d/113Ramd64-devset-options
- /usr/local/etc/poudriere.d/113Ramd64-development-options
- /usr/local/etc/poudriere.d/devset-options
- /usr/local/etc/poudriere.d/development-options
- /usr/local/etc/poudriere.d/113Ramd64-options
- /usr/local/etc/poudriere.d/options

Desta lista, o Poudrierenullfs(5)-monta a *primeira árvore existente* de diretório para o diretório /var/db/ports das jails de compilação. Portanto, todas as opções personalizadas são usadas para todos os ports durante essa execução do `testport`.

Depois que a estrutura de diretório para um conjunto é fornecida, as opções para um port específico podem ser alteradas. Por exemplo:

```
# poudriere options -c www/firefox -z devset
```

O diálogo de configuração para o www/firefox é mostrado e as opções podem ser editadas. As opções selecionadas são salvas no set `devset`.

Poudriere é muito flexível na configuração das opções. Elas podem ser configuradas para jails específicas, árvores de ports e para vários ports por um comando. Veja poudriere(8) para detalhes.

10.5.10. Fornecendo um Arquivo make.conf Customizado

Semelhante ao uso de conjuntos (sets), o Poudriere também usará um make.conf personalizado se for fornecido. Nenhum argumento de linha de comando especial é necessário. Em vez disso, o Poudriere procura por arquivos existentes que correspondam a um esquema de nomes derivado da linha de comando. Por exemplo:

```
# poudriere testport -j 113Ramd64 -p development -z devset -o www/firefox
```

faz o Poudriere verificar a existência desses arquivos nesta ordem:

- /usr/local/etc/poudriere.d/make.conf
- /usr/local/etc/poudriere.d/devset-make.conf
- /usr/local/etc/poudriere.d/development-make.conf
- /usr/local/etc/poudriere.d/113Ramd64-make.conf
- /usr/local/etc/poudriere.d/113Ramd64-development-make.conf
- /usr/local/etc/poudriere.d/113Ramd64-devset-make.conf
- /usr/local/etc/poudriere.d/113Ramd64-development-devset-make.conf

Ao contrário dos conjuntos, todos os arquivos encontrados serão anexados, *naquela ordem*, em um make.conf dentro das jails de compilação. Assim, é possível ter variáveis gerais, destinadas a afetar todas as compilações /usr/local/etc/poudriere.d/make.conf. Variáveis especiais, destinadas a afetar apenas determinadas jails ou conjuntos, podem ser setadas em arquivos especiais como make.conf, assim como /usr/local/etc/poudriere.d/113Ramd64-development-devset-make.conf.

Exemplo 111. Usando make.conf para Alterar o Perl Padrão

Para compilar um conjunto com uma versão não padrão do Perl, por exemplo, `5.20`, usando um conjunto chamado `perl5-20`, crie um perl5-20-make.conf com esta entrada:

```
DEFAULT_VERSIONS+= perl=5.20
```

Observe o uso de `+=` de modo que, se a variável já estiver definida no make.conf padrão, seu conteúdo não será sobrescrito.

10.5.11. Remoção de Distfiles Não Mais Necessários

Poudriere vem com um mecanismo embutido para remover distfiles desatualizados que não são mais usados por qualquer port de uma determinada árvore. O comando

```
# poudriere distclean -p portstree
```

irá escanear a pasta distfiles, `DISTFILES_CACHE` dentro do poudriere.conf, contra a árvore de ports dada pelo argumento `-p portstree` e solicitar a remoção desses distfiles. Para pular o prompt e remover incondicionalmente todos os arquivos não utilizados, o argumento `-y` pode ser adicionado:

```
# poudriere distclean -p portstree -y
```

Capítulo 11. Atualizando um Port

Quando um port não estiver na versão mais recente disponibilizada pelos autores, atualize a sua cópia de trabalho local do /usr/ports. O port pode já ter sido atualizado para a nova versão.

Ao trabalhar com diversos ports, provavelmente será mais fácil usar o Subversion para manter toda a coleção de ports atualizada, conforme descrito no Handbook. Isso trará o benefício adicional de rastrear todas as dependências de ports.

O próximo passo é ver se há uma atualização já pendente. Para fazer isso, existem duas opções. Há uma interface de pesquisa no Relatório de Problemas do FreeBSD (PR) ou banco de dados de bugs. Selecione `Ports & Packages` no menu de seleção múltipla `Product` e digite o nome do port no campo `Summary`.

No entanto, às vezes as pessoas esquecem de colocar o nome do port no campo Resumo de maneira não ambígua. Nesse caso, tente pesquisar o campo `Comment` na seção `Detailled Bug Information`, ou tente o Sistema de Monitoramento de Ports do FreeBSD (também conhecido como `portsmon`). Este sistema tenta classificar os PRs do port por portname. Para procurar por PRs sobre um port específico, use a Visão geral de um port.

Se não houver nenhum PR pendente, o próximo passo é enviar um email para o mantenedor do port, como apresentado em `make maintainer`. Essa pessoa pode já estar trabalhando em uma atualização ou ter algum motivo para não atualizar o port neste momento (devido a, por exemplo, problemas de estabilidade da nova versão), e não há necessidade de duplicar seu trabalho. Note que os ports não mantidos são listadas com um mantenedor `ports@FreeBSD.org`, que é apenas a lista de discussão geral de ports, então enviar emails provavelmente não ajudará nesse caso.

Se o mantenedor lhe pedir para fazer a atualização ou não houver mantenedor, então ajude o FreeBSD preparando a atualização! Por favor, faça isso usando o comando diff(1) do sistema base.

Para criar um `diff` adequado para um único patch, copie o arquivo que precisa de patching para something.orig, salve as alterações em something e depois crie o patch:

```
% diff -u something.orig something > something.diff
```

Caso contrário, use o método `svn diff` (Usando o Subversion para Criar Patches) ou copie o conteúdo do port para um diretório completamente diferente e use o resultado do diff(1) recursivo para os diretórios novos e antigos do port (por exemplo, se o diretório de ports modificado for chamado superedit e o original está na nossa árvore como superedit.bak então salve o resultado do comando `diff -ruN superedit.bak superedit`). Tanto o diff unificado ou como o de contexto é aceito, mas os committers do port geralmente preferem diffs unificados. Observe o uso da opção `-N` — essa é a maneira correta de forçar o diff a lidar adequadamente com o caso de novos arquivos sendo adicionados ou de arquivos antigos sendo excluídos. Antes de nos enviar o diff, por favor, examine a saída para se certificar de que todas as alterações fazem sentido. (Em particular, primeiro limpe os diretórios de trabalho com `make clean`).

Se alguns arquivos foram adicionados, copiados, movidos ou removidos, adicione essas informações ao relatório de problemas, para que o committer que pegar o

patch saiba quais comandos svn(1) executar.

Para simplificar operações comuns com arquivos de patch, use `make makepatch` como descrito em Patching. Existem outras ferramentas, como /usr/ports/Tools/scripts/patchtool.py. Antes de usá-lo, por favor leia /usr/ports/Tools/scripts/README.patchtool.

Se o port não é mantido e você o utiliza ativamente, por favor, considere se voluntariar como o seu mantenedor. O FreeBSD tem mais de 4000 ports sem mantenedores, e esta é uma área onde mais voluntários são sempre necessários. (Para uma descrição detalhada das responsabilidades dos mantenedores, consulte a seção no Developer's Handbook.)

Para enviar o diff, use o formulário de envio de bugs (no produto `Ports & Packages`, e no componente `Individual Port(s)`). Sempre inclua a categoria com o nome do port, seguido por dois pontos e uma breve descrição do problema. Exemplos: `category/portname: add FOO option`; `category/portname: Update to XY`. Por favor mencione quaisquer arquivos adicionados ou deletados na mensagem, pois eles devem ser explicitamente especificados no svn(1) ao fazer o commit. Não comprima ou codifique o diff.

Antes de enviar o bug, revise a seção Escrevendo um relatório de problema no artigo Relatórios de Problemas. Ele contém muito mais informações sobre como escrever relatórios úteis de problemas.

Se a atualização for motivada por preocupações de segurança ou por uma falha grave em um port que já está disponível na arvore, notifique a Equipe de Gerenciamento de Ports portmgr@FreeBSD.org para solicitar imediata recompilação e redistribuição do pacote do port. Caso contrário, usuários desavisados do `pkg` continuarão a instalar a versão antiga via `pkg install` por várias semanas.

Por favor, use o diff(1) ou `svn diff` para criar atualizações para ports existentes. Outros formatos incluem o arquivo inteiro e impossibilitam ver o que mudou. Quando os diffs não são incluídos, toda a atualização pode ser ignorada.

Agora que tudo isso foi feito, leia sobre como manter-se atualizado Mantendo-se Atualizado.

11.1. Usando o Subversion para Criar Patches

Quando possível, envie por favor um svn(1)diff. Eles são mais fáceis de manusear do que os diffs entre diretórios "novos e antigos". Nele é mais fácil de ver o que mudou e também é mais fácil de atualizar o diff no caso de algo ter sido modificado na Coleção de Ports desde que o diff foi gerado, ou no caso do committer pedir que algo seja corrigido. Além disso, um patch gerado com `svn diff` pode ser facilmente aplicado com `svn patch` e irá economizar algum tempo para o committer.

```
% cd ~/my_wrkdir ①
% svn co https://svn.FreeBSD.org/ports/head/dns/pdnsd ②
% cd ~/my_wrkdir/pdnsd
```

① Isso pode ser em qualquer lugar, é claro. Compilações de ports não se limitam ao /usr/ports/.

② O svn.FreeBSD.org é o servidor Subversion público do FreeBSD. Veja Mirrors do Subversion para mais informações.

Enquanto estiver no diretório de ports, faça as alterações necessárias. Se você adicionar, copiar, mover ou remover um arquivo, use o `svn` para registrar essas alterações:

```
% svn add new_file
% svn copy some_file file_copy
% svn move old_name new_name
% svn remove deleted_file
```

Certifique-se de verificar o port usando a lista de verificação Testando o Port e Verificando o Port com `portlint`.

```
% svn status
% svn update ①
```

① Isso tentará mesclar as diferenças entre o patch e a versão do repositório atual. Veja a saída cuidadosamente. A letra na frente de cada nome de arquivo indica o que foi feito com ele. Consulte Prefixos de Atualização de Arquivos do Subversion para uma lista completa.

Tabela 51. Prefixos de Atualização de Arquivos do Subversion

U	O arquivo foi atualizado sem problemas.
G	O arquivo foi atualizado sem problemas (somente quando está trabalhando com um repositório remoto).
M	O arquivo foi modificado e foi mesclado sem conflitos.
C	O arquivo foi modificado e foi mesclado com conflitos.

E se o status `C` for exibido como resultado de um `svn update`, isso significa que algo mudou no repositório Subversion e o svn(1) não foi capaz de mesclar as alterações locais com as do repositório. É sempre uma boa ideia inspecionar as alterações de qualquer maneira, o svn(1) não sabe nada sobre a estrutura de um port, então pode (e provavelmente irá) mesclar coisas que não fazem sentido.

O último passo é fazer um diff(1) unificado das mudanças:

```
% svn diff > ../`make -VPKGNAME`.diff
```

Se os arquivos foram adicionados, copiados, movidos ou removidos, inclua os comandos svn(1)`add`, `copy`, `move` e `remove` que foram usados. O `svn move` ou o `svn copy` deve ser executado antes de aplicar o patch. O `svn add` ou `svn remove` deve ser

executado após o patch ser aplicado.

Envie o patch seguindo as diretrizes de envios de relatórios de problemas.

11.2. UPDATE e MOVED

11.2.1. /usr/ports/UPDATING

Se a atualização do port exigir etapas especiais, como alteração de arquivos de configuração ou execução de um programa específico, ela deverá ser documentada neste arquivo. O formato de uma entrada neste arquivo é:

```
YYYYMMDD:
  AFFECTS: users of portcategory/portname
  AUTHOR: Your name <Your email address>

  Special instructions
```

Quando incluir instruções exatas para o portmaster, portupgrade e/ou instruções ao pkg, por favor, certifique-se de escapar o shell escaping corretamente. Por exemplo, *não* use:

```
# pkg delete -g -f docbook-xml* docbook-sk* docbook[2345]??-* docbook-
4*
```

Como mostrado, o comando só irá funcionar com bourne shells. Em vez disso, use o formato abaixo, que funcionará com ambos bourne shell e c-shell:

```
# pkg delete -g -f docbook-xml\* docbook-sk\* docbook\[2345\]\?\?-\*
docbook-4\*
```

Recomenda-se que a linha AFFECTS contenha uma glob que corresponda a todos os ports afetados pela entrada, para que as ferramentas automatizadas possam analisá-la com a maior facilidade possível. Se uma atualização diz respeito a todas as versõs do BIND 9 o conteúdo de `AFFECTS` deve ser `usuários do dns/bind9*`, *não deve* ser `usuários do BIND 9`

11.2.2. /usr/ports/MOVED

Este arquivo é usado para listar os ports movidos ou removidos. Cada linha no arquivo é composta por nome do port, para onde o port foi movido, quando e por quê. Se o port foi removido, a seção detalhando onde ele estava pode ser deixado em branco. Cada seção deve ser separada pelo caractere `|` (pipe), assim:

```
old name|new name (blank for deleted)|date of move|reason
```

A data deve ser inserida no formato `YYYY-MM-DD`. Novas entradas são adicionadas ao final da lista para mantê-las em ordem cronológica, com a entrada mais antiga no topo da lista.

Se um port foi removido, e depois restaurado, exclua a linha neste arquivo que informa que ele foi removido.

Se um port foi renomeado e depois renomeado de volta para seu nome original, adicione uma nova entrada com o nome intermediário para o nome antigo e remova a entrada antiga para não criar um loop.

Quaisquer alterações devem ser validadas com `Tools/scripts/MOVEDlint.awk`.

Se estiver usando um diretório de ports diferente de /usr/ports, use:

```
% cd /home/user/ports
% env PORTSDIR=$PWD Tools/scripts/MOVEDlint.awk
```

Capítulo 12. Segurança

12.1. Por Que Segurança é Tão Importante

Bugs ocasionalmente são inseridos em software. Indiscutivelmente, os mais perigosos deles são aqueles que abrem vulnerabilidades de segurança. Do ponto de vista técnico, tais vulnerabilidades devem ser fechadas exterminando os bugs que as causam. No entanto, as políticas para lidar com meros bugs e vulnerabilidades de segurança são muito diferentes.

Um bug pequeno típico afeta apenas os usuários que ativaram alguma combinação de opções que acionam o bug. O desenvolvedor acabará lançando um patch seguido de uma nova versão do software, livre do bug, mas a maioria dos usuários não irá se incomodar em fazer a atualização de imediato porque o bug nunca os incomodou. Um bug crítico que pode causar perda de dados representa um problema grave. No entanto, usuários prudentes sabem que muitos acidentes são possíveis, e que além de erros de software, podem levar à perda de dados, e portanto eles fazem backups dos dados importantes, além disso, um bug crítico será descoberto muito rapidamente.

Uma vulnerabilidade de segurança é diferente. Primeiro, ela pode permanecer despercebida por anos, porque geralmente não causa mau funcionamento do software. Segundo, uma parte mal-intencionada pode usá-la para obter acesso não autorizado a um sistema vulnerável, para destruir ou alterar dados confidenciais; no pior dos casos, o usuário nem notará os danos causados. Terceiro, expor um sistema vulnerável geralmente ajuda os invasores a invadir outros sistemas que não poderiam ser comprometidos de outra forma. Portanto, fechar uma vulnerabilidade por si só não é suficiente: notifique a audiência afetada da maneira mais clara e abrangente, o que lhes permitirá avaliar o perigo e tomar as medidas adequadas.

12.2. Corrigindo Vulnerabilidades de Segurança

Embora seja sobre o assunto de ports e pacotes, uma vulnerabilidade de segurança pode inicialmente aparecer na distribuição original ou nos arquivos do port. No primeiro caso, o desenvolvedor de software original provavelmente lançará um patch ou uma nova versão instantaneamente. Atualize o port prontamente com relação à correção do autor. Se a correção for atrasada por algum motivo, marque o port como `FORBIDDEN` ou adicione um arquivo patch no port. No caso de um port vulnerável, conserte o port o mais rápido possível. Em ambos os casos, siga o procedimento padrão para enviar alterações a menos que tenha direitos para enviá-lo diretamente para a árvore de ports.

> **!** Ser um committer de ports não é suficiente para fazer um commit em um port arbitrário. Lembre-se que os ports geralmente possuem mantenedores, e eles devem ser respeitados.

Por favor, certifique-se de que a revisão do port é incrementada assim que a vulnerabilidade for fechada. É assim que os usuários que atualizam pacotes regularmente verão que precisam executar uma atualização. Além disso, um novo pacote será compilado e distribuído por FTP e mirrors WWW, substituindo o vulnerável. Atualize `PORTREVISION` a menos que tenha mudado o `DISTVERSION` durante a correção da vulnerabilidade. Isso é, incremente `PORTREVISION` se adicionar um arquivo de patch ao port, mas não o incremente se atualizar o port para a versão mais recente do software,

pois DISTVERSION já terá sido alterado. Por favor, consulte a seção correspondente para mais informações.

12.3. Mantendo a Comunidade Informada

12.3.1. O Banco de Dados VuXML

Uma medida muito importante e urgente a ser tomada o mais cedo possível ao descobrir uma vulnerabilidade de segurança é notificar a comunidade de usuários de port sobre o perigo. Essa notificação serve a dois propósitos. Primeiro, se o perigo for realmente severo, será prudente aplicar uma solução instantânea. Por exemplo, parar o serviço de rede afetado ou até mesmo desinstalar o port completamente até que a vulnerabilidade seja fechada. Segundo, muitos usuários tendem a atualizar pacotes instalados apenas ocasionalmente. Eles saberão pela notificação que eles *devem* atualizar o pacote o quanto antes assim que uma versão com a correção estiver disponível.

Dado o grande número de ports na árvore, um aviso de segurança não pode ser emitido em cada incidente sem criar um flood e perder a atenção do público quando se tratar de assuntos realmente sérios. Portanto, vulnerabilidades de segurança encontradas em ports são registradas no banco de dados VuXML do FreeBSD. Os membros do Security Officer Team também monitoram os problemas que requerem suas intervenções.

Committers podem eles mesmos atualizar o banco de dados VuXML, ajudar o Security Officer Team e fornecer informações cruciais para a comunidade mais rapidamente. Aqueles que não são committers ou que descobriram uma vulnerabilidade excepcionalmente severa não devem hesitar em contatar o Security Officer Team diretamente, conforme descrito na página Informações de Segurança do FreeBSD.

O banco de dados VuXML é um documento XML. Seu arquivo fonte vuln.xml é mantido dentro do port security/vuxml. Portanto, o caminho completo do arquivo será PORTSDIR/security/vuxml/vuln.xml. Toda vez que uma vulnerabilidade de segurança é descoberta em um port, favor adicionar uma entrada para ela nesse arquivo. Até que esteja familiarizado com o VuXML, a melhor coisa a fazer é encontrar uma entrada existente que seja parecida, depois copiá-la e usá-la como modelo.

12.3.2. Uma Breve Introdução ao VuXML

O formato XML completo é complexo e está muito além do escopo deste livro. No entanto, para obter informações básicas sobre a estrutura de uma entrada VuXML, apenas a noção de tags é necessária. Os nomes de tags XML são colocados entre colchetes angulares. Cada abertura <tag> deve ter um fechamento correspondente </tag>. As tags podem ser aninhadas. Se aninhadas, as tags internas devem ser fechadas antes das externas. Há uma hierarquia de tags, ou seja, regras mais complexas de aninhamento. Isso é semelhante ao HTML. A principal diferença é que o XML é e *X*tensible, isto é, com base na definição de tags personalizadas. Devido à sua estrutura intrínseca, o XML por outro lado coloca dados amórficos de uma maneira mais organizada. O VuXML é especialmente adaptado para marcar descrições de vulnerabilidades de segurança.

Agora considere uma entrada VuXML realista:

```
<vuln vid="f4bc80f4-da62-11d8-90ea-0004ac98a7b9"> ①
  <topic>Several vulnerabilities found in Foo</topic> ②
  <affects>
    <package>
      <name>foo</name> ③
      <name>foo-devel</name>
      <name>ja-foo</name>
      <range><ge>1.6</ge><lt>1.9</lt></range> ④
      <range><ge>2.*</ge><lt>2.4_1</lt></range>
      <range><eq>3.0b1</eq></range>
    </package>
    <package>
      <name>openfoo</name> ⑤
      <range><lt>1.10_7</lt></range> ⑥
      <range><ge>1.2,1</ge><lt>1.3_1,1</lt></range>
    </package>
  </affects>
  <description>
    <body xmlns="http://www.w3.org/1999/xhtml">
      <p>J. Random Hacker reports:</p> ⑦
      <blockquote
        cite="http://j.r.hacker.com/advisories/1">
        <p>Several issues in the Foo software may be exploited
          via carefully crafted QUUX requests.  These requests will
          permit the injection of Bar code, mumble theft, and the
          readability of the Foo administrator account.</p>
      </blockquote>
    </body>
  </description>
  <references> ⑧
    <freebsdsa>SA-10:75.foo</freebsdsa> ⑨
    <freebsdpr>ports/987654</freebsdpr> ⑩
    <cvename>CAN-2010-0201</cvename> ⑪
    <cvename>CAN-2010-0466</cvename>
    <bid>96298</bid> ⑫
    <certsa>CA-2010-99</certsa> ⑬
    <certvu>740169</certvu> ⑭
    <uscertsa>SA10-99A</uscertsa> ⑮
    <uscertta>SA10-99A</uscertta> ⑯
    <mlist
msgid="201075606@hacker.com">http://marc.theaimsgroup.com/?l=bugtraq&amp;m=20388660782
5605</mlist> ⑰
    <url>http://j.r.hacker.com/advisories/1</url> ⑱
  </references>
  <dates>
    <discovery>2010-05-25</discovery> ⑲
    <entry>2010-07-13</entry> ⑳
    <modified>2010-09-17</modified>
  </dates>
</vuln>
```

① Os nomes das tags devem ser auto explicativos, por isso vamos dar uma olhada apenas nos campos que precisam ser preenchidos:

② Esta é a tag de nível superior de uma entrada VuXML. Ela tem um atributo obrigatório, `vid`, especificando um identificador universalmente único (UUID) para essa entrada (entre aspas). Gere um UUID para cada nova entrada VuXML (e não se esqueça de substituí-lo pelo modelo UUID, a menos que esteja escrevendo a entrada desde o início). Use uuidgen(1) para gerar um UUID VuXML.

③ Esta é uma descrição de uma linha do problema encontrado.

④ Os nomes dos pacotes afetados são listados nesta tag. Vários nomes podem ser fornecidos, pois vários pacotes podem ser baseados em um único master port ou produto de software. Isso pode incluir branches stable e de desenvolvimento, versões de localidade e slave ports englobando diferentes escolhas de opções importantes de configuração em build-time.

⑤ Versões afetadas de pacote(s) são especificadas com um ou mais intervalos usando uma combinação de elementos `<lt>`, `<le>`, `<eq>`, `<ge>` e `<gt>`. Verifique se os intervalos de versão fornecidos não se sobrepõem.Em uma especificação de range, `(asterisco) indica o menor número de versão. Em particular, 2.` é menor do que `2.a`. Portanto, um asterisco pode ser usado em um intervalo para corresponder todas as possíveis versões `alfa`, `beta` e `RC`. Por exemplo,`<ge>2.</ge><lt>3.</lt>` irá seletivamente corresponder a cada versão `2.x` enquanto `<ge>2.0</ge><lt>3.0</lt>` não irá, pois a versão `2.r3` será ignorada e a versão `3.b` estará dentro do range.O exemplo acima especifica que as versões afetadas vão de `1.6` até menor que `1.9`, versões `2.x` antes de `2.4_1` e versão `3.0b1`.

⑥ Vários grupos de pacotes relacionados (essencialmente, ports) podem ser listados na seção `<affected>`. Isso pode ser usado se vários produtos de software (como FooBar, FreeBar e OpenBar) crescerem da mesma base de código e ainda compartilharem seus bugs e vulnerabilidades. Observe a diferença de listar vários nomes em uma única seção <package>.

⑦ Os intervalos de versão têm que incluir `PORTEPOCH` e `PORTREVISION` se aplicáveis. Lembre-se que, de acordo com as regras de agrupamento, uma versão com um valor diferente de zero para o `PORTEPOCH` é maior que qualquer versão sem `PORTEPOCH`, por exemplo, `3.0,1` é maior que `3.1` ou até mesmo do que `8.9`.

⑧ Este é um resumo do problema. XHTML é usado neste campo. Pelo menos `<p>` e `</p>` tem que aparecer. Marcações mais complexas podem ser usadas, mas apenas por uma questão de precisão e clareza: Sem enfeitar, por favor. Esta seção contém referências a documentos relevantes. Quanto mais referências aplicadas, melhor.

⑨ Isto é um aviso de segurança do FreeBSD.

⑩ Isto é um relatório de problemas do FreeBSD.

⑪ Isto é um identificador MITRE CVE.

⑫ Isto é um ID de bug do SecurityFocus.

⑬ Isto é um aviso de segurança US-CERT.

⑭ Isto é uma nota de vulnerabilidade US-CERT.

⑮ Isto é um alerta de Cyber Segurança US-CERT.

⑯ Isto é um Alerta Técnico de Cyber Segurança US-CERT.

⑰ Esta é uma URL para uma postagem arquivada em uma lista de discussão. O atributo `msgid` é opcional e pode especificar o ID da mensagem no post.

⑱ Esta é uma URL genérica. Apenas se nenhuma das outras categorias de referência for aplicável.

⑲ Esta é a data em que o problema foi divulgado (*YYYY-MM-DD*).

⑳ Esta é a data quando a entrada foi adicionada (*YYYY-MM-DD*).

Esta é a data em que qualquer informação na entrada foi modificada pela última vez (*YYYY-MM-DD*). Novas entradas não devem incluir este campo. Adicione-a ao editar uma entrada existente.

12.3.3. Testando Alterações no Banco de Dados VuXML

Este exemplo descreve uma nova entrada para uma vulnerabilidade no pacote `dropbear` que foi corrigido na versão `dropbear-2013.59`.

Como pré-requisito, instale uma nova versão do port security/vuxml.

Primeiro, verifique se já existe uma entrada para esta vulnerabilidade. Se houvesse essa entrada, ela deveria corresponder com a versão anterior do pacote, `2013.58`:

```
% pkg audit dropbear-2013.58
```

Se não houver nenhuma, adicione uma nova entrada para esta vulnerabilidade.

```
% cd ${PORTSDIR}/security/vuxml
% make newentry
```

Verifique sua sintaxe e formatação:

```
% make validate
```

Pelo menos um desses pacotes precisa ser instalado:textproc/libxml2, textproc/jade.

Verifique se a seção `<affected>` da entrada irá coincidir com os pacotes corretos:

```
% pkg audit -f ${PORTSDIR}/security/vuxml/vuln.xml dropbear-2013.58
```

Certifique-se de que a entrada não produza correspondências incorretas.

Agora, verifique se as versões corretas do pacote são correspondidas pela entrada:

```
% pkg audit -f ${PORTSDIR}/security/vuxml/vuln.xml dropbear-2013.58 dropbear-2013.59
dropbear-2012.58 is vulnerable:
dropbear -- exposure of sensitive information, DoS
```

```
CVE: CVE-2013-4434
CVE: CVE-2013-4421
WWW: http://portaudit.FreeBSD.org/8c9b48d1-3715-11e3-a624-00262d8b701d.html

1 problem(s) in the installed packages found.
```

A versão anterior é encontrada enquanto a última não.

Capítulo 13. O Que Fazer e Não Fazer

13.1. Introdução

Aqui está uma lista comum de o que fazer ou não, encontrada durante o processo de portabilidade. Verifique o port com relação a essa lista, mas também verifique os ports no banco de dados de PR's que outros enviaram. Envie quaisquer comentários sobre os ports, conforme descrito em Relatórios de Bugs e Comentários Gerais. Verificar os ports no banco de dados de PR's irá tornar o processo mais rápido para que possamos fazer o seu commit e para provar que você sabe o que está fazendo.

13.2. WRKDIR

Não escreva nada em arquivos fora do WRKDIR. WRKDIR é o único lugar garantido com permissão de escrita durante a compilação do port (consulte instalando ports a partir de um CDROM para um exemplo de construção de ports de uma árvore somente leitura). Os arquivos pkg-* podem ser modificados pela redefinição de uma variável em vez de sobrescrever o arquivo.

13.3. WRKDIRPREFIX

Certifique-se de que o port honre a variável WRKDIRPREFIX. A maioria dos ports não precisa se preocupar com isso. Em particular, quando se refere a uma variável WRKDIR de outro port, observe que o local correto é WRKDIRPREFIXPORTSDIR/subdir/name/work e não PORTSDIR/subdir/name/work ou .CURDIR/../../subdir/name/work ou algo assim.

Além disso, se definir WRKDIR, certifique-se de adicionar ${WRKDIRPREFIX}${.CURDIR} na frente.

13.4. Diferenciando Sistemas Operacionais e Versões de OS

Alguns códigos precisam de modificações ou compilação condicional com base na versão do FreeBSD Unix em que ele está sendo executado. A maneira preferida de distinguir as versões do FreeBSD é usar as macros __FreeBSD_version e __FreeBSD__ definidas em sys/param.h. Se este arquivo não estiver incluído, adicione o código,

```
#include <sys/param.h>
```

para o lugar adequado no arquivo .c.

__FreeBSD__ é definido em todas as versões do FreeBSD para seu principal número de versão. Por exemplo, no FreeBSD 9.x, __FreeBSD__ é definido para 9.

```
#if __FreeBSD__ >= 9
#  if __FreeBSD_version >= 901000
     /* 9.1+ release specific code here */
```

```
#  endif
#endif
```

Uma lista completa de valores `FreeBSD_version` está disponível em `ValoresFreeBSD_version`.

13.5. Escrevendo Algo Depois do bsd.port.mk

Não escreva nada depois da linha `.include <bsd.port.mk>`. Isso geralmente pode ser evitado incluindo bsd.port.pre.mk em algum lugar no meio do Makefile e bsd.port.post.mk no fim.

> Inclua o par bsd.port.pre.mk/bsd.port.post.mk ou apenas bsd.port.mk; não misture o uso dos dois.

bsd.port.pre.mk define apenas algumas variáveis, que podem ser usadas em testes no Makefile, bsd.port.post.mk define o restante.

Aqui estão algumas variáveis importantes definidas no arquivo bsd.port.pre.mk (esta não é a lista completa, por favor leia bsd.port.mk para a lista completa).

Variável	Descrição
`ARCH`	A arquitetura retornada por `uname -m` (por exemplo, `i386`)
`OPSYS`	O tipo de sistema operacional, conforme retornado por `uname -s` (por exemplo, `FreeBSD`)
`OSREL`	A versão de lançamento do sistema operacional (por exemplo, `2.1.5` ou `2.2.7`)
`OSVERSION`	A versão numérica do sistema operacional; o mesmo que `__FreeBSD_version`.
`LOCALBASE`	A base da árvore "local" (por exemplo, `/usr/local`)
`PREFIX`	Onde o port se instala (veja mais sobre a variável `PREFIX`).

> Quando `MASTERDIR` for necessário, sempre o defina antes de incluir o bsd.port.pre.mk.

Aqui estão alguns exemplos de coisas que podem ser adicionadas depois do bsd.port.pre.mk:

```
# no need to compile lang/perl5 if perl5 is already in system
.if ${OSVERSION} > 300003
BROKEN= perl is in system
.endif
```

Sempre use tab em vez de espaços após o argumento `BROKEN=`.

13.6. Uso de Declarações exec em Wrapper Scripts

Se o port instalar um script shell cuja finalidade é executar outro programa, e se a execução desse programa for a última ação executada pelo script, certifique-se de executar o programa usando a função exec, por exemplo:

```
#!/bin/sh
exec %%LOCALBASE%%/bin/java -jar %%DATADIR%%/foo.jar "$@"
```

A declaração de exec substitui o processo do shell pelo programa especificado. E se exec for omitido, o processo do shell permanece na memória enquanto o programa está sendo executado e consome desnecessariamente recursos do sistema.

13.7. Faça as Coisas Racionalmente

O Makefile deve fazer as coisas de uma maneira simples e razoável. Tornar algumas linhas mais curtas ou mais legíveis é sempre melhor. Exemplos incluem usar um construtor .if do make em vez de um construtor if do shell, não redefinir do-extract se redefinir EXTRACT* é o suficiente e usar GNU_CONFIGURE ao invés de CONFIGURE_ARGS += --prefix=${PREFIX}.

Se um monte de código novo é necessário para fazer algo, já pode haver uma implementação dele em bsd.port.mk. Embora seja difícil de ler, há muitos problemas aparentemente difíceis para os quais bsd.port.mk já fornece uma solução simples.

13.8. Respeite Ambos CC e CXX

O port deve respeitar tanto CC e CXX. O que queremos dizer com isso é que o port não deve definir os valores dessas variáveis absolutamente, sobrepondo os valores existentes; em vez disso, pode anexar quaisquer valores necessários aos valores existentes. Isso é para que as opções de build que afetam todos os ports possam ser definidas globalmente.

Se o port não respeitar essas variáveis, por favor adicione NO_PACKAGE=ignores either cc or cxx ao Makefile.

Aqui está um exemplo do Makefile respeitando ambos CC e CXX. Note o ?=:

```
CC?= gcc
```

```
CXX?= g++
```

Aqui está um exemplo que não respeita nem CC nem CXX:

```
CC= gcc
```

```
CXX= g++
```

Ambos `CC` e `CXX` podem ser definidos em sistemas FreeBSD no arquivo /etc/make.conf. O primeiro exemplo define um valor se não foi definido anteriormente no arquivo /etc/make.conf, preservando quaisquer definições de todo o sistema. O segundo exemplo atrapalha qualquer coisa previamente definida.

13.9. Respeite `CFLAGS`

O port deve respeitar a variável `CFLAGS`. O que queremos dizer com isso é que o port não deve definir o valor dessa variável absolutamente, substituindo o valor existente. Em vez disso, pode anexar quaisquer valores necessários ao valor existente. Isso é para que as opções de build que afetam todos os ports possam ser definidas globalmente.

Se isso não acontecer, por favor adicione `NO_PACKAGE=ignores cflags` ao Makefile.

Aqui está um exemplo de Makefile respeitando `CFLAGS`. Note o `+=`:

```
CFLAGS+= -Wall -Werror
```

Aqui está um exemplo que não respeita `CFLAGS`:

```
CFLAGS= -Wall -Werror
```

`CFLAGS` são definidas em sistemas FreeBSD no arquivo /etc/make.conf. O primeiro exemplo acrescenta flags adicionais para a variável `CFLAGS`, preservando quaisquer definições de todo o sistema. O segundo exemplo atrapalha qualquer coisa previamente definida.

Remove flags de otimização do Makefile de terceiros. O sistema de variáveis `CFLAGS` contém flags de otimização de todo o sistema. Um exemplo de um Makefile não modificado:

```
CFLAGS= -O3 -funroll-loops -DHAVE_SOUND
```

Usando flags de otimização do sistema, o Makefile seria semelhante a este exemplo:

```
CFLAGS+= -DHAVE_SOUND
```

13.10. Logs de Compilação Detalhados

Faz com que o sistema de build dos ports exiba todos os comandos executados durante o estágio de build. Os registros de build completos são cruciais para depurar problemas de ports.

Exemplo de log de compilação não informativo (ruim):

```
CC     source1.o
CC     source2.o
CCLD   someprogram
```

Exemplo de log de compilação detalhado (bom):

```
cc -O2 -pipe -I/usr/local/include -c -o source1.o source1.c
cc -O2 -pipe -I/usr/local/include -c -o source2.o source2.c
cc -o someprogram source1.o source2.o -L/usr/local/lib -lsomelib
```

Alguns sistemas de build, como o CMake, ninja e GNU configure são configurados para registro detalhado pelo framework do ports. Em outros casos, os ports podem precisar de ajustes individuais.

13.11. Feedback

Envie mudanças aplicáveis e correções para o mantenedor upstream para inclusão na próxima versão do código. Isso torna a atualização para a próxima versão muito mais fácil.

13.12. README.html

README.html não faz parte do port, mas é gerado com o comando `make readme`. Não inclua este arquivo em patches ou commits.

Se o comando `make readme` falhar, certifique-se de que o valor padrão de `ECHO_MSG` não tenha sido modificado pelo port.

13.13. Marcando um Port não Instalável com a variável `BROKEN`, `FORBIDDEN` ou `IGNORE`

Em certos casos, os usuários devem ser impedidos de instalar um port. Existem várias variáveis que podem ser usadas no Makefile de um port para informar ao usuário que o port não pode ser instalado. O valor dessas variáveis make será o motivo que é mostrado aos usuários do porque o port se recusa a se instalar. Por favor, use a variável correta. Cada variável transmite significados radicalmente diferentes, tanto para usuários quanto para sistemas automatizados que dependem do Makefile, assim como o cluster de compilação de ports, FreshPorts e portsmon.

13.13.1. Variáveis

- `BROKEN` é reservada para ports que atualmente não compilam, instalam, desinstalam ou que não são executados corretamente. Use-o para ports em que o problema é considerado temporário.

  Se instruído, o cluster de build ainda tentará compilar eles para ver se o problema subjacente foi resolvido. (No entanto, em geral, o cluster é executado sem isso.)

Por exemplo, use BROKEN quando um port:

  - não compila
  - falha em sua configuração ou no processo de instalação
  - instala arquivos fora do ${PREFIX}
  - não remove todos os seus arquivos corretamente após a desinstalação (no entanto, pode ser aceitável, e desejável, que o port deixe arquivos modificados pelo usuário por fora do port)
  - tem problemas em tempo de execução em sistemas nos quais é necessário que ele seja executado corretamente.

- A variável FORBIDDEN é usada para ports que contêm uma vulnerabilidade de segurança ou causam grande preocupação em relação à segurança de um sistema FreeBSD com um determinado port instalado (por exemplo, um programa de confiança inseguro ou um programa que fornece serviços facilmente exploráveis). Marcar ports como FORBIDDEN assim que um determinado software tiver uma vulnerabilidade e não houver atualização liberada. Idealmente, atualize os ports o mais rápido possível quando uma vulnerabilidade de segurança for descoberta para reduzir o número de hosts vulneráveis do FreeBSD (gostamos de ser conhecidos pela segurança), mas às vezes há um intervalo de tempo entre a divulgação de uma vulnerabilidade e uma versão atualizada do software vulnerável. Não marque um port como FORBIDDEN por qualquer outro motivo que não seja segurança.
- A variável IGNORE é reservada para ports que não devem ser compilados por algum outro motivo. Use-a para ports em que o problema é considerado estrutural. O cluster de build não criará, sob nenhuma circunstância, ports marcados com a variável IGNORE. Por exemplo, use IGNORE quando um port:
  - não funciona na versão instalada do FreeBSD
  - tem um distfile que não pôde ser obtido automaticamente devido a restrições de licenciamento
  - não funciona com algum outro port atualmente instalado (por exemplo, o port depende do www/apache20 mas www/apache22 está instalado)

> Se um port entrar em conflito com um port que está atualmente instalado (por exemplo, se eles instalam um arquivo no mesmo local que executa uma função diferente), use a variável CONFLICTS como uma alternativa. CONFLICTS ajustará IGNORE por si próprio.

13.13.2. Notas de Implementação

Não coloque os valores de BROKEN, IGNORE e variáveis relacionadas entre aspas. Devido à forma como a informação é mostrada ao usuário, o texto das mensagens para cada variável é diferente:

```
BROKEN= fails to link with base -lcrypto
```

```
IGNORE= unsupported on recent versions
```

resultando nesta saída a partir do comando `make describe`:

```
===>  foobar-0.1 is marked as broken: fails to link with base -lcrypto.
```

```
===>  foobar-0.1 is unsupported on recent versions.
```

13.14. Considerações Arquitetônicas

13.14.1. Notas Gerais sobre Arquiteturas

O FreeBSD roda em muito mais arquiteturas de processador do que apenas as conhecidas baseadas em x86. Alguns ports possuem restrições específicas para uma ou mais dessas arquiteturas.

Para a lista de arquiteturas suportadas, execute:

```
cd ${SRCDIR}; make targets
```

Os valores são mostrados no formato `TARGET`/`TARGET_ARCH`. O makevar `ARCH` somente leitura do ports é configurado com base no valor de `TARGET_ARCH`. Os Makefiles dos Ports devem testar o valor deste Makevar.

13.14.2. Marcando um Port como de Arquitetura Neutra

Os ports que não possuem requisitos ou arquivos dependentes de arquitetura são identificados com `NO_ARCH=yes`.

> `NO_ARCH` pretende indicar que não há necessidade de compilar um pacote para cada uma das arquiteturas suportadas. O objetivo é reduzir a quantidade de recursos gastos na compilação e distribuição de pacotes, como largura de banda de rede e espaço em disco em mirrors e na mídia de distribuição. Atualmente, entretanto, nossa infraestrutura de pacotes (por exemplo, gerenciadores de pacotes, mirrors e compiladores de pacotes) não estão configurados para se beneficiar totalmente do `NO_ARCH`.

13.14.3. Marcando um port para ser ignorado apenas em determinadas arquiteturas

- Para marcar um port com `IGNORE` apenas em determinadas arquiteturas, existem duas outras variáveis de conveniência que irão setar automaticamente `IGNORE`: `ONLY_FOR_ARCHS` e `NOT_FOR_ARCHS`. Exemplos:

  ```
  ONLY_FOR_ARCHS= i386 amd64
  ```

```
NOT_FOR_ARCHS=  ia64 sparc64
```

Uma mensagem de `IGNORE` customizada pode ser definida usando as variáveis `ONLY_FOR_ARCHS_REASON` e `NOT_FOR_ARCHS_REASON`. É possível definir entradas por arquitetura com as variáveis `ONLY_FOR_ARCHS_REASON_ARCH` e `NOT_FOR_ARCHS_REASON_ARCH`.

13.14.4. Unknown Title!

- Se um port baixar e instalar binários i386, defina a variável `IA32_BINARY_PORT`. Se esta variável estiver definida,/usr/lib32 deve estar presente para versões IA32 de bibliotecas e o kernel deve suportar compatibilidade com IA32. Se uma dessas duas dependências não forem satisfeitas, `IGNORE` será definido automaticamente.

13.14.5. Considerações Específicas do Cluster

- Alguns ports tentam se ajustar à máquina exata em que estão sendo compilados, definindo `-march=native` para o compilador. Isso deve ser evitado: liste-o em uma opção desativada por padrão ou exclua-o completamente.

  Caso contrário, o pacote padrão produzido pelo cluster de compilação pode não rodar em todas as máquinas desse `ARCH`.

13.15. Marcando um Port para Remoção com `DEPRECATED` ou `EXPIRATION_DATE`

Lembre-se que `BROKEN` e `FORBIDDEN` devem ser usados como um recurso temporário se um port não estiver funcionando. Ports permanentemente quebrados serão removidos da árvore por completo.

Quando fizer sentido, os usuários podem ser avisados sobre uma remoção de port pendente com as variáveis `DEPRECATED` e `EXPIRATION_DATE`. A primeira é uma string que indica porque o port está programado para remoção; a segunda é uma string no formato ISO 8601 (YYYY-MM-DD). Ambos serão mostrados ao usuário.

É possível definir a variável `DEPRECATED` sem uma `EXPIRATION_DATE` (por exemplo, recomendando uma versão mais nova do port), mas o contrário não faz sentido.

Não existe uma política definida sobre o tempo de aviso a ser dado. A prática atual parece ser de um mês para problemas relacionados à segurança e dois meses para problemas de compilação. Isso também dá a algum interessado um pouco de tempo para resolver os problemas.

13.16. Evite o Uso do Construtor `.error`

A maneira correta de um Makefile sinalizar que o port não pode ser instalado devido a algum fator externo (por exemplo, o usuário especificou uma combinação ilegal de opções de compilação) é definir um valor não vazio para `IGNORE`. Este valor será formatado e mostrado ao usuário pelo comando `make install`.

É um equívoco comum usar `.error` para esse propósito. O problema com isso é que muitas ferramentas automatizadas que funcionam com a árvore de ports falharão nessa situação. A ocorrência mais comum disso é encontrada ao tentar construir o arquivo /usr/ports/INDEX (Veja Executando `make describe`). No entanto, comandos ainda mais triviais, como `make maintainer` também irão falhar neste cenário. E Isto não é aceitável.

Exemplo 112. Como Evitar o Uso de `.error`

O primeiro dos próximos dois trechos de Makefile irá fazer o `make index` falhar, enquanto o segundo não:

```
.error "option is not supported"
```

```
IGNORE=option is not supported
```

13.17. Uso de sysctl

O uso de sysctl é desencorajado, exceto nos targets. Isso porque ele seria processado na execução de qualquer `makevar`, como os usados durante um `make index`, e assim teria que executar o comando, retardando ainda mais esse processo.

Use apenas sysctl(8) através de `SYSCTL`, pois contém o caminho completo e pode ser modificado, se alguém tiver uma necessidade especial.

13.18. Atualizando Distfiles

De vez em quando os autores de software alteram o conteúdo dos distfiles liberados sem alterar o nome do arquivo. Verifique se as alterações são oficiais e se foram realizadas pelo autor. Já aconteceu no passado em que o distfile foi silenciosamente alterado nos servidores de download com a intenção de prejudicar ou comprometer a segurança do usuário final.

Coloque o antigo distfile de lado, faça o download do novo, descompacte-o e compare o conteúdo com o diff(1). Se não houver nada suspeito, atualize o distinfo.

Certifique-se de resumir as diferenças no log do PR e do commit, para que outras pessoas saibam que nada de ruim aconteceu.

Contate os autores do software e confirme as alterações com eles.

13.19. Uso de Padrões POSIX

Os ports do FreeBSD geralmente esperam conformidade com POSIX. Alguns softwares e sistemas de compilação fazem suposições baseadas em um sistema operacional ou ambiente específico que pode causar problemas quando usado em um port.

Não use /proc se houver outras maneiras de obter as informações. Por exemplo, `setprogname(argv[0])` dentro de `main()` e depois getprogname(3) para saber o nome do executável.

Não confie em comportamento não documentado pelo POSIX.

Não registre timestamps no caminho crítico do aplicativo se ele também funcionar sem. Obter registros de timestamps pode ser lento, dependendo da precisão dos registros de timestamp no SO. Se os timestamps forem realmente necessários, determine o quão precisos eles devem ser e use uma API documentada para fornecer a precisão necessária.

Um número razoável de syscalls simples (por exemplo gettimeofday(2), getpid(2)) são muito mais rápidos no Linux™ do que em qualquer outro sistema operacional, devido ao armazenamento em cache e às otimizações de desempenho do vsyscall. Não confie que seus custos sejam baratos em aplicativos de desempenho crítico. Em geral, tente evitar as syscalls se possível.

Não confie no comportamento de sockets específicos do Linux™. Em particular, os tamanhos padrão do buffer de socket são diferentes (chamadas setsockopt(2) com `SO_SNDBUF` e `SO_RCVBUF`, e enquanto o send(2) do Linux™'s bloqueia quando o buffer do socket está cheio, o do FreeBSD falhará e definirá `ENOBUFS` no errno).

Se for necessário depender de um comportamento não padrão, encapsule-o adequadamente em uma API genérica, verifique o comportamento no estágio de configuração e pare se ele estiver ausente.

Verifique as páginas de manual para ver se a função usada é uma interface POSIX (na seção "STANDARDS" da página de manual).

Não assuma que /bin/sh é o bash. Certifique-se de que uma linha de comando passada para system(3) irá funcionar com um shell compatível com POSIX.

Uma lista de bashismos comum está disponível aqui.

Verifique se os cabeçalhos estão incluídos no POSIX ou da maneira recomendada na página do manual. Por exemplo,sys/types.h é muitas vezes esquecido, o que não é tanto um problema para o Linux™ como é para o FreeBSD.

13.20. Miscelânea

Sempre verifique duas vezes os arquivos pkg-descr e pkg-plist. Se estiver revisando um port e uma melhor formulação puder ser alcançada, faça isso.

Não faça mais cópias da Licença GNU General Public License em nosso sistema. Obrigado.

Por favor, tenha cuidado ao notar quaisquer questões legais! Não nos deixe distribuir software ilegalmente!

Capítulo 14. Um Exemplo de Makefile

Aqui está um exemplo de Makefile que pode ser usado para criar um novo port. Certifique-se de remover todos os comentários extras (entre colchetes).

O formato apresentado é o recomendado para ordenar variáveis, linhas vazias entre seções e assim por diante. Esse formato é projetado para que as informações mais importantes sejam fáceis de serem localizadas. Recomendamos usar o portlint para verificar o Makefile.

```
[the header...just to make it easier for us to identify the ports.]
# $FreeBSD: head/pt_BR.ISO8859-1/books/porters-handbook/book.xml 54410 2020-08-05
22:13:01Z dbaio $
[ ^^^^^^^^^ This will be automatically replaced with RCS ID string by SVN
when it is committed to our repository.  If upgrading a port, do not alter
this line back to "$FreeBSD: head/pt_BR.ISO8859-1/books/porters-handbook/book.xml
54410 2020-08-05 22:13:01Z dbaio $".  SVN deals with it automatically.]

[section to describe the port itself and the master site - PORTNAME
 and PORTVERSION or the DISTVERSION* variables are always first,
 followed by CATEGORIES, and then MASTER_SITES, which can be followed
 by MASTER_SITE_SUBDIR.  PKGNAMEPREFIX and PKGNAMESUFFIX, if needed,
 will be after that.  Then comes DISTNAME, EXTRACT_SUFX and/or
 DISTFILES, and then EXTRACT_ONLY, as necessary.]
PORTNAME=   xdvi
DISTVERSION=    18.2
CATEGORIES= print
[do not forget the trailing slash ("/")!
 if not using MASTER_SITE_* macros]
MASTER_SITES=   ${MASTER_SITE_XCONTRIB}
MASTER_SITE_SUBDIR= applications
PKGNAMEPREFIX=  ja-
DISTNAME=   xdvi-pl18
[set this if the source is not in the standard ".tar.gz" form]
EXTRACT_SUFX=   .tar.Z

[section for distributed patches -- can be empty]
PATCH_SITES=    ftp://ftp.sra.co.jp/pub/X11/japanese/
PATCHFILES= xdvi-18.patch1.gz xdvi-18.patch2.gz
[If the distributed patches were not made relative to ${WRKSRC},
 this may need to be tweaked]
PATCH_DIST_STRIP=   -p1

[maintainer; *mandatory*!  This is the person who is volunteering to
 handle port updates, build breakages, and to whom a users can direct
 questions and bug reports.  To keep the quality of the Ports Collection
 as high as possible, we do not accept new ports that are assigned to
 "ports@FreeBSD.org".]
MAINTAINER= asami@FreeBSD.org
COMMENT=    DVI Previewer for the X Window System
```

```
[license -- should not be empty]
LICENSE=    BSD2CLAUSE
LICENSE_FILE=   ${WRKSRC}/LICENSE

[dependencies -- can be empty]
RUN_DEPENDS=    gs:print/ghostscript

[If it requires GNU make, not /usr/bin/make, to build...]
USES= gmake
[If it is an X application and requires "xmkmf -a" to be run...]
USES= imake

[this section is for other standard bsd.port.mk variables that do not]
 belong to any of the above]
[If it asks questions during configure, build, install...]
IS_INTERACTIVE= yes
[If it extracts to a directory other than ${DISTNAME}...]
WRKSRC=     ${WRKDIR}/xdvi-new
[If it requires a "configure" script generated by GNU autoconf to be run]
GNU_CONFIGURE=  yes
[et cetera.]

[If it requires options, this section is for options]
OPTIONS_DEFINE= DOCS EXAMPLES FOO
OPTIONS_DEFAULT=    FOO
[If options will change the files in plist]
OPTIONS_SUB=yes

FOO_DESC=       Enable foo support

FOO_CONFIGURE_ENABLE=   foo

[non-standard variables to be used in the rules below]
MY_FAVORITE_RESPONSE=   "yeah, right"

[then the special rules, in the order they are called]
pre-fetch:
    i go fetch something, yeah

post-patch:
    i need to do something after patch, great

pre-install:
    and then some more stuff before installing, wow

[and then the epilogue]

.include <bsd.port.mk>
```

Capítulo 15. Ordem das Variáveis nos Makefiles de Port

As primeiras seções do Makefile devem sempre vir na mesma ordem. Este padrão faz com que todos possam ler facilmente qualquer port sem ter que procurar variáveis em uma ordem aleatória.

A primeira linha de um Makefile é sempre um comentário contendo o ID de controle de versão do Subversion, seguido por uma linha vazia. Em novos ports, parece assim:

```
# $FreeBSD: head/pt_BR.ISO8859-1/books/porters-handbook/book.xml 54410 2020-08-05
22:13:01Z dbaio $
```

Nos ports existentes, o Subversion expandiu essa entrada ficando assim:

```
# $FreeBSD: head/pt_BR.ISO8859-1/books/porters-handbook/book.xml 54410 2020-08-05
22:13:01Z dbaio $
```

As seções e variáveis descritas aqui são obrigatórias em um port comum. Em um port slave, muitas seções e variáveis podem ser ignoradas.

Cada bloco seguinte deve ser separado do bloco anterior por uma única linha em branco.

Nos blocos a seguir, apenas defina as variáveis que são requeridas pelo port. Defina essas variáveis na ordem em que são mostradas aqui.

15.1. Bloco PORTNAME

Este bloco é o mais importante. Ele define o nome do port, a versão, o local do arquivo de distribuição e a categoria. As variáveis devem estar nesta ordem:

- PORTNAME
- PORTVERSION[1]
- DISTVERSIONPREFIX
- DISTVERSION[1]
- DISTVERSIONSUFFIX
- PORTREVISION
- PORTEPOCH
- CATEGORIES
- MASTER_SITES

- `MASTER_SITE_SUBDIR` (descontinuado)
- `PKGNAMEPREFIX`
- `PKGNAMESUFFIX`
- `DISTNAME`
- `EXTRACT_SUFX`
- `DISTFILES`
- `DIST_SUBDIR`
- `EXTRACT_ONLY`

15.2. Bloco `PATCHFILES`

Este bloco é opcional. As variáveis são:

- `PATCH_SITES`
- `PATCHFILES`
- `PATCH_DIST_STRIP`

15.3. Bloco `MAINTAINER`

Este bloco é obrigatório. As variáveis são:

- `MAINTAINER`
- `COMMENT`

15.4. Bloco `LICENSE`

Este bloco é opcional, embora seja altamente recomendado. As variáveis são:

- `LICENSE`
- `LICENSE_COMB`
- `LICENSE_GROUPS` ou `LICENSE_GROUPS_NAME`
- `LICENSE_NAME` ou `LICENSE_NAME_NAME`
- `LICENSE_TEXT` ou `LICENSE_TEXT_NAME`
- `LICENSE_FILE` ou `LICENSE_FILE_NAME`
- `LICENSE_PERMS` ou `LICENSE_PERMS_NAME_`
- `LICENSE_DISTFILES` ou `LICENSE_DISTFILES_NAME`

Se houver várias licenças, ordene as variáveis LICENSE_VAR_NOME pelo nome de licença.

15.5. Mensagens Genéricas BROKEN/IGNORE/DEPRECATED

Este bloco é opcional. As variáveis são:

- DEPRECATED
- EXPIRATION_DATE
- FORBIDDEN
- BROKEN
- BROKEN_*
- IGNORE
- IGNORE_*
- ONLY_FOR_ARCHS
- ONLY_FOR_ARCHS_REASON*
- NOT_FOR_ARCHS
- NOT_FOR_ARCHS_REASON*

BROKEN_* e IGNORE_* podem ser qualquer variável genérica, por exemplo, IGNORE_amd64, BROKEN_FreeBSD_10, etc. Com exceção das variáveis que dependem de uma variável USES, coloque essas em USES e USE_x. Por exemplo, IGNORE_WITH_PHP só funciona se php estiver definido e a variável BROKEN_SSL somente se ssl estiver definido.

Se o port estiver marcado como BROKEN quando algumas condições forem atendidas, e tais condições puderem ser testadas somente após incluir o bsd.port.options.mk ou bsd.port.pre.mk, então essas variáveis devem ser definidas mais tarde, em O Restante das Variáveis.

15.6. O Bloco de Dependências

Este bloco é opcional. As variáveis são:

- FETCH_DEPENDS
- EXTRACT_DEPENDS
- PATCH_DEPENDS
- BUILD_DEPENDS
- LIB_DEPENDS
- RUN_DEPENDS
- TEST_DEPENDS

15.7. Flavors

Este bloco é opcional.

Comece esta seção com as definições de `FLAVORS`. Continue com as possíveis variáveis assistentes de Flavors. Veja Usando FLAVORS para maiores informações.

Variáveis de definição de construção não disponíveis como assistentes, usando `.if ${FLAVOR:U} == foo` devem ir em abaixo de suas respectivas seções.

15.8. USES e USE_x

Comece esta seção com a definição da variável `USES` e, em seguida, possíveis variáveis `USE_x`.

Mantenha as variáveis relacionadas juntas. Por exemplo, se estiver usando a variável `USE_GITHUB`, coloque sempre as variáveis `GH_*` logo após ela.

15.9. Variáveis Padrão bsd.port.mk

Este bloco de seção é para variáveis que podem ser definidas em bsd.port.mk que não pertencem a nenhum dos blocos de seção anteriores.

A ordem não é importante, no entanto, tente manter variáveis semelhantes juntas. Por exemplo, variáveis `USERS` e `GROUPS`. Variáveis de configuração `CONFIGURE* e*CONFIGURE`. Lista de arquivos e diretórios `PORTDOCS` e `PORTEXAMPLES`.

15.10. Opções e Assistentes

Se o port usa o framework de opções, defina `OPTIONS_DEFINE` e `OPTIONS_DEFAULT`, então as outras variáveis `OPTIONS*, depois as de descrições*DESC`, e então os assistentes de opções. Tente e ordene todas essas variáveis alfabeticamente.

Exemplo 113. Exemplo de Ordenamento das Variáveis de Opções

As opções `FOO` e `BAR` não possuem uma descrição padrão, portanto, é necessário escrever uma. As outras opções já possuem em Mk/bsd.options.desc.mk então escrever uma não é necessário. Opções `DOCS` e `EXAMPLES` usam os assistentes de destino para instalar seus arquivos, eles são mostrados aqui por completo, apesar de pertencerem a Os Targets, então outras variáveis e destinos podem ser inseridos antes deles.

```
OPTIONS_DEFINE= DOCS EXAMPLES FOO BAR
OPTIONS_DEFAULT=    FOO
OPTIONS_RADIO=  SSL
OPTIONS_RADIO_SSL=    OPENSSL GNUTLS
OPTIONS_SUB=    yes

BAR_DESC=       Enable bar support
FOO_DESC=       Enable foo support
```

```
BAR_CONFIGURE_WITH= bar=${LOCALBASE}
FOO_CONFIGURE_ENABLE=   foo
GNUTLS_CONFIGURE_ON=    --with-ssl=gnutls
OPENSSL_CONFIGURE_ON=   --with-ssl=openssl

post-install-DOCS-on:
      ${MKDIR} ${STAGEDIR}${DOCSDIR}
      cd ${WRKSRC}/doc && ${COPYTREE_SHARE} . ${STAGEDIR}${DOCSDIR}

post-install-EXAMPLES-on:
      ${MKDIR} ${STAGEDIR}${EXAMPLESDIR}
      cd ${WRKSRC}/ex && ${COPYTREE_SHARE} . ${STAGEDIR}${EXAMPLESDIR}
```

15.11. O Restante das Variáveis

E então, o restante das variáveis que não são mencionadas nos blocos anteriores.

15.12. Os Targets

Depois que todas as variáveis são definidas, targets opcionais make(1) podem ser definidos. Mantenha `pre-*` antes de `post-*` e na mesma ordem em que as diferentes etapas são executadas:

- `fetch`
- `extract`
- `patch`
- `configure`
- `build`
- `install`
- `test`

Ao usar os assistentes de opções, os targets são classificados alfabeticamente, mas mantenha `*-on` antes do `*-off`. Quando também estiver usando o target principal, mantenha o target principal antes dos opcionais:

```
post-install:
    # install generic bits

post-install-DOCS-on:
    # Install documentation

post-install-X11-on:
    # Install X11 related bits

post-install-X11-off:
```

```
    # Install bits that should be there if X11 is disabled
```

Capítulo 16. Mantendo-se Atualizado

A coleção de Ports do FreeBSD está em constante mudança. Aqui estão algumas informações sobre como se manter atualizado.

16.1. FreshPorts

Uma das maneiras mais fáceis de saber sobre atualizações que já foram submetidas é assinando o FreshPorts. Múltiplos ports podem ser monitoradas. Os mantenedores são fortemente encorajados a se inscrever, porque eles receberão uma notificação não apenas de suas próprias mudanças, mas também de quaisquer mudanças que qualquer outro committer do FreeBSD tenha feito. (Geralmente é necessário para acompanhar as mudanças na estrutura de ports subjacentes - embora seja mais educado receber um aviso antecipado daqueles que submeterem essas mudanças, às vezes isso é negligenciado ou impraticável. Além disso, em alguns casos, as alterações são muito menores por natureza. Esperamos que todos usem seu melhor julgamento nesses casos.)

Para usar o FreshPorts, é necessário uma conta. Aqueles com endereços de email registrados `@FreeBSD.org` verão um link opt-in no lado direito das páginas web. Aqueles que já possuem uma conta no FreshPorts, mas não estão usando endereço de email `@FreeBSD.org` podem alterar o e-mail para `@FreeBSD.org`, se inscrever, e então alterar o email novamente.

O FreshPorts também possui um recurso de teste de sanidade que testa automaticamente cada commit na árvore de ports do FreeBSD. Se inscrito neste serviço, um committer receberá notificações de quaisquer erros que o FreshPorts detectar durante o teste de sanidade de seus commits.

16.2. A interface Web para o Repositório do Código Fonte

É possível visualizar os arquivos no repositório de código fonte usando uma interface web. As alterações que afetam todo o sistema de ports agora são documentadas no arquivo CHANGES. As alterações que afetam os ports individuais agora são documentadas no arquivo UPDATING. No entanto, a resposta definitiva a qualquer questão é, sem dúvida, ler o código fonte do arquivo bsd.port.mk e arquivos associados.

16.3. A Lista de Discussão de Ports do FreeBSD

Como um mantenedor de ports, considere assinar a lista de discussão de ports do FreeBSD. Mudanças importantes na maneira como os ports funcionam serão anunciadas nela e depois serão inseridas no arquivo CHANGES.

Se o volume de mensagens nesta lista de discussão for muito alto, considere seguir a lista de anúncios de ports do FreeBSD que contém apenas os anúncios.

16.4. O Cluster de Compilação de Ports do FreeBSD

Um dos pontos fortes menos divulgados do FreeBSD é que um cluster inteiro de máquinas é dedicado na compilação continua da Coleção de Ports, para cada um dos principais releases do SO e para cada arquitetura Tier-1.

Ports individuais são compilados a menos que sejam especificamente marcados com `IGNORE`. Ports marcados com `BROKEN` ainda sofrem tentativas de compilação, para verificarem se o problema foi resolvido. (Isso é feito passando `TRYBROKEN` para o Makefile do port.)

16.5. Portscout: o Scanner de Distfile de Ports do FreeBSD

O cluster de compilação é dedicado na compilação da última versão de cada port com os distfiles que já foram baixados. No entanto, como a Internet muda continuamente, os distfiles podem desaparecer rapidamente. Portscout, o scanner de distfile de Ports do FreeBSD, tenta consultar cada site de download para cada port e assim descobrir se cada distfile ainda está disponível.Portscout pode gerar relatórios em HTML e enviar emails sobre novos ports disponíveis para aqueles que os solicitam. A menos que não seja assinado de outra forma, os mantenedores são solicitados a verificar periodicamente as mudanças, seja manualmente ou usando o feed RSS.

A primeira página do Portscout fornece o endereço de e-mail do mantenedor de port, o número de ports pelos quais o mantenedor é responsável, o número desses ports com novos distfiles e a porcentagem dos ports que estão desatualizados. A função de pesquisa permite pesquisar por endereço de email de um mantenedor específico e permite também selecionar se apenas os ports desatualizados serão mostrados.

Ao clicar no endereço de email de um mantenedor, uma lista de todos os seus ports é exibida, junto com a categoria do port, o número da versão atual, se há ou não uma nova versão, quando o port foi atualizado e finalmente quando foi sua última checagem. Uma função de pesquisa nesta página permite que o usuário pesquise por um port específico.

Clicar em um nome de port na lista exibe as informações FreshPorts do port.

16.6. O Sistema de Monitoramento de Ports do FreeBSD

Outro recurso útil é o Sistema de Monitoramento de Ports do FreeBSD (também conhecido como `portsmon`). Este sistema compreende em um banco de dados que processa informações de várias fontes e permite que ele seja acessado através de uma interface web. Atualmente, os Relatórios de Problemas (PRs) dos ports, os logs de erros do cluster de compilação e os arquivos individuais da coleção de ports são usados. No futuro, isso será expandido para incluir pesquisa de distfile, bem como outras fontes.

Para começar, use a página de busca Overview of One Port para encontrar todas as informações sobre um port.

Este é o único recurso disponível que mapeia entradas de PR para nomes de ports. Os remetentes de PR nem sempre incluem o nome do port em sua Sinopse, embora preferiríamos que eles o

fizessem. Logo, `portsmon` é um bom lugar para descobrir se um port existente tem algum PR arquivado, qualquer erro de compilação ou se um novo port que o mantenedor está considerando criar já foi submetido.

Capítulo 17. Usando Macros USES

17.1. Uma introdução ao USES

As macros USES facilitam declarar requisitos e configurações de um port. Elas podem adicionar dependências, alterar o comportamento de compilação do port, adicionar metadados a pacotes e assim por diante, tudo selecionando valores simples e predefinidos.

Cada seção deste capítulo descreve um possível valor para USES, juntamente com seus possíveis argumentos. Argumentos são anexados ao valor após dois pontos (:). Vários argumentos são separados por vírgulas (,).

Exemplo 114. Usando Vários Valores

```
USES=   bison perl
```

Exemplo 115. Adicionando um Argumento

```
USES=   tar:xz
```

Exemplo 116. Adicionando Vários Argumentos

```
USES=   drupal:7,theme
```

Exemplo 117. Entrelaçando Tudo Isso Junto

```
USES=   pgsql:9.3+ cpe python:2.7,build
```

17.2. 7z

Argumentos possíveis: (none), p7zip, partial

Extrair usando 7z(1) ao invés de bsdtar(1) e definir EXTRACT_SUFX=.7z. A opção p7zip força uma dependência do 7z a partir de archivers/p7zip se aquele do sistema base não for capaz de extrair os arquivos. EXTRACT_SUFX não é alterado se a opção partial é usada, isso pode ser usado se o arquivo de distribuição principal não tiver extensão .7z.

17.3. ada

Argumentos possíveis: (none), 5, 6

Depende de um compilador capaz de usar Ada e define a variável CC de acordo. O padrão é usar gcc 5 do ports. Use a opção de versão :X para forçar a compilação com uma versão diferente.

17.4. autoreconf

Argumentos possíveis: (none), build

Execute autoreconf. Ele encapsula os comandos aclocal, autoconf, autoheader, automake, autopoint e libtoolize. Cada comando aplica-se a ${AUTORECONF_WRKSRC} /configure.ac ou seu nome antigo ${AUTORECONF_WRKSRC}/configure.in. E se configure.ac define subdiretórios com seus próprios configure.ac usando AC_CONFIG_SUBDIRS, autoreconf irá recursivamente atualizar aqueles também. O argumento :build só adiciona dependências de build-time sobre essas ferramentas, mas não executa o autoreconf. Um port pode definir AUTORECONF_WRKSRC se WRKSRC não contiver o caminho para o configure.ac.

17.5. blaslapack

Argumentos possíveis: (none), atlas, netlib(padrão), gotoblas, openblas

Adiciona dependências das bibliotecas Blas / Lapack.

17.6. bdb

Argumentos possíveis: (none), 48, 5(padrão), 6

Adiciona uma dependência à biblioteca Berkeley DB. O padrão utiliza databases/db5. Também pode depender de databases/db48 ao usar o argumento :48 ou databases/db6 com :6. É possível declarar um intervalo de valores aceitáveis, :48+ procura pela versão maior instalada e utiliza a 4.8 se nenhuma outra estiver instalada. INVALID_BDB_VER pode ser usado para especificar versões que não funcionam com este port. O framework expõe as seguintes variáveis ao port:

BDB_LIB_NAME

O nome da biblioteca Berkeley DB. Por exemplo, ao usar databases/db5, contém db-5.3.

BDB_LIB_CXX_NAME

O nome da biblioteca Berkeley DBC++. Por exemplo, ao usar databases/db5, contém db_cxx-5.3.

BDB_INCLUDE_DIR

A localização do diretório incluso Berkeley DB. Por exemplo, ao usar databases/db5, ele irá conter ${LOCALBASE}/include/db5.

BDB_LIB_DIR

A localização do diretório da biblioteca Berkeley DB. Por exemplo, ao usar databases/db5, contém ${LOCALBASE}/lib.

`BDB_VER`

A versão detectada de Berkeley DB. Por exemplo, se estiver usando `USES=bdb:48+` e Berkeley DB 5 estiver instalado, irá conter `5`.

databases/db48 está obsoleto e não é suportado. Não deve ser usado por nenhum port.

17.7. `bison`

Argumentos possíveis: (none), `build`, `run`, `both`

Utiliza devel/bison por padrão, sem argumentos ou com o argumento `build`, isso implica que `bison` seja uma dependência de build-time, `run` implica como dependência de run-time e `both` implica em dependências build-time e run-time.

17.8. `cabal`

Não devem ser criados Ports de bibliotecas Haskell, veja Bibliotecas Haskell para maiores informações.

Argumentos possíveis: (none), `hpack`

Define valores e targets padrões usados para compilar software Haskell usando o Cabal. Uma dependência de compilação no port do compilador Haskell (GHC) é adicionada. Se o argumento `hpack` for fornecido, uma dependência de compilação do devel/hs-hpack será adicionada e o `hpack` será chamado na etapa de configuração para gerar o arquivo .cabal.

O framework fornece as seguintes variáveis:

`USE_CABAL`

Se o software usar dependências Haskell, liste-as nesta variável. Cada item deve estar presente no Hackage e ser listado no formato `packagename-0.1.2`. As dependências podem ter revisões, especificadas após o símbolo `_`. A geração automática de lista de dependências é suportada, consulte Compilando Aplicações Haskell com `cabal`.

`CABAL_FLAGS`

Lista de flags a serem passadas para o `cabal-install` durante o estágio de configuração e compilação. As flags são passadas sem alterações (verbatim).

`EXECUTABLES`

Lista de arquivos executáveis instalados pelo port. Valor padrão: `${PORTNAME}`. Os itens desta lista são adicionados automaticamente ao pkg-plist.

`SKIP_CABAL_PLIST`

Se definido, não adicione itens `${EXECUTABLES}` ao pkg-plist.

`opt_USE_CABAL`

Adiciona itens ao `${USE_CABAL}`, dependendo da opção `opt`.

`opt_EXECUTABLES`

Adiciona itens ao `${EXECUTABLES}`, dependendo da opção `opt`.

`opt_CABAL_FLAGS`

Se a opção `opt` estiver ativada, acrescente o valor a `${CABAL_FLAGS}`. Caso contrário, anexe `-value` para desativar a flag.

`FOO_DATADIR_VARS`

Para um executável chamado `FOO`, liste os pacotes Haskell, cujos arquivos de dados devem estar acessíveis pelo executável.

17.9. `cargo`

Argumentos possíveis: (none)

Utiliza Cargo para configuração, compilação e testes. Ele pode ser usado para portar aplicativos Rust que usam o sistema de build Cargo. Para obter mais informações, consulte Compilando Aplicações Rust com `cargo`..

17.10. `charsetfix`

Argumentos possíveis: (none)

Previne que o port instale arquivos charset.alias. Estes arquivos devem ser instalados apenas pelo converters/libiconv. `CHARSETFIX_MAKEFILEIN` pode ser definido para um caminho relativo ao `WRKSRC` se charset.alias não for instalado pelo ${WRKSRC}/Makefile.in.

17.11. `cmake`

Argumentos possíveis: (none), `env`, `notall`, `noman`

Utiliza QMake para configuração e compilação.

Por padrão, uma compilação out-of-source é executada, deixando os fontes em `WRKSRC` livres de artefatos de compilação. Com o argumento `insource`, uma compilação in-source será executada. A configuração deste argumento deve ser a exceção quando uma compilação regular out-of-source não funcionar.

Por padrão, o argumento Ninja é usado para a compilação. Em alguns casos isso não funciona corretamente. Com o argumento `noninja`, a compilação irá usar o `make` para as compilações. Ele só deve ser usado se uma compilação baseada no Ninja não funcionar.

Com o argumento `run`, uma dependência de execução é registrada além de uma dependência de compilação.

Para maiores informações, veja Usando o `cmake`.

17.12. `compiler`

Argumentos possíveis: (none), `env` (padrão, implícito) c++17-lang, c++14-lang, c++11-lang, gcc-c++11-lib, c++11-lib, c++0x, c11, openmp, nestedfct, features

Determina qual compilador usar com base em qualquer um desejo. Use c++17-lang se o port precisar de um compilador compatível com c++17, c++14-lang se o port precisar de um compilador compatível com c++14, c++11-lang se o port precisar de um compilador compatível com c++11, gcc-c++11-lib se o port precisar do compilador {gcc-plus-plus} com uma biblioteca c++11, ou c++11-lib se o port precisar de uma biblioteca padrão c++11-ready. Se o port precisar de um compilador que compreenda as funções c++0X, C11, OpenMP ou funções aninhadas, os parâmetros correspondentes deverão ser usados.

Use `features` para solicitar uma lista de recursos suportados pelo compilador padrão. Depois de incluir o arquivo bsd.port.pre.mk o port pode inspecionar os resultados usando estas variáveis:

- `COMPILER_TYPE`: o compilador padrão no sistema, gcc ou clang
- `ALT_COMPILER_TYPE`: o compilador alternativo no sistema, gcc ou clang. Apenas definido se dois compiladores estiverem presentes na base do sistema.
- `COMPILER_VERSION`: os dois primeiros dígitos da versão do compilador padrão.
- `ALT_COMPILER_VERSION`: os dois primeiros dígitos da versão do compilador alternativo, se presente.
- `CHOSEN_COMPILER_TYPE`: o compilador escolhido, gcc ou clang
- `COMPILER_FEATURES`: os recursos suportados pelo compilador padrão. Atualmente lista a biblioteca C++.

17.13. `cpe`

Argumentos possíveis: (none)

Inclui informações da Common Platform Enumeration (CPE) no manifesto do pacote como uma string CPE 2.3 formatada. Veja as especificações CPE para mais detalhes. Para adicionar informações de CPE a um port, siga estas etapas:

1. Procure pelo registro oficial CPE para o produto de software, usando o mecanismo de pesquisa CPE do NVD ou no dicionário oficial CPE (aviso, o arquivo XML é muito grande). *Nunca crie os dados da CPE.*
2. Adicione `cpe` na variável `USES` e compare o resultado de `make -V CPE_STR` com o registro no dicionário CPE. Continue com um passo de cada vez até `make -V CPE_STR` ficar correto.
3. Se o nome do produto (segundo campo, com o valor padrão para `PORTNAME`) estiver incorreto, defina `CPE_PRODUCT`.
4. Se o nome do fornecedor (primeiro campo, com o valor padrão para `CPE_PRODUCT`) estiver

incorreto, defina `CPE_VENDOR`.

5. Se o campo de versão (terceiro campo, com o valor padrão para `PORTVERSION`) estiver incorreto, defina `CPE_VERSION`.
6. Se o campo de atualização (quarto campo, valor padrão vazio) estiver incorreto, defina `CPE_UPDATE`.
7. Se ainda não estiver correto, verifique o arquivo Mk/Uses/cpe.mk para detalhes adicionais, ou entre em contato com o Ports Security Team ports-secteam@FreeBSD.org.
8. Derive o máximo possível do nome CPE a partir de variáveis existentes, tal como as variáveis `PORTNAME` e `PORTVERSION`. Use modificadores de variáveis para extrair as partes relevantes delas, em vez de colocar o nome direto no código.
9. *Sempre* execute `make -V CPE_STR` e verifique a saída antes de fazer o commit de qualquer coisa que mude o `PORTNAME` ou `PORTVERSION` ou qualquer outra variável que é usada para derivar a variável `CPE_STR`.

17.14. `cran`

Argumentos possíveis: (none), `auto-plist`, `compiles`

Utiliza a Comprehensive R Archive Network. Especifique `auto-plist` para gerar automaticamente o arquivo pkg-plist. Especifique `compiles` se o port tiver código que precise ser compilado.

17.15. `desktop-file-utils`

Argumentos possíveis: (none)

Utiliza update-desktop-database a partir de devel/desktop-file-utils. Uma etapa extra de post-install será executada sem interferir em nenhuma etapa de post-install que já esteja no Makefile do port. Uma linha com `@desktop-file-utils` será adicionada ao plist.

17.16. `desthack`

Argumentos possíveis: (none)

Altera o comportamento do GNU configure para suportar corretamente a variável `DESTDIR` no caso do software original não suportar.

17.17. `display`

Argumentos possíveis: (none), *ARGS*

Define um display environment virtual. Se a variável de ambiente `DISPLAY` não estiver definida, então Xvfb é adicionado como uma dependência de compilação e a variável `CONFIGURE_ENV` é estendida com o número do port da instância do Xvfb em execução no momento. O parâmetro *ARGS* é definido como `install` por padrão e controla a fase na qual se inicia e para a exibição virtual.

17.18. dos2unix

Argumentos possíveis: (none)

O port tem arquivos com terminações de linha no formato do DOS que precisam ser convertidos. Inúmeras variáveis podem ser definidas para controlar quais arquivos serão convertidos. O padrão é converter *todos* arquivos, incluindo binários. Veja Substituições Automáticas Simples para exemplos.

- `DOS2UNIX_REGEX`: casa nomes de arquivos com base em uma expressão regular.
- `DOS2UNIX_FILES`: casa com nomes de arquivos literais.
- `DOS2UNIX_GLOB`: casa com nomes de arquivos baseados em um padrão glob.
- `DOS2UNX_WRKSRC`: o diretório onde iniciar as conversões. O padrão é `${WRKSRC}`.

17.19. drupal

Argumentos possíveis: `7`, `module`, `theme`

Automatiza a instalação de um port que é um tema ou módulo Drupal. Use com a versão Drupal que o port está esperando. Por exemplo, `USES=drupal:7,module` diz que este port cria um módulo do Drupal 6. Um tema do Drupal 7 pode ser especificado com `USES=drupal:7,theme`.

17.20. fakeroot

Argumentos possíveis: (none)

Altera alguns comportamentos padrão dos sistemas de compilação para permitir instalar como um usuário normal. Veja https://wiki.debian.org/FakeRoot para mais informações sobre `fakeroot`.

17.21. fam

Argumentos possíveis: (none), `fam`, `gamin`

Usa um File Alteration Monitor como uma dependência de biblioteca, devel/fam ou devel/gamin. Usuários finais podem definir WITH_FAM_SYSTEM para especificar sua preferência.

17.22. firebird

Argumentos possíveis: (none), `25`

Adiciona uma dependência da biblioteca client do banco de dados do Firebird.

17.23. fonts

Argumentos possíveis: (none), `fc`, `fcfontsdir`(padrão), `fontsdir`, `none`

Adiciona uma dependência de tempo de execução nas ferramentas necessárias para registrar fontes. Dependendo do argumento, adiciona entradas para o plist `@fc ${FONTSDIR}`, `@fcfontsdir ${FONTSDIR}`, `@fontsdir ${FONTSDIR}`, ou nenhuma entrada se o argumento for `none`. Valor padrão de `FONTSDIR` é ${PREFIX}/shared/fonts/${FONTNAME} e `FONTNAME` é `${PORTNAME}`. Adiciona `FONTSDIR` para `PLIST_SUB` e `SUB_LIST`

17.24. fortran

Argumentos possíveis: `gcc` (padrão)

Usa o compilador GNU Fortran.

17.25. fuse

Argumentos possíveis: `2` (padrão), `3`

O port irá depender da biblioteca FUSE e irá manipular a dependência do módulo do kernel dependendo da versão do FreeBSD.

17.26. gem

Argumentos possíveis: (none), `noautoplist`

Manipula a compilação com RubyGems. Se `noautoplist` for usado, a lista de empacotameno não será gerada automaticamente.

17.27. gettext

Argumentos possíveis: (none)

Descontinuado. Incluirá ambos `gettext-runtime` e `gettext-tools`.

17.28. gettext-runtime

Argumentos possíveis: (none), `lib` (padrão),`build`, `run`

Utiliza devel/gettext-runtime. Por padrão, sem argumentos ou com o argumento `lib`, implica uma dependência da biblioteca libintl.so. `build` e `run` implicam, respectivamente, uma dependência de gettext em build-time e run-time.

17.29. gettext-tools

Argumentos possíveis: (none), `build` (padrão),`run`

Utiliza devel/gettext-tools. Por padrão, sem argumento ou com o argumento `build`, uma dependência de msgfmt em build-time é registrada. Com o argumento `run`, uma dependência em run-time é registrada.

17.30. ghostscript

Argumentos possíveis: *X*, `build`, `run`, `nox11`

Uma versão *X* específica pode ser usada. Versões possíveis são `7`, `8`, `9` e `agpl` (padrão). `nox11` indica que a versão `-nox11` do port é necessária. `build` e `run` adicionam dependências de Ghostscript em build-time e run-time. O padrão é ambas as dependências, build-time e run-time.

17.31. gl

Argumentos possíveis: (none)

Fornece uma maneira fácil para depender dos componentes GL. Os componentes devem ser listados na variável `USE_GL`. Os componentes disponíveis são:

`egl`
: adiciona uma dependência de biblioteca libEGL.so de graphics/libglvnd

`gbm`
: Adiciona uma dependência de biblioteca libgbm.so de graphics/mesa-libs

`gl`
: Adiciona uma dependência de biblioteca libGL.so de graphics/libglvnd

`glesv2`
: Adiciona uma dependência de biblioteca libGLESv2.so de graphics/libglvnd

`glew`
: Adiciona uma dependência de biblioteca libGLEW.so de graphics/glew

`glu`
: Adiciona uma dependência de biblioteca libGLU.so de graphics/libGLU

`glut`
: Adiciona uma dependência de biblioteca libglut.so de graphics/freeglut

`opengl`
: Adiciona uma dependência de biblioteca libOpenGL.so de graphics/libglvnd

17.32. gmake

Argumentos possíveis: (none)

Utiliza devel/gmake como uma dependência em run-time e configura o ambiente para usar `gmake` como `make` padrão para a compilação.

17.33. gnome

Argumentos possíveis: (none)

Fornece uma maneira fácil para depender dos componentes do GNOME. Os componentes devem ser listados na variável USE_GNOME. Os componentes disponíveis são:

- atk
- atkmm
- cairo
- cairomm
- dconf
- esound
- evolutiondataserver3
- gconf2
- gconfmm26
- gdkpixbuf
- gdkpixbuf2
- glib12
- glib20
- glibmm
- gnomecontrolcenter3
- gnomedesktop3
- gnomedocutils
- gnomemenus3
- gnomemimedata
- gnomeprefix
- gnomesharp20
- gnomevfs2
- gsound
- gtk-update-icon-cache
- gtk12
- gtk20
- gtk30
- gtkhtml3
- gtkhtml4
- gtkmm20

- gtkmm24
- gtkmm30
- gtksharp20
- gtksourceview
- gtksourceview2
- gtksourceview3
- gtksourceviewmm3
- gvfs
- intlhack
- intltool
- introspection
- libartlgpl2
- libbonobo
- libbonoboui
- libgda5
- libgda5-ui
- libgdamm5
- libglade2
- libgnome
- libgnomecanvas
- libgnomekbd
- libgnomeprint
- libgnomeprintui
- libgnomeui
- libgsf
- libgtkhtml
- libgtksourceviewmm
- libidl
- librsvg2
- libsigc++12
- libsigc++20
- libwnck
- libwnck3
- libxml++26
- libxml2

- libxslt
- metacity
- nautilus3
- orbit2
- pango
- pangomm
- pangox-compat
- py3gobject3
- pygnome2
- pygobject
- pygobject3
- pygtk2
- pygtksourceview
- referencehack
- vte
- vte3

A dependência padrão é em built-time e run-time, pode ser alterada com `:build` ou `:run`. Por exemplo:

```
USES=       gnome
USE_GNOME=  gnomemenus3:build intlhack
```

Veja Usando o GNOME para maiores informações.

17.34. go

Não devem ser criados Ports de bibliotecas Go, veja Bibliotecas Go para maiores informações.

Argumentos possíveis: (none), `modules`, `no_targets`, `run`

Define valores e targets padrão usados para compilar aplicações Go. Uma dependência de compilação no port do compilador Go selecionada via `GO_PORT` é adicionada. Por padrão, a compilação é executada no modo GOPATH. Se o software Go usar módulos, o modo de reconhecimento de módulos pode ser ativado com o argumento `modules`. `no_targets` irá configurar o ambiente de compilação com `GO_ENV`, `GO_BUILDFLAGS` mas irá pular os targets `post-extract` e `do-{build,install,test}`. `run` também adicionará uma dependência de tempo de execução do que estiver em `GO_PORT`.

O processo de compilação é controlado por várias variáveis:

`GO_PKGNAME`

O nome do pacote Go ao compilar no modo GOPATH. Este é o diretório que será criado em `${GOPATH}/src`. Se não estiver definido explicitamente e `GH_SUBDIR` ou `GL_SUBDIR` estiverem presente, o valor `GO_PKGNAME` será inferido deles. Isso não é necessário quando compilado no modo de reconhecimento de módulos.

`GO_TARGET`

Os pacotes a serem compilados. O valor padrão é `${GO_PKGNAME}`. `GO_TARGET` também pode ser uma tupla na forma `package:path` onde path pode ser um nome de arquivo simples ou um caminho completo começando com `${PREFIX}`.

`GO_TESTTARGET`

Os pacotes para testar. O valor padrão é `./⋯` (o pacote atual e todos os subpacotes).

`CGO_CFLAGS`

Valores adicionais da variável `CFLAGS` a serem passados para o compilador C pelo `Go`.

`CGO_LDFLAGS`

Valores adicionais da variável `LDFLAGS` a serem passados para o compilador C pelo `Go`.

`GO_BUILDFLAGS`

Argumentos de compilação adicionais para passar para o `go build`.

`GO_TESTFLAGS`

Argumentos de compilação adicionais para passar para o `go test`.

`GO_PORT`

O port do compilador Go a ser utilizado. Por padrão é lang/go mas pode ser definido para lang/go-devel no `make.conf` para testes de futuras versões Go.

Esta variável não deve ser definida por ports individuais!

Ver Compilando Aplicações Go para exemplos de uso.

17.35. `gperf`

Argumentos possíveis: (none)

Adiciona uma dependência devel/gperf em buildtime se `gperf` não estiver presente no sistema base.

17.36. `grantlee`

Argumentos possíveis: `5`, `selfbuild`

Manipula a dependência em Grantlee. Especifique `5` para depender da versão baseada no Qt5, devel/grantlee5. `selfbuild` é usado internamente pelo devel/grantlee5 para obter os números de suas versões.

17.37. groff

Argumentos possíveis: build, run, both

Registra uma dependência de textproc/groff se não estiver presente no sistema base.

17.38. gssapi

Argumentos possíveis: (none), base (padrão), heimdal, mit, flags, bootstrap

Manipular as dependências necessárias para os consumers do GSS-API. Apenas as bibliotecas que fornecem os mecanismos do Kerberos estão disponíveis. Por padrão, ou definido como base, a biblioteca GSS-API do sistema base é usada. Também pode ser definido para heimdal para usar security/heimdal ou mit para usar security/krb5.

Quando a instalação local do Kerberos não está em LOCALBASE defina a variável HEIMDAL_HOME (para heimdal) ou a variável KRB5_HOME (para krb5) para a instalação local do Kerberos.

Essas variáveis são exportadas para os ports para serem usadas:

- GSSAPIBASEDIR
- GSSAPICPPFLAGS
- GSSAPIINCDIR
- GSSAPILDFLAGS
- GSSAPILIBDIR
- GSSAPILIBS
- GSSAPI_CONFIGURE_ARGS

As opções de flags podem estar lado a lado com base, heimdal ou mit para adicionar automaticamente GSSAPICPPFLAGS, GSSAPILDFLAGS e GSSAPILIBS para CFLAGS, LDFLAGS e LDADD, respectivamente. Por exemplo, use base,flags.

A opção bootstrap é um prefixo especial apenas para o uso do security/krb5 e security/heimdal. Por exemplo, use bootstrap,mit.

Exemplo 118. Uso Típico

```
OPTIONS_SINGLE= GSSAPI
OPTIONS_SINGLE_GSSAPI=  GSSAPI_BASE GSSAPI_HEIMDAL GSSAPI_MIT GSSAPI_NONE

GSSAPI_BASE_USES=   gssapi
GSSAPI_BASE_CONFIGURE_ON=   --with-gssapi=${GSSAPIBASEDIR}
${GSSAPI_CONFIGURE_ARGS}
GSSAPI_HEIMDAL_USES=    gssapi:heimdal
GSSAPI_HEIMDAL_CONFIGURE_ON=    --with-gssapi=${GSSAPIBASEDIR}
${GSSAPI_CONFIGURE_ARGS}
GSSAPI_MIT_USES=    gssapi:mit
```

```
GSSAPI_MIT_CONFIGURE_ON=    --with-gssapi=${GSSAPIBASEDIR}
${GSSAPI_CONFIGURE_ARGS}
GSSAPI_NONE_CONFIGURE_ON=   --without-gssapi
```

17.39. horde

Argumentos possíveis: (none)

Adicionar dependências de builtime e runtime em devel/pear-channel-horde. Outras dependências Horde podem ser adicionadas com `USE_HORDE_BUILD` e `USE_HORDE_RUN`. Veja Módulos Horde para maiores informações.

17.40. iconv

Argumentos possíveis: (none), `lib`, `build`, `patch`, `translit`, `wchar_t`

Utilização de funções `iconv`, seja do port converters/libiconv como uma dependência de buil-time e run-time, ou do sistema base em um 10-CURRENT após um `iconv` nativo ser comitado em r254273. Por padrão, sem argumentos ou com o argumento `lib`, implica em `iconv` com dependências de build-time e run-time. `build` implica uma dependência de build-time e `patch` implica uma dependência de patch-time. Se o port usa extensões iconv `WCHAR_T` ou `//TRANSLIT` , adicione os argumentos relevantes para que o iconv correto seja usado. Para mais informações, veja Usando `iconv`.

17.41. imake

Argumentos possíveis: (none), `env`, `notall`, `noman`

Adiciona devel/imake como uma dependência de built-time e executa `xmkmf -a` durante o estágio `configure`. Se o argumento `env` é passado, o target `configure` não é definido. Se a flag `-a` for um problema para o port, adicione o argumento `notall`. E se `xmkmf` não gerar um target `install.man`, adicione o argumento `noman`.

17.42. kde

Argumentos possíveis: `5`

Adiciona dependência de componentes KDE. Veja Usando o KDE para maiores informações.

17.43. kmod

Argumentos possíveis: (none), `debug`

Preenche o boilerplate para os ports de módulo do kernel, atualmente:

- Adiciona `kld` em `CATEGORIES`.

- Define `SSP_UNSAFE`.
- Defina `IGNORE` se as fontes do kernel não forem encontradas em `SRC_BASE`.
- Define `KMODDIR` para /boot/modules por padrão, adiciona isso para a variável `PLIST_SUB` e `MAKE_ENV`, e o cria após a instalação. Se a variável `KMODDIR` está definida para o /boot/kernel, ela será reescrita para /boot/modules. Isso evita quebrar pacotes ao atualizar o kernel devido ao /boot/kernel ser renomeado para /boot/kernel.old no processo.
- Manipula módulos cross-referencing do kernel acerca da instalação e desinstalação, usando `@kld`.
- Se o argumento `debug` é passado, o port pode instalar uma versão de debug do módulo no arquivo KERN_DEBUGDIR/KMODDIR. Por padrão, a variável `KERN_DEBUGDIR` é copiada da `DEBUGDIR` e definido para /usr/lib/debug. O framework irá cuidar da criação e remoção de quaisquer diretórios necessários.

17.44. `lha`

Argumentos possíveis: (none)

Define `EXTRACT_SUFX` para `.lzh`

17.45. `libarchive`

Argumentos possíveis: (none)

Registra uma dependência de archivers/libarchive. Quaisquer ports dependendo de libarchive deve incluir `USES=libarchive`.

17.46. `libedit`

Argumentos possíveis: (none)

Registra uma dependência de devel/libedit. Quaisquer ports dependendo de libedit devem incluir `USES=libedit`.

17.47. `libtool`

Argumentos possíveis: (none), `keepla`, `build`

Scripts `libtool` de patches. Isso deve ser adicionado a todos os ports que usam `libtool`. O argumento `keepla` pode ser usado para manter arquivos .la. Alguns ports não vêm com sua própria cópia da libtool e precisam de uma dependência de devel/libtool em build time, use o argumento `:build` para adicionar essa dependência.

17.48. `linux`

Argumentos possíveis: `c6`, `c7`

Framework de compatibilidade de ports com Linux. Especifique `c6` para depender de pacotes do CentOS 6. Especifique `c7` para depender de pacotes do CentOS 7. Os pacotes disponíveis são:

- `allegro`
- `alsa-plugins-oss`
- `alsa-plugins-pulseaudio`
- `alsalib`
- `atk`
- `avahi-libs`
- `base`
- `cairo`
- `cups-libs`
- `curl`
- `cyrus-sasl2`
- `dbusglib`
- `dbuslibs`
- `devtools`
- `dri`
- `expat`
- `flac`
- `fontconfig`
- `gdkpixbuf2`
- `gnutls`
- `graphite2`
- `gtk2`
- `harfbuzz`
- `jasper`
- `jbigkit`
- `jpeg`
- `libasyncns`
- `libaudiofile`
- `libelf`
- `libgcrypt`
- `libgfortran`
- `libgpg-error`
- `libmng`

- libogg
- libpciaccess
- libsndfile
- libsoup
- libssh2
- libtasn1
- libthai
- libtheora
- libv4l
- libvorbis
- libxml2
- mikmod
- naslibs
- ncurses-base
- nspr
- nss
- openal
- openal-soft
- openldap
- openmotif
- openssl
- pango
- pixman
- png
- pulseaudio-libs
- qt
- qt-x11
- qtwebkit
- scimlibs
- sdl12
- sdlimage
- sdlmixer
- sqlite3
- tcl85
- tcp_wrappers-libs

- `tiff`
- `tk85`
- `ucl`
- `xorglibs`

17.49. `localbase`

Argumentos possíveis: (none), `ldflags`

Garante que as bibliotecas de dependências em `LOCALBASE` sejam usadas em vez das do sistema base. Especifique `ldflags` para adicionar `-L${LOCALBASE}/lib` para a variável `LDFLAGS` ao invés de `LIBS`. Ports que dependem de bibliotecas que também estão presentes no sistema base devem usar isso. Também é usado internamente por algumas outras variáveis `USES`.

17.50. `lua`

Argumentos possíveis: (none), `XY`, `XY+`, `-XY`, `XY-ZA`, `module`, `flavors`, `build`, `run`, `env`

Adiciona uma dependência de Lua. Por padrão, esta é uma dependência de biblioteca, a menos que seja invalidado por uma opção `build` ou `run`. A opção `env` evita a adição de qualquer dependência, enquanto ainda define todas as variáveis usuais.

A versão padrão é definida pelo mecanismo usual `DEFAULT_VERSIONS`, a menos que uma versão ou intervalo de versões seja especificado como um argumento, por exemplo, `51` or `51-53`.

Os aplicativos que usam Lua são normalmente compilados para apenas uma única versão do Lua. No entanto, os módulos de biblioteca destinados a serem carregados pelo código Lua devem usar a opção `module` para compilar com vários flavors.

Para maiores informações, veja Usando Lua.

17.51. `lxqt`

Argumentos possíveis: (none)

Manipular dependências para o LXQt Desktop Environment. Use a variável `USE_LXQT` para selecionar os componentes necessários para o port. Veja Usando o LXQt para maiores informações.

17.52. `makeinfo`

Argumentos possíveis: (none)

Adiciona uma dependência de build-time em `makeinfo` se o mesmo não estiver presente no sistema base.

17.53. `makeself`

Argumentos possíveis: (none)

Indica que os arquivos de distribuição são archives makeself e define as dependências apropriadas.

17.54. `mate`

Argumentos possíveis: (none)

Fornece uma maneira fácil para depender de componentes do MATE. Os componentes devem ser listados em `USE_MATE`. Os componentes disponíveis são:

- `autogen`
- `caja`
- `common`
- `controlcenter`
- `desktop`
- `dialogs`
- `docutils`
- `icontheme`
- `intlhack`
- `intltool`
- `libmatekbd`
- `libmateweather`
- `marco`
- `menus`
- `notificationdaemon`
- `panel`
- `pluma`
- `polkit`
- `session`
- `settingsdaemon`

A dependência padrão é em built-time e run-time, pode ser alterada com `:build` ou `:run`. Por exemplo:

```
USES=       mate
USE_MATE=   menus:build intlhack
```

17.55. meson

Argumentos possíveis: (none)

Fornece suporte para projetos baseados no Meson. Para maiores informações, consulte Usando meson.

17.56. metaport

Argumentos possíveis: (none)

Define as seguintes variáveis para facilitar a criação de um metaport: MASTER_SITES, DISTFILES, EXTRACT_ONLY, NO_BUILD, NO_INSTALL, NO_MTREE, NO_ARCH.

17.57. mysql

Argumentos possíveis: (none), version, client (padrão), server, embedded

Fornece suporte para o MySQL. Se nenhuma versão for informada, tenta encontrar a versão atual instalada. Fall back para a versão padrão, MySQL-5.6. As possíveis versões são 55, 55m, 55p, 56, 56p, 56w, 57, 57p, 80, 100m, 101m e 102m. Os sufixos m e p são para MariaDB e Percona, variantes do MySQL. server e embbeded adicionam uma dependência de build- e run-time do servidor MySQL. Ao usar server ou embbeded, é adicionado client para também adicionar uma dependência no arquivo libmysqlclient.so. Um port pode definir IGNORE_WITH_MYSQL se algumas versões não forem suportadas.

O framework define a variável MYSQL_VER para a versão detectada do MySQL.

17.58. mono

Argumentos possíveis: (none), nuget

Adiciona uma dependência no framework Mono (atualmente apenas C#) definindo as dependências apropriadas.

Especifique nuget quando o port usa pacotes nuget. NUGET_DEPENDS precisa ser definido com os nomes e versões dos pacotes nuget no formato name=version. Uma pacote de origem opcional pode ser adicionado usando name=version:origin.

O target auxiliar, buildnuget, exibirá o conteúdo da variável NUGET_DEPENDS com base no arquivo packages.config fornecido.

17.59. motif

Argumentos possíveis: (none)

Utiliza x11-toolkits/open-motif como uma dependência de biblioteca. Os usuários finais podem definir WANT_LESSTIF para a dependência estar em x11-toolkits/lesstif ao invés de x11-toolkits/open-

motif.

17.60. ncurses

Argumentos possíveis: (none), base, port

Utiliza ncurses, e faz com que algumas variáveis úteis sejam definidas.

17.61. ninja

Argumentos possíveis: (none)

Utiliza ninja para compilar o port.

17.62. objc

Argumentos possíveis: (none)

Adiciona dependências de objetive C (compilador, biblioteca de runtime) se o sistema base não suportar isto.

17.63. openal

Argumentos possíveis: al, soft (padrão), yes, alut

Utiliza OpenAL. O backend pode ser especificado, com a implementação do software como padrão. O usuário pode especificar um backend preferido com WANT_OPENAL. Os valores válidos para este manipulador são soft (padrão) e si.

17.64. pathfix

Argumentos possíveis: (none)

Procura pelos arquivos Makefile.in e configure na variável PATHFIX_WRKSRC (padrão é WRKSRC) e corrige os caminhos comuns para garantir que eles respeitem a hierarquia do FreeBSD. Por exemplo, ele corrige o diretório de instalação dos arquivos .pc do pkgconfig para ${PREFIX}/libdata/pkgconfig. Se o port usa USES=autoreconf, Makefile.am será adicionado automaticamente a PATHFIX_MAKEFILEIN.

Se o port tem definido USES=cmake ele vai procurar pelo arquivo CMakeLists.txt dentro da variável PATHFIX_WRKSRC. Se necessário, esse nome de arquivo padrão pode ser alterado com PATHFIX_CMAKELISTSTXT.

17.65. pear

Argumentos possíveis: env

Adiciona uma dependência do devel/pear. Ele irá configurar o comportamento padrão do software usando o Repositório de Extensão e Aplicativos do PHP. O uso do argumento `env` apenas configura as variáveis de ambiente PEAR. Veja Módulos PEAR para maiores informações.

17.66. `perl5`

Argumentos possíveis: (none)

Depende do Perl. A configuração é feita usando a variável `USE_PERL5`.

`USE_PERL5` pode conter as fases que precisam usar Perl, pode ser `extract`, `patch`, `build`, `run` ou `test`.

`USE_PERL5` também pode conter `configure`, `modbuild` ou `modbuildtiny` quando Makefile.PL, Build.PL ou Módulo::Build::Tiny's, flavor de Build.PL é necessário.

O padrão de `USE_PERL5` é `build run`. Ao usar `configure`, `modbuild` ou `modbuildtiny`, uso de `build` e `run` são implícitos.

Veja Usando Perl para maiores informações.

17.67. `pgsql`

Argumentos possíveis: (none), `X.Y`, `X.Y+`, `X.Y-`, `X.Y-Z.A`

Fornece suporte para o PostgreSQL. O mantenedor do port pode definir a versão requisitada. Podem ser especificadas versões mínima e máxima ou um intervalo; por exemplo, `9.0-`, `8.4+`, `8.4-9.2`.

Por padrão, a dependência adicionada será o cliente, mas se o port exigir componentes adicionais, isso poderá ser feito usando `WANT_PGSQL=component[:target]`; por exemplo, `WANT_PGSQL=server:configure pltcl plperl`. Os componentes disponíveis são:

- `client`
- `contrib`
- `docs`
- `pgtcl`
- `plperl`
- `plpython`
- `pltcl`
- `server`

17.68. `php`

Argumentos possíveis: (none), `phpize`, `ext`, `zend`, `build`, `cli`, `cgi`, `mod`, `web`, `embed`, `pecl`, `flavors`, `noflavors`

Fornece suporte para o PHP. Adiciona uma dependência de run-time na versão padrão do PHP,

lang/php56.

`phpize`
: Utilizado para compilar uma extensão do PHP. Habilita flavors.

`ext`
: Usado para compilar, instalar e registrar uma extensão do PHP. Habilita flavors.

`zend`
: Usado para criar, instalar e registrar uma extensão do Zend. Habilita flavors.

`build`
: Define PHP também como uma dependência de build-time.

`cli`
: Precisa da versão CLI do PHP.

`cgi`
: Precisa da versão CGI do PHP.

`mod`
: Precisa do módulo Apache para o PHP.

`web`
: Precisa do módulo Apache ou a versão CGI do PHP.

`embed`
: Precisa da versão da biblioteca embarcada do PHP.

`pecl`
: Fornece padrões para baixar extensões PHP do repositório PECL. Habilita flavors.

`flavors`
: Habilita a geração de PHP flavors automático. Flavors serão gerados para todas as versões do PHP, exceto as presentes na variável `IGNORE_WITH_PHP`.

`noflavors`
: Desativa a geração automática de flavors do PHP. *Deve apenas* ser usado com extensões fornecidas pelo próprio PHP.

Variáveis são usadas para especificar quais módulos PHP são necessários, bem como qual versão do PHP são suportadas.

`USE_PHP`
: A lista das extensões PHP requisitadas em run-time. Adicione `:build` ao nome da extensão para adicionar uma dependência em build-time. Exemplo: `pcre xml:build gettext`

`IGNORE_WITH_PHP`

O port não funciona com a versão do PHP fornecida. Para possíveis valores, observe o conteúdo da variável `_ALL_PHP_VERSIONS` no arquivo Mk/Uses/php.mk.

Ao compilar uma extensão do PHP ou Zend com `:ext` ou `:zend`, estas variáveis podem ser definidas:

`PHP_MODNAME`

O nome da extensão do PHP ou Zend. O valor padrão é `${PORTNAME}`.

`PHP_HEADER_DIRS`

Uma lista de subdiretórios dos quais instalar arquivos header. O framework sempre irá instalar os arquivos header que estão presentes no mesmo diretório que a extensão.

`PHP_MOD_PRIO`

A prioridade na qual carregar a extensão. É um número entre `00` e `99`.

Para extensões que não dependem de nenhuma extensão, a prioridade é definida automaticamente como `20`, para extensões que dependem de outra extensão, a prioridade é definida automaticamente como `30`. Algumas extensões podem precisar ser carregadas antes de todas as outras extensões, por exemplo www/php56-opcache. Algumas podem precisar ser carregadas após uma extensão com prioridade de `30`. Nesse caso, adicione `PHP_MOD_PRIO=XX` no Makefile do port. Por exemplo:

```
USES=       php:ext
USE_PHP=    wddx
PHP_MOD_PRIO=   40
```

Estas variáveis estão disponíveis para uso em `PKGNAMEPREFIX` ou `PKGNAMESUFFIX`:

`PHP_PKGNAMEPREFIX`

Contém `phpXY-` onde *XY* é a versão do PHP atual. Use com módulos e extensões PHP.

`PHP_PKGNAMESUFFIX`

Contém `-phpXY` onde *XY* é a versão do PHP atual do flavor. Use com aplicativos PHP.

`PECL_PKGNAMEPREFIX`

Contém `phpXY-pecl` onde *XY* é a versão atual do PHP do flavor. Usar com módulos PECL.

Com flavors, todas as extensões PHP, extensões PECL, módulos PEAR *devem ter* um nome de pacote diferente, então todos devem usar uma dessas três variáveis em suas variáveis `PKGNAMEPREFIX` ou `PKGNAMESUFFIX`.

17.69. `pkgconfig`

Argumentos possíveis: (none), `build` (padrão), `run`, `both`

Utiliza devel/pkgconf. Sem argumentos ou com o argumento `build`, implica em `pkg-config` como

uma dependência de build-time. `run` implica em uma dependência em run-time e `both` implica em dependências de run-time e build-time.

17.70. `pure`

Argumentos possíveis: (none), `ffi`

Utiliza lang/pure. Usado largamente para build relacionado com ports pure. Com o argumento `ffi`, isso implica em devel/pure-ffi como uma dependência em run-time.

17.71. `pyqt`

Argumentos possíveis: (none), `4`, `5`

Utiliza PyQt. Se o port é parte do próprio PyQT, defina `PYQT_DIST`. Use a variável `USE_PYQT` para selecionar os componentes que o port precisa. Os componentes disponíveis são:

- `core`
- `dbus`
- `dbussupport`
- `demo`
- `designer`
- `designerplugin`
- `doc`
- `gui`
- `multimedia`
- `network`
- `opengl`
- `qscintilla2`
- `sip`
- `sql`
- `svg`
- `test`
- `webkit`
- `xml`
- `xmlpatterns`

Estes componentes só estão disponíveis com PyQT4:

- `assistant`
- `declarative`

- `help`
- `phonon`
- `script`
- `scripttools`

Estes componentes só estão disponíveis com PyQT5:

- `multimediawidgets`
- `printsupport`
- `qml`
- `serialport`
- `webkitwidgets`
- `widgets`

A dependência padrão para cada componente são build e run-time, para selecionar apenas build ou run, adicione `_build` ou `_run` para o nome do componente. Por exemplo:

```
USES=       pyqt
USE_PYQT=   core doc_build designer_run
```

17.72. `python`

Argumentos possíveis: (none), `XY`, `X.Y+`, `-XY`, `XY-ZA`, `patch`, `build`, `run`, `test`

Utiliza Python. Uma versão suportada ou um intervalo de versões podem ser especificados. Se o Python for necessário apenas no momento de build, run-time ou para os testes, ele pode ser definido como uma dependência de build, run ou teste com `build`, `run` ou `test`. Se o Python também for necessário durante a fase de patch, use `patch`. Veja Usando Python para maiores informações.

`PYTHON_NO_DEPENDS=yes` pode ser usado quando as variáveis exportadas pelo framework serem necessárias, mas uma dependência de Python não. Pode acontecer quando usado com `USES=shebangfix`, e o objetivo é apenas consertar os shebangs, mas não adicionar uma dependência do Python.

17.73. `qmail`

Argumentos possíveis: (none), `build`, `run`, `both`, `vars`

Utiliza mail/qmail. Com o argumento `build`, implica no `qmail` como uma dependência de build-time. `run` implica em uma dependência de run-time. Usando nenhum argumento ou o argumento `both` implica em dependências de run-time e build-time. `vars` só ira definir variáveis QMAIL para o port usar.

17.74. qmake

Argumentos possíveis: (none), norecursive, outsource, no_env, no_configure

Utiliza QMake para configuração. Para mais informações, veja Usando qmake.

17.75. qt

Argumentos possíveis: 5, no_env

Adiciona dependência de componentes Qt. no_env é passado diretamente para USES= qmake. Veja Usando o Qt para maiores informações.

17.76. qt-dist

Argumentos possíveis: (none) ou 5 e (none) ou um de 3d, activeqt, androidextras, base, canvas3d, charts, connectivity, datavis3d, declarative, doc, gamepad, graphicaleffects, imageformats, location, macextras, multimedia, networkauth, purchasing, quickcontrols2, quickcontrols, remoteobjects, script, scxml, sensors, serialbus, serialport, speech, svg, tools, translations, virtualkeyboard, wayland, webchannel, webengine, websockets, webview, winextras, x11extras, xmlpatterns

Fornece suporte para a compilação de componentes Qt 5. Ele cuida da definição do ambiente de configuração apropriado para o port compilar.

Exemplo 119. Compilando Componentes do Qt 5

O port é o componente networkauth do Qt 5, que faz parte do arquivo de distribuição networkauth.

```
PORTNAME=   networkauth
DISTVERSION=    ${QT5_VERSION}

USES=       qt-dist:5
```

Se PORTNAME não corresponder ao nome do componente, ele poderá ser passado como argumento em qt-dist.

Exemplo 120. Compilando Componentes do Qt 5 com Nomes Diferentes

O port é o componente gui do Qt 5, que faz parte do arquivo de distribuição base.

```
PORTNAME=   gui
DISTVERSION=    ${QT5_VERSION}

USES=       qt-dist:5,base
```

17.77. readline

Argumentos possíveis: (none), port

Usa readline como uma dependência de biblioteca e define CPPFLAGS e LDFLAGS como necessários. Se o argumento port é usado ou se readline não estiver presente no sistema base, adiciona uma dependência em devel/readline

17.78. samba

Possíveis argumentos: build, env, lib, run

Manipula dependências do Samba. env não irá adicionar qualquer dependência e apenas irá configurar as variáveis. build e run irão adicionar dependências de run-time e build-time de smbd. lib irá adicionar uma dependência em libsmbclient.so. As variáveis que são exportadas são:

SAMBAPORT
: A origem do port padrão Samba.

SAMBAINCLUDES
: A localização dos arquivos header do Samba.

SAMBALIBS
: O diretório onde as bibliotecas compartilhadas do Samba estão disponíveis.

17.79. scons

Argumentos possíveis: (none)

Fornece suporte para o uso do devel/scons. Veja Usando scons para maiores informações.

17.80. shared-mime-info

Argumentos possíveis: (none)

Utiliza update-mime-database a partir de misc/shared-mime-info. Este uses irão adicionar automaticamente uma etapa de post-install de tal forma que o próprio port ainda possa especificar sua própria etapa de post-install, se necessário. Também adiciona uma entrada @shared-mime-info para o plist.

17.81. shebangfix

Argumentos possíveis: (none)

Muitos softwares usam locais incorretos para interpretadores de scripts, principalmente /usr/bin/perl e /bin/bash. A macro shebangfix corrige linhas shebang em scripts listados em SHEBANG_REGEX, SHEBANG_GLOB ou SHEBANG_FILES.

`SHEBANG_REGEX`

Contém *uma* expressão regular estendida e é usado com o argumento `-iregex` do find(1). Veja `USES=shebangfix` com a variável `SHEBANG_REGEX`.

`SHEBANG_GLOB`

Contém uma lista de padrões usados com o argumento `-name` do find(1). Veja `USES=shebangfix` com a variável `SHEBANG_GLOB`.

`SHEBANG_FILES`

Contém uma lista de arquivos ou globs sh(1). A macro shebangfix é executada a partir de `${WRKSRC}`, assim `SHEBANG_FILES` pode conter caminhos relativos a `${WRKSRC}`. Ele também pode lidar com caminhos absolutos se arquivos fora de `${WRKSRC}` requisitarem uma correção. Veja `USES=shebangfix` com a variável `SHEBANG_FILES`.

Atualmente Bash, Java, Ksh, Lua, Perl, PHP, Python, Rubi, Tcl e Tk são suportados por padrão.

Aqui estão três variáveis de configuração:

`SHEBANG_LANG`

A lista de interpretadores suportados.

`interp_CMD`

O caminho para o interpretador de comandos no FreeBSD. O valor padrão é `${LOCALBASE}/bin/interp`.

`interp_OLD_CMD`

A lista de invocações erradas de interpretadores. Estes são tipicamente caminhos obsoletos, ou caminhos usados em outros sistemas operacionais que estão incorretos no FreeBSD. Eles serão substituídos pelo caminho correto na variável `interp__CMD`.

Estes vão *sempre* ser parte da variável `interp_OLD_CMD:"/usr/bin/envinterp" /bin/interp /usr/bin/interp /usr/local/bin/interp`.

A variável `interp_OLD_CMD` contém vários valores. Qualquer entrada com espaços deve estar entre aspas. Veja Especificando todos os Caminhos ao Adicionar um Interpretador para `USES=shebangfix`.

A correção de shebangs é feita durante a fase `patch`. Se os scripts forem criados com shebangs incorretos durante a fase `build`, o processo de build (por exemplo, o script configure, ou o Makefiles) deve ser corrigido ou ter o caminho certo (por exemplo, com `CONFIGURE_ENV`, `CONFIGURE_ARGS`, `MAKE_ENV` ou `MAKE_ARGS`) para gerar as shebangs certas.

Os caminhos corretos para os interpretadores suportados estão disponíveis em `interp_CMD`.

Quando usado com `USES=python`, e o objetivo é apenas consertar os shebangs, mas a

dependência de Python em si não é desejada, use a variável `PYTHON_NO_DEPENDS=yes`.

Exemplo 121. Adicionando outro interpretadoror para `USES=shebangfix`

Para adicionar outro interpretador, defina a variável `SHEBANG_LANG`. Por exemplo:

```
SHEBANG_LANG=   lua
```

Exemplo 122. Especificando todos os Caminhos ao Adicionar um Interpretador para `USES=shebangfix`

Se isto não estiver definido ainda, e não tiver valores padrão para `interp_OLD_CMD` e `interp_CMD` a entrada Ksh poderia ser definida como:

```
SHEBANG_LANG=   ksh
ksh_OLD_CMD=    "/usr/bin/env ksh" /bin/ksh /usr/bin/ksh
ksh_CMD=    ${LOCALBASE}/bin/ksh
```

Exemplo 123. Adicionando uma Localização Estranha para um Interpretador

Alguns softwares usam localizações estranhas para um interpretador. Por exemplo, um aplicativo pode esperar que Python esteja localizado em /opt/bin/python2.7. O caminho estranho a ser substituído pode ser declarado no Makefile do port:

```
python_OLD_CMD= /opt/bin/python2.7
```

Exemplo 124. `USES=shebangfix` *com a variável* `SHEBANG_REGEX`

Para corrigir todos os arquivos em `${WRKSRC}/scripts` finalizados com .pl, .sh ou .cgi faça assim:

```
USES=   shebangfix
SHEBANG_REGEX=  ./scripts/.*\.(sh|pl|cgi)
```

`SHEBANG_REGEX` é usada executando `find -E`, que usa expressões regulares modernas, também conhecidas como expressões regulares estendidas. Veja re_format(7) para maiores informações.

Exemplo 125. `USES=shebangfix` *com a variável* `SHEBANG_GLOB`

Para corrigir todos os arquivos em `${WRKSRC}` finalizados com .pl ou .sh, faça assim:

```
USES=   shebangfix
```

```
SHEBANG_GLOB=   *.sh *.pl
```

Exemplo 126. `USES=shebangfix` *com a variável* `SHEBANG_FILES`

Para corrigir os arquivos script/foobar.pl e script/*.sh dentro de `${WRKSRC}`, faça assim:

```
USES=   shebangfix
SHEBANG_FILES=  scripts/foobar.pl scripts/*.sh
```

17.82. `sqlite`

Argumentos possíveis: (none), `2`, `3`

Adiciona uma dependência SQLite. A versão padrão usada é 3, mas usar a versão 2 também é possível usando o modificador `:2`.

17.83. `ssl`

Argumentos possíveis: (none), `build`, `run`

Fornece suporte para OpenSSL. Uma dependência apenas de compilação ou run-time pode ser especificada usando `build` ou `run`. Estas variáveis estão disponíveis para uso do port, elas também são adicionadas para a variável `MAKE_ENV`:

`OPENSSLBASE`
: Caminho para a base de instalação do OpenSSL.

`OPENSSLDIR`
: Caminho para arquivos de configuração do OpenSSL.

`OPENSSLLIB`
: Caminho para as bibliotecas do OpenSSL.

`OPENSSLINC`
: Caminho para os includes do OpenSSL.

`OPENSSLRPATH`
: Se definido, o caminho que o vinculador precisa usar para localizar as bibliotecas do OpenSSL.

Se um port não for compilado com um flavor OpenSSL, defina a variável `BROKEN_SSL`, e possivelmente a variável `BROKEN_SSL_REASON_flavor`:

```
BROKEN_SSL= libressl
BROKEN_SSL_REASON_libressl= needs features only available in OpenSSL
```

17.84. tar

Argumentos possíveis: (none), Z, bz2, bzip2, lzma, tbz, tbz2, tgz, txz, xz

Define a variável EXTRACT_SUFX para .tar, .tar.Z, .tar.bz2, .tar.bz2, .tar.lzma, .tbz, .tbz2, .tgz, .txz ou .tar.xz respectivamente.

17.85. tcl

Argumentos possíveis: *version*, wrapper, build, run, tea

Adiciona uma dependência para o Tcl. Uma versão específica pode ser requisitada usando *version*. A versão pode estar vazia, um ou mais números exatos de versão (atualmente 84, 85 ou 86), ou um número mínimo de versão (atualmente 84+, 85+ ou 86+). Para solicitar apenas um wrapper sem uma versão especifica, use wrapper. Uma dependência somente de compilação ou run-time pode ser especificada usando build ou run. Para compilar o port usando Tcl Extension Architecture, use o tea. Depois de incluir bsd.port.pre.mk o port pode inspecionar os resultados usando estas variáveis:

- TCL_VER: seleciona a versão major.minor do Tcl
- TCLSH: caminho completo do interpretador do Tcl
- TCL_LIBDIR: caminho das bibliotecas do Tcl
- TCL_INCLUDEDIR: caminho dos arquivos de cabeçalho C do Tcl
- TK_VER: versão major.minor do Tk que foi escolhida
- WISH: caminho completo do interpretador do Tk
- TK_LIBDIR: caminho das bibliotecas do Tk
- TK_INCLUDEDIR: caminho dos arquivos de cabeçalho C do Tk

17.86. terminfo

Argumentos possíveis: (none)

Adiciona @terminfo ao arquivo plist. Use quando o port instalar arquivos *.terminfo em ${PREFIX}/shared/misc.

17.87. tk

Os mesmos argumentos para tcl

Um pequeno wrapper ao usar os dois Tcl e Tk. As mesmas variáveis são retornadas assim como quando estiver usando Tcl.

17.88. uidfix

Argumentos possíveis: (none)

Altera algum comportamento padrão (principalmente de variáveis) do sistema de compilação para permitir instalar este port como um usuário normal. Tente isso no port antes de usar USES=fakeroot ou de aplicar algum patch.

17.89. uniquefiles

Argumentos possíveis: (none), dirs

Torna arquivos ou diretórios 'exclusivos', adicionando um prefixo ou sufixo. Se o argumento dirs é usado, o port precisa de um prefixo (e apenas um prefixo) baseado em UNIQUE_PREFIX para diretórios padrão DOCSDIR, EXEMPLESDIR, DATADIR, WWWDIR, ETCDIR. Estas variáveis estão disponíveis para os ports:

- UNIQUE_PREFIX: O prefixo a ser usado para diretórios e arquivos. Padrão: ${PKGNAMEPREFIX}.
- UNIQUE_PREFIX_FILES: Uma lista de arquivos que precisam ser prefixados. Padrão: vazio.
- UNIQUE_SUFFIX: O sufixo para ser usado por arquivos. Padrão: ${PKGNAMESUFFIX}.
- UNIQUE_SUFFIX_FILES: Uma lista de arquivos que precisam estar com um sufixo. Padrão: vazio.

17.90. varnish

Argumentos possíveis: 4, 6

Manipula dependências do Varnish Cache. 4 irá adicionar uma dependência do www/varnish4. 6 irá adicionar uma dependência do www/varnish6.

17.91. webplugin

Argumentos possíveis: (none), ARGS

Cria e remove automaticamente links simbólicos para cada aplicação que suporta o framework do webplugin. ARGS pode ser um dos:

- gecko: suporte a plug-ins baseados no Gecko
- native: suporte a plug-ins para o Gecko, Opera e WebKit-GTK
- linux: suporte a plug-ins do Linux
- all (padrão, implícito): suporta todos os tipos de plug-ins
- (entradas individuais): suporta apenas os navegadores listados

Essas variáveis podem ser ajustadas:

- WEBPLUGIN_FILES: Sem padrão, deve ser definido manualmente. Os arquivos de plug-in para instalar.
- WEBPLUGIN_DIR: O diretório para instalar os arquivos de plug-in, padrão PREFIX/lib/browser_plugins/WEBPLUGIN_NAME. Defina isso se o port instalar arquivos de plug-in fora do diretório padrão para previnir links simbólicos quebrados.

- `WEBPLUGIN_NAME`: O diretório final para instalar os arquivos de plug-in, padrão `PKGBASE`.

17.92. `xfce`

Argumentos possíveis: (none), `gtk2`

Fornece suporte para ports relacionados ao Xfce. Veja Usando o Xfce para detalhes.

O argumento `gtk2` especifica que o port requer suporte a GTK2. Ele adiciona recursos adicionais fornecidos por alguns componentes principais, por exemplo, x11/libxfce4menu e x11-wm/xfce4-panel.

17.93. `xorg`

Argumentos possíveis: (none)

Fornece uma maneira fácil para depender dos componentes X.org. Os componentes devem ser listados na variável `USE_XORG`. Os componentes disponíveis são:

Tabela 52. Componentes Disponíveis do X.Org

Nome	Descrição
`dmx`	Biblioteca de extensão DMX
`fontenc`	Biblioteca fontenc
`fontutil`	Crie um índice de arquivos de fontes X em um diretório
`ice`	Biblioteca Inter Client Exchange para X11
`libfs`	Biblioteca FS
`pciaccess`	Biblioteca Genérica de acesso ao PCI
`pixman`	Biblioteca de manipulação de pixels de baixo nível
`sm`	Biblioteca de Gerenciamento de Sessão para X11
`x11`	Biblioteca X11
`xau`	Biblioteca do Protocolo de Autenticação para o X11
`xaw`	Biblioteca de Widgets do X Athena
`xaw6`	Biblioteca de Widgets do X Athena
`xaw7`	Biblioteca de Widgets do X Athena
`xbitmaps`	Arquivos bitmaps do X.Org
`xcb`	Biblioteca do protocolo X C-language Binding (XCB)
`xcomposite`	Biblioteca de extensão X Composite

Nome	Descrição
`xcursor`	Biblioteca de carregamento do cursor X no lado do cliente
`xdamage`	Biblioteca de extensão X Damage
`xdmcp`	Biblioteca do Protocolo de Controle do X Display Manager
`xext`	Biblioteca de Extensão X11
`xfixes`	Biblioteca de extensão X Fixes
`xfont`	Biblioteca de fontes do X
`xfont2`	Biblioteca de fontes do X
`xft`	API de fontes do lado do cliente para aplicativos X
`xi`	Biblioteca de extensão X Input
`xinerama`	Biblioteca X11 Xinerama
`xkbfile`	Biblioteca XKB
`xmu`	Biblioteca de Utilitários Diversos do X
`xmuu`	Biblioteca de Utilitários Diversos do X
`xorg-macros`	Macros aclocal de desenvolvimento X.Org
`xorg-server`	Servidor X do X.Org e programas relacionados
`xorgproto`	xorg protocol headers
`xpm`	Biblioteca Pixmap do X
`xpresent`	Biblioteca de Extensão X Present
`xrandr`	Biblioteca de extensão X Resize e Rotate
`xrender`	Biblioteca de extensão X Render
`xres`	Biblioteca de uso X Resource
`xscrnsaver`	Biblioteca XScrnSaver
`xshmfence`	Memória compartilhada 'SyncFence' primitiva de sincronização
`xt`	Biblioteca X Toolkit
`xtrans`	Código de rede abstrato para X
`xtst`	Extensão X Test
`xv`	Biblioteca de Extensão X Video
`xvmc`	Biblioteca X Video Extension Motion Compensation
`xxf86dga`	X DGA Extension
`xxf86vm`	Extensão X Vidmode

17.94. `xorg-cat`

Argumentos possíveis: `app`, `data`, `doc`, `driver`, `font`, `lib`, `proto`, `util`, `xserver` e (none) ou um de `autotools` (default), `meson`

Forneça suporte para compilação de componentes Xorg. Ele cuida da definição de dependências comuns e de um ambiente de configuração apropriado necessário. Isso é destinado apenas aos componentes do Xorg.

A categoria deve corresponder às categorias upstream.

O segundo argumento é o sistema de compilação a ser usado. autotools é o padrão, mas meson também é suportado.

17.95. `zip`

Argumentos possíveis: (none), `infozip`

Indica que os arquivos de distribuição usam o algoritmo de compactação ZIP. Para arquivos que usam o algoritmo InfoZip, o argumento `infozip` deve ser passado para definir as dependências apropriadas.

Capítulo 18. Valores `__FreeBSD_version`

Aqui está uma lista conveniente dos valores `__FreeBSD_version` definidos em sys/param.h:

18.1. Versões do FreeBSD 13

Tabela 53. Valores do `__FreeBSD_version` para o FreeBSD 13

Valor	Revisão	Data	Release
1300000	r339436	19 de outubro de 2018	13.0-CURRENT.
1300001	r339730	25 de outubro de 2018	13.0-CURRENT after bumping OpenSSL shared library version numbers.
1300002	r339765	25 de outubro de 2018	13.0-CURRENT after restoration of sys/joystick.h.
1300003	r340055	2 de novembro de 2018	13.0-CURRENT after vop_symlink API change (`a_target` is now `const`.)
1300004	r340841	23 de novembro de 2018	13.0-CURRENT depois de habilitar o código crtbegin e crtend.
1300005	r341836	11 de dezembro de 2018	13.0-CURRENT depois de habilitar checksums para inodes do UFS.
1300006	r342398	24 de dezembro de 2018	13.0-CURRENT depois de consertar o include `sys/random.h` para ser utilizável em C ++.
1300007	r342629	30 de dezembro de 2018	13.0-CURRENT depois de mudar o tamanho do `struct linux_cdev` nas plataformas de 32-bits.
1300008	r342772	4 de janeiro de 2019	13.0-CURRENT depois de adicionar os sysctls `kern.smp.threads_per_core` e `kern.smp.cores`.

Valor	Revisão	Data	Release
1300009	r343213	20 de janeiro de 2019	13.0-CURRENT após a modificação da estrutura `struct ieee80211vap` para resolver a corrida ioctl/detach para a estrutura ieee80211com.
1300010	r343485	27 de janeiro de 2019	13.0-CURRENT depois de incrementar o `SPECNAMELEN` de 63 para MAXNAMELEN (255).
1300011	r344041	12 de fevereiro de 2019	13.0-CURRENT depois que o renameat(2) foi corrigido para funcionar com kernels construídos com a opção `CAPABILITIES`.
1300012	r344062	12 de fevereiro de 2019	13.0-CURRENT após `taskqgroup_attach()` e `taskqgroup_attach_cpu()` tomar um device_t e um ponteiro de recurso struct como argumentos para denotar interrupções de dispositivo.
1300013	r344300	19 de fevereiro de 2019	13.0-CURRENT após a remoção do drm e do drm2.
1300014	r344779	4 de março de 2019	13.0-CURRENT depois de atualizar o clang, llvm, lld, lldb, compiler-rt e libc++ para a 8.0.0 rc3.
1300015	r345196	15 de março de 2019	13.0-CURRENT depois de desanonimizar as threads e os proc state enums, de forma que as aplicações userland podem usá-las sem redefinir os nomes dos valores.

Valor	Revisão	Data	Release
1300016	r345236	16 de março de 2019	13.0-CURRENT depois de habilitar o código crtbegin e crtend.
1300017	r345305	19 de março de 2019	13.0-CURRENT after exposing the Rx mbuf buffer size to drivers in iflib.
1300018	r346012	16 de março de 2019	13.0-CURRENT after introduction of funlinkat syscall in r345982.
1300019	r346282	16 de abril de 2019	13.0-CURRENT after addition of is_random_seeded(9) to random(4).
1300020	r346358	18 de abril de 2019	13.0-CURRENT after restoring random(4) availability tradeoff prior to r346250 and adding new tunables and diagnostic sysctls for programmatically discovering early seeding problems after boot.
1300021	r346645	24 de abril de 2019	13.0-CURRENT after LinuxKPI uses bus_dma(9) to be compatible with an IOMMU.
1300022	r347089	4 de maio de 2019	13.0-CURRENT after fixing regression issue after rr346645 in the LinuxKPI.
1300023	r347192	6 de maio de 2019	13.0-CURRENT after list-ifying kernel dump device configuration.
1300024	r347325	8 de maio de 2019	13.0-CURRENT after bumping the Mellanox driver version numbers (mlx4en(4); mlx5en(4)).

Valor	Revisão	Data	Release
1300025	r347532	13 de maio de 2019	13.0-CURRENT after renaming `vm.max_wired` to `vm.max_user_wired` and changing its type.
1300026	r347596	14 de maio de 2019	13.0-CURRENT after adding context member to ww_mutex in LinuxKPI.
1300027	r347601	14 de maio de 2019	13.0-CURRENT after adding prepare to pm_ops in LinuxKPI.
1300028	r347925	17 de maio de 2019	13.0-CURRENT after removal of bm, cs, de, ed, ep, ex, fe, pcn, sf, sn, tl, tx, txp, vx, wb, and xe drivers.
1300029	r347984	20 de maio de 2019	13.0-CURRENT after removing some header pollution due to `sys/eventhandler.h`. Affected files may now need to explicitly include one or more of `sys/eventhandler.h`, `sys/ktr.h`, `sys/lock.h`, or `sys/mutex.h`, when the missing header may have been included implicitly prior to 1300029.
1300030	r348350	29 de maio de 2019	13.0-CURRENT after adding relocation support to libdwarf on powerpc64 to fix handling of DWARF information on unlinked objects. Original commit in r348347.

Valor	Revisão	Data	Release
1300031	r348808	8 de junho de 2019	13.0-CURRENT after adding dpcpu and vnet section fixes to i386 kernel modules to avoid panics in certain conditions. i386 kernel modules need to be recompiled with the linker script magic in place or they will refuse to load.
1300032	r349151	17 de junho de 2019	13.0-CURRENT after separating kernel crc32() implementation to its own header (gsb_crc32.h) and renaming the source to gsb_crc32.c.
1300033	r349277	21 de junho de 2019	13.0-CURRENT after additions to LinuxKPI's `rcu` list.
1300034	r349352	24 de junho de 2019	13.0-CURRENT after NAND and NANDFS removal.
1300035	r349846	8 de julho de 2019	13.0-CURRENT after merging the vm_page hold and wire mechanisms.
1300036	r349972	13 de julho de 2019	13.0-CURRENT after adding arm_drain_writebuf() and arm_sync_icache() for compatibility with NetBSD and OpenBSD.
1300037	r350307	24 de julho de 2019	13.0-CURRENT after removal of libcap_random(3).
1300038	r350437	30 de julho de 2019	13.0-CURRENT after removal of gzip'ed a.out support.
1300039	r350665	7 de agosto de 2019	13.0-CURRENT after merge of fusefs from projects/fuse2.

Valor	Revisão	Data	Release
1300040	r351140	16 de agosto de 2019	13.0-CURRENT after deletion of sys/dir.h which has been deprecated since 1997.
(Não mudou)	r351423	23 de agosto de 2019	13.0-CURRENT after changing most arguments to ping6(8).
1300041	r351480	25 de agosto de 2019	13.0-CURRENT after removal of zlib 1.0.4 after the completion of kernel zlib unification.
1300042	r351522	27 de agosto de 2019	13.0-CURRENT after addition of kernel-side support for in-kernel TLS.
1300043	r351698	2 de setembro de 2019	13.0-CURRENT after removal of gets(3).
1300044	r351701	2 de setembro de 2019	13.0-CURRENT after adding sysfs create/remove functions that handles multiple files in one call to the LinuxKPI.
1300045	r351729	3 de setembro de 2019	13.0-CURRENT after adding sysctlbyname system call
1300046	r351937	6 de setembro de 2019	13.0-CURRENT after LinuxKPI sysfs improvements.
1300047	r352110	9 de setembro de 2019	13.0-CURRENT after changing the synchonization rules for vm_page reference counting..
1300048	r352700	25 de setembro de 2019	13.0-CURRENT after adding a shm_open2 syscall to support the upcoming memfd_create syscall.

Valor	Revisão	Data	Release
1300049	r353274	7 de outubro de 2019	13.0-CURRENT after factoring out the VNET shutdown check into an own vnet structure field.
1300050	r353358	9 de outubro de 2019	13.0-CURRENT after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 9.0.0 final release r372316.
1300051	r353685	17 de outubro de 2019	13.0-CURRENT after splitting out a more generic debugnet(4) from netdump(4).
1300052	r353698	17 de outubro de 2019	13.0-CURRENT after promoting the page busy field to a first class lock that no longer requires the object lock for consistency.
1300053	r353700	17 de outubro de 2019	13.0-CURRENT after implementing NetGDB.
1300054	r353868	21 de outubro de 2019	13.0-CURRENT after removing obsoleted KPIs that were used to access interface address lists.
1300055	r354335	4 de novembro de 2019	13.0-CURRENT after enabling device class group attributes in the LinuxKPI.
1300056	r354460	7 de novembro de 2019	13.0-CURRENT after fixing a potential OOB read security issue in libc++.
1300057	r354694	13 de novembro de 2019	13.0-CURRENT after adding support for AT_EXECPATH to elf_aux_info(3).

Valor	Revisão	Data	Release
1300058	r354820	18 de novembro de 2019	13.0-CURRENT after widening the vm_page aflags field to 16 bits.
1300059	r354835	18 de novembro de 2019	13.0-CURRENT after converting the in-tree sysent targets to use the new makesyscalls.lua.
1300060	r354922	20 de novembro de 2019	13.0-CURRENT after adding /etc/os-release as a symbolic link to /var/run/os-release.
1300061	r354977	21 de novembro de 2019	13.0-CURRENT after adding functions to bitstring(3) to find contiguous sequences of set or unset bits.
1300062	r355309	2 de dezembro de 2019	13.0-CURRENT after adding TCP_STATS support.
1300063	r355537	8 de dezembro de 2019	13.0-CURRENT after removal of VI_DOOMED (use VN_IS_DOOMED instead).
1300064	r355658	9 de dezembro de 2019	13.0-CURRENT after correcting the C++ version check for declaring timespec_get(3).
1300065	r355643	12 de dezembro de 2019	13.0-CURRENT after adding sigsetop extensions commonly found in musl libc and glibc.
1300066	r355679	12 de dezembro de 2019	13.0-CURRENT after changing the internal interface between the NFS modules as part of the introduction of NFS 4.2.

Valor	Revisão	Data	Release
1300067	r355732	13 de dezembro de 2019	13.0-CURRENT after removing the deprecated `callout_handle_init`, `timeout`, and `untimeout` functions.
1300068	r355828	16 de dezembro de 2019	13.0-CURRENT after doubling the value of `ARG_MAX`, for 64 bit platforms.
1300069	r356051	24 de dezembro de 2019	13.0-CURRENT after the addition of busdma templates.
1300070	r356113	27 de dezembro de 2019	13.0-CURRENT after eliminating the last MI difference in AT_* definitions (for powerpc).
1300071	r356135	27 de dezembro de 2019	13.0-CURRENT after making USB statistics be per-device instead of per bus.
1300072	r356185	29 de dezembro de 2019	13.0-CURRENT after removal of GEOM_SCHED class and gsched tool.
1300073	r356263	2 de janeiro de 2020	13.0-CURRENT after removing arm/arm as a valid target.
1300074	r356337	3 de janeiro de 2020	13.0-CURRENT after removing flags argument from VOP_UNLOCK.
1300075	r356409	6 de janeiro de 2020	13.0-CURRENT after adding own counter for cancelled USB transfers.
1300076	r356511	8 de janeiro de 2020	13.0-CURRENT after pushing vnop implementation into the fileop layer in posix_fallocate.

Valor	Revisão	Data	Release
(Não mudou)	r357396	2 de fevereiro de 2020	13.0-CURRENT after removal of armv5 architecture code from the src tree.
1300077	r357455	3 de fevereiro de 2020	13.0-CURRENT after removal of sparc64 architecture code from the src tree.
1300078	r358020	17 de fevereiro de 2020	13.0-CURRENT after changing `struct vnet` and the VNET magic cookie.
1300079	r358164	20 de fevereiro de 2020	13.0-CURRENT after upgrading ncurses to 6.2.x
1300080	r358172	20 de fevereiro de 2020	13.0-CURRENT after adding realpathat syscall to VFS.
1300081	r358218	21 de fevereiro de 2020	13.0-CURRENT after after recent linuxkpi changes.
1300082	r358497	1 de março de 2020	13.0-CURRENT after removal of bktr(4).
1300083	r358834	10 de março de 2020	13.0-CURRENT after removal of amd(8), r358821.
1300084	r358851	10 de março de 2020	13.0-CURRENT after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 10.0.0-rc3 c290cb61fdc.
1300085	r359261	23 de março de 2020	13.0-CURRENT after the import of the kyua test framework.
1300086	r359347	26 de março de 2020	13.0-CURRENT after switching powerpc and powerpcspe to the lld linker.

Valor	Revisão	Data	Release
1300087	r359374	27 de março de 2020	13.0-CURRENT after refactoring the driver and consumer interfaces for in-kernel cryptography.
1300088	r359530	1 de abril de 2020	13.0-CURRENT after removing support for procfs process debugging.
1300089	r359727	8 de abril de 2020	13.0-CURRENT after cloning the RCU interface into a sleepable and a non-sleepable part in the LinuxKPI.
1300090	r359747	9 de abril de 2020	13.0-CURRENT after removing the old NFS lock device driver that uses Giant.
1300091	r359839	12 de abril de 2020	13.0-CURRENT after implementing a close_range(2) syscall.
1300092	r359920	14 de abril de 2020	13.0-CURRENT after reworking unmapped mbufs in KTLS to carry ext_pgs in the mbuf itself.
1300093	r360418	27 de abril de 2020	13.0-CURRENT after adding support for kernel TLS receive offload.
1300094	r360796	7 de maio de 2020	13.0-CURRENT after linuxkpi changes.
1300095	r361275	20 de maio de 2020	13.0-CURRENT after adding HyperV socket support for FreeBSD guests.

Valor	Revisão	Data	Release
1300096	r361410	23 de maio de 2020	13.0-CURRENT after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 10.0.1 rc1 f79cd71e145.
1300097	r361724	2 de junho de 2020	13.0-CURRENT after implementing __is_constexpr() function macro in the LinuxKPI.
1300098	r362159	14 de junho de 2020	13.0-CURRENT after changing the `export_args ex_flags` field so that is 64bits.
1300099	r362453	20 de junho de 2020	13.0-CURRENT after making liblzma use libmd implementation of SHA256.
1300100	r362640	26 de junho de 2020	13.0-CURRENT after changing the internal API between the NFS kernel modules.
1300101	r363077	10 de julho de 2020	13.0-CURRENT after implementing the array_size() function in the LinuxKPI.
1300102	r363562	26 de julho de 2020	13.0-CURRENT after implementing lockless lookup in the VFS layer.
1300103	r363757	1 de agosto de 2020	13.0-CURRENT after making rights mandatory for NDINIT_ALL.
1300104	r363783	2 de agosto de 2020	13.0-CURRENT after vnode layout changes.
1300105	r363894	5 de agosto de 2020	13.0-CURRENT after vaccess() change.

Valor	Revisão	Data	Release
1300106	r364092	11 de agosto de 2020	13.0-CURRENT after adding an argument to newnfs_connect() that indicates use TLS for the connection.
1300107	r364109	11 de agosto de 2020	13.0-CURRENT after change to clone the task struct fields related to RCU.
1300108	r364233	14 de agosto de 2020	13.0-CURRENT after adding a few wait_bit functions to the linuxkpi, which are needed for DRM from Linux v5.4.
1300109	r364274	6 de agosto de 2020	13.0-CURRENT after vget() argument removal and namei flags renumbering.
(Não mudou)	r364284	6 de agosto de 2020	13.0-CURRENT after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to release/11.x llvmorg-11.0.0-rc1-47-gff47911ddfc.
1300110	r364331	18 de agosto de 2020	13.0-CURRENT after deleting the unused `use_ext` argument to `nfscl_reqstart()`.
1300111	r364476	22 de agosto de 2020	13.0-CURRENT after adding TLS support to the kernel RPC.
1300112	r364747	25 de agosto de 2020	13.0-CURRENT after merging OpenZFS support.
1300113	r364753	25 de agosto de 2020	13.0-CURRENT after adding atomic and bswap functions to libcompiler_rt.

Valor	Revisão	Data	Release
1300114	r365459	8 de setembro de 2020	13.0-CURRENT after changing arm64 AT_HWCAP definitions for elf_aux_info(3).
1300115	r365705	14 de setembro de 2020	13.0-CURRENT after fixing crunchgen(1) application build with `WARNS=6`.
1300116	r366062	22 de setembro de 2020	13.0-CURRENT after the introduction of the powerpc64le ARCH.
1300117	r366070	23 de setembro de 2020	13.0-CURRENT after reimplementing purgevfs to iterate vnodes instead of the entire hash.
1300118	r366374	2 de outubro de 2020	13.0-CURRENT after adding backlight support and `dmi_*` functions to the linuxkpi.
1300119	r366432	6 de outubro de 2020	13.0-CURRENT after populating the acquire context field of a `ww_mutex` in the LinuxKPI.
1300120	r366666	13 de outubro de 2020	13.0-CURRENT after the fix to arm64 write-only mappings.
1300121	r366719	15 de outubro de 2020	13.0-CURRENT after the addition of `VOP_EAGAIN`.
1300122	r366782	17 de outubro de 2020	13.0-CURRENT after the addition of `ptsname_r`.
1300123	r366871	20 de outubro de 2020	13.0-CURRENT after `VOP`, `VPTOCNP`, and `INACTIVE` changes.
1300124	r367162	30 de outubro de 2020	13.0-CURRENT after adding `cache_vop_mkdir` and renaming `cache_rename` to `cache_vop_rename`.

Valor	Revisão	Data	Release
1300125	r367347	4 de novembro de 2020	13.0-CURRENT after using a `rms` lock for teardown handling in `zfs`.
1300126	r367384	5 de novembro de 2020	13.0-CURRENT after rationalizing per-cpu zones.
1300127	r367432	6 de novembro de 2020	13.0-CURRENT after moving `malloc_type_internal` into `malloc_type`.
1300128	r367522	9 de novembro de 2020	13.0-CURRENT after LinuxKPI additions to implement ACPI bits required by `drm-kmod` in the base system.
1300129	r367627	12 de novembro de 2020	13.0-CURRENT after retiring `malloc_last_fail`.
1300130	r367777	November 17, 2020	13.0-CURRENT after p_pd / pwddesc split from p_fd / filedesc.

18.2. Versões do FreeBSD 12

Tabela 54. Valores do `__FreeBSD_version` para o FreeBSD 12

Valor	Revisão	Data	Release
1200000	r302409	7 de julho de 2016	12.0-CURRENT.
1200001	r302628	12 de julho de 2016	12.0-CURRENT after removing collation from `[a-z]`-type ranges.
1200002	r304395	18 de agosto de 2016	12.0-CURRENT after removing unused and obsolete `openbsd_poll` system call.
1200003	r304608	22 de agosto de 2016	12.0-CURRENT after adding C++11 `thread_local` support in rev r303795.

Valor	Revisão	Data	Release
1200004	r304752	24 de agosto de 2016	12.0-CURRENT after fixing LC_*_MASK for newlocale(3) and querylocale(3) (rev r304703).
1200005	r304789	25 de agosto de 2016	12.0-CURRENT after changing some ioctl interfaces in rev r304787 between the iSCSI userspace programs and the kernel.
1200006	r305256	1º de setembro de 2016	12.0-CURRENT after crunchgen(1) META_MODE fix in r305254.
1200007	r305421	5 de setembro de 2016	12.0-CURRENT after resolving a deadlock between `device_detach()` and usbd_do_request_flags(9).
1200008	r305833	15 de setembro de 2016	12.0-CURRENT after removing the 4.3BSD compatible macro `m_copy()` in r305824.
1200009	r306077	21 de setembro de 2016	12.0-CURRENT after removing `bio_taskqueue()` in r305988.
1200010	r306276	23 de setembro de 2016	12.0-CURRENT after mounting msdosfs(5) with longnames support by default.
1200011	r306556	1 de outubro de 2016	12.0-CURRENT after adding `fb_memattr` field to `fb_info` in r306555.
1200012	r306592	2 de outubro de 2016	12.0-CURRENT after net80211(4) changes (rev r306590, r306591).

Valor	Revisão	Data	Release
1200013	r307140	12 de outubro de 2016	12.0-CURRENT after installing header files required development with libzfs_core.
1200014	r307529	17 de outubro de 2016	12.0-CURRENT after merging common code in rtwn(4) and urtwn(4), and adding support for 802.11ac devices.
1200015	r308874	20 de novembro de 2016	12.0-CURRENT after some ABI change for unbreaking powerpc.
1200016	r309017	22 de novembro de 2016	12.0-CURRENT after removing `PG_CACHED` -related fields from `vmmeter`.
1200017	r309124	25 de novembro de 2016	12.0-CURRENT after upgrading our copies of clang, llvm, lldb, compiler-rt and libc++ to 3.9.0 release, and adding lld 3.9.0.
1200018	r309676	7 de dezembro de 2016	12.0-CURRENT after adding the `ki_moretdname` member to `struct kinfo_proc` and `struct kinfo_proc32` to export the whole thread name to user-space utilities.
1200019	r310149	16 de dezembro de 2016	12.0-CURRENT after starting to lay down the foundation for 11ac support.
1200020	r312087	13 de janeiro de 2017	12.0-CURRENT after removing `fgetsock` and `fputsock`.
1200021	r313858	16 de fevereiro de 2017	12.0-CURRENT after removing MCA and EISA support.

Valor	Revisão	Data	Release
1200022	r314040	21 de fevereiro de 2017	12.0-CURRENT after making the LinuxKPI task struct persistent across system calls.
(Não mudou)	r314373	2 de março de 2017	12.0-CURRENT after removing System V Release 4 binary compatibility support.
1200023	r314564	2 de março de 2017	12.0-CURRENT after upgrading our copies of clang, llvm, lld, lldb, compiler-rt and libc++ to 4.0.0.
1200024	r314865	7 de março de 2017	12.0-CURRENT after removal of pcap-int.h
1200025	r315430	16 de março de 2017	12.0-CURRENT after addition of the <dev/mmc/mmc_ioctl.h> header.
1200026	r315662	16 de março de 2017	12.0-CURRENT after hiding `struct inpcb` and `struct tcpcb` from userland.
1200027	r315673	21 de março de 2017	12.0-CURRENT after making CAM SIM lock optional.
1200028	r316683	10 de abril de 2017	12.0-CURRENT after renaming `smp_no_rendevous_barrier()` to `smp_no_rendezvous_barrier()` in r316648.
1200029	r317176	19 de abril de 2017	12.0-CURRENT after the removal of `struct vmmeter` from `struct pcpu` from r317061.
1200030	r317383	24 de abril de 2017	12.0-CURRENT after removing NATM support including en(4), fatm(4), hatm(4), and patm(4).

Valor	Revisão	Data	Release
1200031	r318736	23 de maio de 2017	12.0-CURRENT after types `ino_t`, `dev_t`, `nlink_t` were extended to 64bit and `struct dirent` changed layout (also known as ino64).
1200032	r319664	8 de junho de 2017	12.0-CURRENT after removal of `groff`.
1200033	r320043	17 de junho de 2017	12.0-CURRENT after the type of the `struct event` member `data` was increased to 64bit, and ext structure members added.
1200034	r320085	19 de junho de 2017	12.0-CURRENT after the NFS client and server were changed so that they actually use the 64bit `ino_t`.
1200035	r320317	24 de junho de 2017	12.0-CURRENT after the `MAP_GUARD` mmap(2) flag was added.
1200036	r320347	26 de junho de 2017	12.0-CURRENT after changing `time_t` to 64 bits on powerpc (32-bit version).
1200037	r320545	1º de julho de 2017	12.0-CURRENT after the cleanup and inlining of `bus_dmamap*` functions (r320528).
1200038	r320879	10 de julho de 2017	12.0-CURRENT after MMC CAM committed. (r320844).
1200039	r321369	22 de julho de 2017	12.0-CURRENT after upgrade of copies of clang, llvm, lld, lldb, compiler-rt and libc++ to 5.0.0 (trunk r308421).
1200040	r321688	29 de julho de 2017	12.0-CURRENT after adding NFS client forced dismount support `umount -N`.

Valor	Revisão	Data	Release
1200041	r322762	21 de agosto de 2017	12.0-CURRENT after WRFSBASE instruction become operational on amd64.
1200042	r322900	25 de agosto de 2017	12.0-CURRENT after PLPMTUD counters were changed to use counter(9).
1200043	r322989	28 de agosto de 2017	12.0-CURRENT after dropping x86 CACHE_LINE_SIZE down to 64 bytes.
1200044	r323349	8 de setembro de 2017	12.0-CURRENT after implementing poll_wait() in the LinuxKPI.
1200045	r323706	18 de setembro de 2017	12.0-CURRENT after adding shared memory support to LinuxKPI. (r323703).
1200046	r323910	22 de setembro de 2017	12.0-CURRENT after adding support for 32-bit compatibility IOCTLs to LinuxKPI.
1200047	r324053	26 de setembro de 2017	12.0-CURRENT after removing M_HASHTYPE_RSS_UDP_IPV4_EX. (r324052).
1200048	r324227	2 de outubro de 2017	12.0-CURRENT after hiding `struct socket` and `struct unpcb` from userland.
1200049	r324281	4 de outubro de 2017	12.0-CURRENT after adding the `value.u16` field to `struct diocgattr_arg`.
1200050	r324342	5 de outubro de 2017	12.0-CURRENT after adding the `armv7` `MACHINE_ARCH`. (r324340).

Valor	Revisão	Data	Release
1200051	r324455	9 de outubro de 2017	12.0-CURRENT after removing libstand.a as a public interface. (r324454).
1200052	r325028	26 de outubro de 2017	12.0-CURRENT after fixing `ptrace()` to always clear the correct thread event when resuming.
1200053	r325506	7 de novembro de 2017	12.0-CURRENT after changing `struct mbuf` layout to add optional hardware timestamps for receive packets.
1200054	r325852	15 de novembro de 2017	12.0-CURRENT after changing the layout of `struct vmtotal` to allow for reporting large memory counters.
1200055	r327740	9 de janeiro de 2018	12.0-CURRENT after adding `cpucontrol -e` support.
1200056	r327952	14 de janeiro de 2018	12.0-CURRENT after upgrading clang, llvm, lld, lldb, compiler-rt and libc++ to 6.0.0 (branches/release_60 r321788).
1200057	r329033	8 de fevereiro de 2018	12.0-CURRENT after applying a clang 6.0.0 fix to make the wine ports build correctly.
1200058	r329166	12 de fevereiro de 2018	12.0-CURRENT após o lua loader ser inserido.

Valor	Revisão	Data	Release
1200059	r330299	2 de março de 2018	12.0-CURRENT after removing the declaration of `union semun` unless `_WANT_SEMUN` is defined. Also the removal of `struct mymsg` and the renaming of kernel-only members of `struct semid_ds` and `struct msgid_ds`.
1200060	r330384	4 de março de 2018	12.0-CURRENT after upgrading clang, llvm, lld, lldb, compiler-rt and libc++ to 6.0.0 release.
1200061	r332100	6 de abril de 2018	12.0-CURRENT after changing syslog(3) to emit RFC 5424 formatted messages.
1200062	r332423	12 de abril de 2018	12.0-CURRENT after changing the Netmap API.
1200063	r333446	10 de maio de 2018	12.0-CURRENT after reworking CTL frontend and backend options to use nv(3), allow creating multiple ioctl frontend ports.
1200064	r334074	22 de maio de 2018	12.0-CURRENT after changing the ifnet address and multicast address TAILQ to CK_STAILQ.
1200065	r334290	28 de maio de 2018	12.0-CURRENT after changing dwatch(1) to allow '-E code' to override profile EVENT_DETAILS.
1200066	r334466	1 de junho de 2018	12.0-CURRENT after removal of in-kernel pmc tables for Intel.

Valor	Revisão	Data	Release
1200067	r334892	9 de junho de 2018	12.0-CURRENT after adding DW_LANG constants to libdwarf.
1200068	r334930	12 de junho de 2018	12.0-CURRENT after changing the interface between the NFS modules.
1200069	r335237	15 de junho de 2018	12.0-CURRENT after changing `struct kerneldumpheader` to version 4 (similar to version 2 in 11-STABLE and previous).
1200070	r335873	2 de julho de 2018	12.0-CURRENT after inlining atomic(9) in modules on amd64 and i386 requiring all modules of consumers to be rebuilt for these architectures.
1200071	r335930	4 de julho de 2018	12.0-CURRENT after changing the ABI and API of epoch(9) (r335924) requiring modules of consumers to be rebuilt.
1200072	r335979	5 de julho de 2018	12.0-CURRENT after changing the ABI and API of `struct xinpcb` and friends.
1200073	r336313	15 de julho de 2018	12.0-CURRENT after changing the ABI and API of `struct if_shared_ctx` and `struct if_softc_ctx` requiring modules of iflib(9) consumers to be rebuilt.
1200074	r336360	16 de julho de 2018	12.0-CURRENT after updating the configuration of libstdc++ to make use of C99 functions.

Valor	Revisão	Data	Release
1200075	r336538	19 de julho de 2018	12.0-CURRENT after zfsloader being folded into loader, and after adding ntpd:ntpd as uid:gid 123:123, and after removing arm big-endian support (MACHINE_ARCH=arm eb).
1200076	r336914	30 de julho de 2018	12.0-CURRENT after KPI changes to timespecadd.
1200077	r337576	10 de agosto de 2018	12.0-CURRENT after timespec_get(3) was added to the system.
1200078	r337863	15 de agosto de 2018	12.0-CURRENT after exec.created hook for jails.
1200079	r338061	19 de agosto de 2018	12.0-CURRENT after converting `arc4random` to using the Chacha20 algorithm and deprecating `arc4random_stir` and `arc4random_addrandom`.
1200080	r338172	22 de agosto de 2018	12.0-CURRENT after removing the drm drivers.
1200081	r338182	21 de agosto de 2018	12.0-CURRENT after KPI changes to NVMe.
1200082	r338285	24 de agosto de 2018	12.0-CURRENT after reverting the removal of the drm drivers.
1200083	r338331	26 de agosto de 2018	12.0-CURRENT after removing `arc4random_stir` and `arc4random_addrandom`.
1200084	r338478	5 de setembro de 2018	12.0-CURRENT after updating objcopy(1) to properly handle little-endian MIPS64 object files.

Valor	Revisão	Data	Release
1200085	r339270	19 de outubro de 2018	12.0-STABLE after updating OpenSSL to version 1.1.1.
1200086	r339732	25 de outubro de 2018	12.0-STABLE after updating OpenSSL shared library version numbers.
1200500	r340471	16 de novembro de 2018	12-STABLE after releng/12.0 was branched.
1200501	r342801	6 de janeiro de 2019	12-STABLE após o merge do fix para o comportamento do linux_destroy_dev () quando ainda existem arquivos abertos a partir da cdev que esta sendo destruída.
1200502	r343126	17 de janeiro de 2019	12-STABLE após habilitar o sys/random.h #include do C ++.
1200503	r344152	15 de fevereiro de 2019	12-STABLE after merge of fixing renameat(2) for CAPABILITIES kernels.
1200504	r345169	15 de março de 2019	12-STABLE after merging CCM for the benefit of the ZoF port.
1200505	r345327	20 de março de 2019	12-STABLE after merging support for selectively disabling ZFS without disabling loader.
1200506	r346168	12 de abril de 2019	12-STABLE after merging llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp 8.0.0 final release r356365.

Valor	Revisão	Data	Release
1200507	r346337	17 de abril de 2019	12-STABLE after MFC of iflib changes in r345303, r345658, and partially of r345305.
1200508	r346784	27 de abril de 2019	12-STABLE after ether_gen_addr availability.
1200509	r347790	16 de maio de 2019	12-STABLE after bumping the Mellanox driver version numbers (mlx4en(4); mlx5en(4)).
1200510	r348036	21 de maio de 2019	12-STABLE after change to struct in linuxkpi from r348035.
1200511	r348243	24 de maio de 2019	12-STABLE after MFC of r347843: adding group_leader member to struct task_struct to the LinuxKPI.
1200512	r348245	24 de maio de 2019	12-STABLE after adding context member to ww_mutex in LinuxKPI.
1200513	r349763	5 de julho de 2019	12-STABLE after MFC of epoch(9) changes: r349763, r340404, r340415, r340417, r340419, r340420.
1200514	r350083	17 de julho de 2019	12-STABLE after additions to LinuxKPI's rcu list.
1200515	r350877	11 de agosto de 2019	12-STABLE after MFC of r349891 (reorganize the SRCS lists as one file per line, and then alphabetize them) and r349972 (add arm_sync_icache() and arm_drain_writebuf() sysarch syscall wrappers).

Valor	Revisão	Data	Release
1200516	r351276	20 de agosto de 2019	12-STABLE after MFC of various changes to iflib r351276.
1200517	r352076	9 de setembro de 2019	12-STABLE after adding sysfs create/remove functions that handles multiple files in one call to the LinuxKPI.
1200518	r352114	10 de setembro de 2019	12-STABLE after additional updates to LinuxKPI's sysfs.
1200519	r352351	15 de setembro de 2019	12-STABLE after MFC of the new fusefs driver.
1201000	r352546	20 de setembro de 2019	releng/12.1 branched from stable/12@r352480.
1201500	r352547	20 de setembro de 2019	12-STABLE after branching releng/12.1.
1201501	r354598	10 de novembro de 2019	12-STABLE after fixing a potential OOB read security issue in libc++.
1201502	r354613	11 de novembro de 2019	12-STABLE after enabling device class group attributes in the LinuxKPI.
1201503	r354928	21 de novembro de 2019	12-STABLE after adding support for AT_EXECPATH to elf_aux_info(3).
1201504	r355658	10 de novembro de 2019	12-STABLE after correcting the C++ version check for declaring timespec_get(3).
1201505	r355899	19 de dezembro de 2019	12-STABLE after adding sigsetop extensions commonly found in musl libc and glibc.
1201506	r355968	21 de dezembro de 2019	12-STABLE after doubling the value of `ARG_MAX`, for 64 bit platforms.

Valor	Revisão	Data	Release
1201507	r356306	2 de janeiro de 2020	12-STABLE after adding functions to bitstring(3) to find contiguous sequences of set or unset bits.
1201508	r356394	6 de janeiro de 2020	12-STABLE after making USB statistics be per-device instead of per bus.
1201509	r356460	7 de janeiro de 2020	12-STABLE after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 9.0.0 final release r372316.
1201510	r356679	13 de janeiro de 2020	12-STABLE after adding own counter for cancelled USB transfers.
1201511	r357333	31 de janeiro de 2020	12-STABLE after adding /etc/os-release as a symbolic link to /var/run/os-release.
1201512	r357612	6 de fevereiro de 2020	12-STABLE after recent LinuxKPI changes.
1201513	r359957	15 de abril de 2020	12-STABLE after cloning the RCU interface into a sleepable and a non-sleepable part in the LinuxKPI.
1201514	r360525	1 de maio de 2020	12-STABLE after implementing full bus_dma(9) support in the LinuxKPI and pulling in all dependencies.
1201515	r360545	1 de maio de 2020	12-STABLE after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 10.0.0 release.

Valor	Revisão	Data	Release
1201516	r360620	4 de maio de 2020	12-STABLE after moving `id_mapped` to end of `bus_dma_impl` structure to preserve KPI.
1201517	r361350	21 de maio de 2020	12-STABLE after renaming `vm.max_wired` to `vm.max_user_wired` and changing its type.
1201518	r362319	18 de junho de 2020	12-STABLE after implementing __is_constexpr() function macro in the LinuxKPI.
1201519	r362916	4 de julho de 2020	12-STABLE after making liblzma use libmd implementation of SHA256.
1201520	r363494	24 de julho de 2020	12-STABLE after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 10.0.1 release.
1201521	r363790	3 de agosto de 2020	12-STABLE after implementing the array_size() function in the LinuxKPI.
1201522	r363832	4 de agosto de 2020	12-STABLE after adding sysctlbyname system call.
1201523	r364390	19 de agosto de 2020	12-STABLE after change to clone the task struct fields related to RCU.
1201524	r365356	5 de setembro de 2020	12-STABLE after splitting XDR off into a separate kernel module, to minimize ZFS dependencies.

Valor	Revisão	Data	Release
1201525	r365471	8 de setembro de 2020	12-STABLE after adding atomic and bswap functions to libcompiler_rt.
1201526	r365608	10 de setembro de 2020	12-STABLE after updating net80211 and kernel privilege checking API changes.
1202000	r365618	11 de setembro de 2020	releng/12.2 branched from stable/12@r365618.
1202500	r365619	11 de setembro de 2020	12-STABLE after branching releng/12.2.
1202501	r365661	12 de setembro de 2020	12-STABLE after followup commits to libcompiler_rt.
1202502	r365816	16 de setembro de 2020	12-STABLE after fixing crunchgen(1) application build with `WARNS=6`.
1202503	r366878	20 de outubro de 2020	12-STABLE after populating the acquire context field of a `ww_mutex` in the LinuxKPI.
1202504	r367511	9 de novembro de 2020	12-STABLE after the addition of `ptsname_r`.

18.3. Versões do FreeBSD 11

Tabela 55. Valores do `__FreeBSD_version` para o FreeBSD 11

Valor	Revisão	Data	Release
1100000	r256284	10 de outubro de 2013	11.0-CURRENT.
1100001	r256776	19 de outubro de 2013	11.0-CURRENT after addition of support for "first boot" rc.d scripts, so ports can make use of this.
1100002	r257696	5 de novembro de 2013	11.0-CURRENT after dropping support for historic ioctls.

Valor	Revisão	Data	Release
1100003	r258284	17 de novembro de 2013	11.0-CURRENT after iconv changes.
1100004	r259424	15 de dezembro de 2013	11.0-CURRENT after the behavior change of `gss_pseudo_random` introduced in r259286.
1100005	r260010	28 de dezembro de 2013	11.0-CURRENT after r259951 - Do not coalesce entries in vm_map_stack(9).
1100006	r261246	28 de janeiro de 2014	11.0-CURRENT after upgrades of libelf and libdwarf.
1100007	r261283	30 de janeiro de 2014	11.0-CURRENT after upgrade of libc++ to 3.4 release.
1100008	r261881	14 de fevereiro de 2014	11.0-CURRENT after libc++ 3.4 ABI compatibility fix.
1100009	r261991	16 de fevereiro de 2014	11.0-CURRENT after upgrade of llvm/clang to 3.4 release.
1100010	r262630	28 de fevereiro de 2014	11.0-CURRENT after upgrade of ncurses to 5.9 release (rev r262629).
1100011	r263102	13 de março de 2014	11.0-CURRENT after ABI change in struct if_data.
1100012	r263140	14 de março de 2014	11.0-CURRENT after removal of Novell IPX protocol support.
1100013	r263152	14 de março de 2014	11.0-CURRENT after removal of AppleTalk protocol support.

Valor	Revisão	Data	Release
1100014	r263235	16 de março de 2014	11.0-CURRENT after renaming <sys/capability.h> to <sys/capsicum.h> to avoid a clash with similarly named headers in other operating systems. A compatibility header is left in place to limit build breakage, but will be deprecated in due course.
1100015	r263620	22 de março de 2014	11.0-CURRENT after `cnt` rename to `vm_cnt`.
1100016	r263660	23 de março de 2014	11.0-CURRENT after addition of `armv6hf` `TARGET_ARCH`.
1100017	r264121	4 de abril de 2014	11.0-CURRENT after GCC support for `__block` definition.
1100018	r264212	6 de abril de 2014	11.0-CURRENT after support for UDP-Lite protocol (RFC 3828).
1100019	r264289	8 de abril de 2014	11.0-CURRENT after FreeBSD-SA-14:06.openssl (rev r264265).
1100020	r265215	1 de maio de 2014	11.0-CURRENT after removing lindev in favor of having /dev/full by default (rev r265212).
1100021	r266151	6 de maio de 2014	11.0-CURRENT after src.opts.mk changes, decoupling make.conf(5) from `buildworld` (rev r265419).

Valor	Revisão	Data	Release
1100022	r266904	30 de maio de 2014	11.0-CURRENT after changes to strcasecmp(3), moving strcasecmp_l(3) and strncasecmp_l(3) from <string.h> to <strings.h> for POSIX 2008 compliance (rev r266865).
1100023	r267440	13 de junho de 2014	11.0-CURRENT after the CUSE library and kernel module have been attached to the build by default.
1100024	r267992	27 de junho de 2014	11.0-CURRENT after sysctl(3)API change.
1100025	r268066	30 de junho de 2014	11.0-CURRENT after regex(3) library update to add ">" and "<" delimiters.
1100026	r268118	1 de julho de 2014	11.0-CURRENT after the internal interface between the NFS modules, including the krpc, was changed by (rev r268115).
1100027	r268441	8 de julho de 2014	11.0-CURRENT after FreeBSD-SA-14:17.kmem (rev r268431).
1100028	r268945	21 de julho de 2014	11.0-CURRENT after hdestroy(3) compliance fix changed ABI.
1100029	r270173	3 de agosto de 2014	11.0-CURRENT after `SOCK_DGRAM` bug fix (rev r269489).
1100030	r270929	1º de setembro de 2014	11.0-CURRENT after `SOCK_RAW` sockets were changed to not modify packets at all.

Valor	Revisão	Data	Release
1100031	r271341	9 de setembro de 2014	11.0-CURRENT after FreeBSD-SA-14:18.openssl (rev r269686).
1100032	r271438	11 de setembro de 2014	11.0-CURRENT after API changes to `ifa_ifwithbroadaddr`, `ifa_ifwithdstaddr`, `ifa_ifwithnet`, and `ifa_ifwithroute`.
1100033	r271657	9 de setembro de 2014	11.0-CURRENT after changing `access`, `eaccess`, and `faccessat` to validate the mode argument.
1100034	r271686	16 de setembro de 2014	11.0-CURRENT after FreeBSD-SA-14:19.tcp (rev r271666).
1100035	r271705	17 de setembro de 2014	11.0-CURRENT after i915 HW context support.
1100036	r271724	17 de setembro de 2014	Version bump to have ABI note distinguish binaries ready for strict mmap(2) flags checking (rev r271724).
1100037	r272674	6 de outubro de 2014	11.0-CURRENT after addition of explicit_bzero(3) (rev r272673).
1100038	r272951	11 de outubro de 2014	11.0-CURRENT after cleanup of TCP wrapper headers.
1100039	r273250	18 de outubro de 2014	11.0-CURRENT after removal of `MAP_RENAME` and `MAP_NORESERVE`.
1100040	r273432	21 de outubro de 2014	11.0-CURRENT after FreeBSD-SA-14:23 (rev r273146).

Valor	Revisão	Data	Release
1100041	r273875	30 de outubro de 2014	11.0-CURRENT after API changes to `syscall_register`, `syscall32_register`, `syscall_register_helper` and `syscall32_register_helper` (rev r273707).
1100042	r274046	3 de novembro de 2014	11.0-CURRENT after a change to `struct tcpcb`.
1100043	r274085	4 de novembro de 2014	11.0-CURRENT after enabling vt(4) by default.
1100044	r274116	4 de novembro de 2014	11.0-CURRENT after adding new libraries/utilities (dpv and figpar) for data throughput visualization.
1100045	r274162	4 de novembro de 2014	11.0-CURRENT after FreeBSD-SA-14:23, FreeBSD-SA-14:24, and FreeBSD-SA-14:25.
1100046	r274470	13 de novembro de 2014	11.0-CURRENT after `kern_poll` signature change (rev r274462).
1100047	r274476	13 de novembro de 2014	11.0-CURRENT after removal of no-at version of VFS syscalls helpers, like `kern_open`.
1100048	r275358	1º de dezembro de 2014	11.0-CURRENT after starting the process of removing the use of the deprecated "M_FLOWID" flag from the network code.
1100049	r275633	9 de dezembro de 2014	11.0-CURRENT after importing an important fix to the LLVM vectorizer, which could lead to buffer overruns in some cases.

Valor	Revisão	Data	Release
1100050	r275732	12 de dezembro de 2014	11.0-CURRENT after adding AES-ICM and AES-GCM to OpenCrypto.
1100051	r276096	23 de dezembro de 2014	11.0-CURRENT after removing old NFS client and server code from the kernel.
1100052	r276479	31 de dezembro de 2014	11.0-CURRENT after upgrade of clang, llvm and lldb to 3.5.0 release.
1100053	r276781	7 de janeiro de 2015	11.0-CURRENT after MCLGET(9) gained a return value (rev r276750).
1100054	r277213	15 de janeiro de 2015	11.0-CURRENT after rewrite of callout subsystem.
1100055	r277528	22 de janeiro de 2015	11.0-CURRENT after reverting callout changes in r277213.
1100056	r277610	23 de janeiro de 2015	11.0-CURRENT after addition of `futimens` and `utimensat` system calls.
1100057	r277897	29 de janeiro de 2015	11.0-CURRENT after removal of d_thread_t.
1100058	r278228	5 de fevereiro de 2015	11.0-CURRENT after addition of support for probing the SCSI VPD Extended Inquiry page (0x86).
1100059	r278442	9 de fevereiro de 2015	11.0-CURRENT after import of xz 5.2.0, which added multi-threaded compression and lzma gained libthr dependency (rev r278433).

Valor	Revisão	Data	Release
1100060	r278846	16 de fevereiro de 2015	11.0-CURRENT after forwarding `FBIO_BLANK` to framebuffer clients.
1100061	r278964	18 de fevereiro de 2015	11.0-CURRENT after `CDAI_FLAG_NONE` addition.
1100062	r279221	23 de fevereiro de 2015	11.0-CURRENT after mtio(4) and sa(4)API and ioctl(2) additions.
1100063	r279728	7 de março de 2015	11.0-CURRENT after adding mutex support to the `pps_ioctl()` API in the kernel.
1100064	r279729	7 de março de 2015	11.0-CURRENT after adding PPS support to USB serial drivers.
1100065	r280031	15 de março de 2015	11.0-CURRENT after upgrading clang, llvm and lldb to 3.6.0.
1100066	r280306	20 de março de 2015	11.0-CURRENT after removal of SSLv2 support from OpenSSL.
1100067	r280630	25 de março de 2015	11.0-CURRENT after removal of SSLv2 support from fetch(1) and fetch(3).
1100068	r281172	6 de abril de 2015	11.0-CURRENT after change to net.inet6.ip6.mif6table sysctl.
1100069	r281550	15 de abril de 2015	11.0-CURRENT after removal of const qualifier from iconv(3).
1100070	r281613	16 de abril de 2015	11.0-CURRENT after moving ALTQ from contrib to net/altq.
1100071	r282256	29 de abril de 2015	11.0-CURRENT after API/ABI change to smb(4) (rev r281985).

Valor	Revisão	Data	Release
1100072	r282319	1 de maio de 2015	11.0-CURRENT after adding reallocarray(3) in libc (rev r282314).
1100073	r282650	8 de maio de 2015	11.0-CURRENT after extending the maximum number of allowed PCM channels in a PCM stream to 127 and decreasing the maximum number of sub-channels to 1.
1100074	r283526	25 de maio de 2015	11.0-CURRENT after adding preliminary support for x86-64 Linux binaries (rev r283424), and upgrading clang and llvm to 3.6.1.
1100075	r283623	27 de maio de 2015	11.0-CURRENT after `dounmount()` requiring a reference on the passed struct mount (rev r283602).
1100076	r283983	4 de junho de 2015	11.0-CURRENT after disabled generation of legacy formatted password databases entries by default.
1100077	r284233	10 de junho de 2015	11.0-CURRENT after API changes to `lim_cur`, `lim_max`, and `lim_rlimit` (rev r284215).
1100078	r286672	12 de agosto de 2015	11.0-CURRENT after crunchgen(1) changes from r284356 to r285986.
1100079	r286874	18 de agosto de 2015	11.0-CURRENT after import of jemalloc 4.0.0 (rev r286866).
1100080	r288943	5 de outubro de 2015	11.0-CURRENT after upgrading clang, llvm, lldb, compiler-rt and libc++ to 3.7.0.

Valor	Revisão	Data	Release
1100081	r289415	16 de outubro de 2015	11.0-CURRENT after undating ZFS to support resumable send/receive (rev r289362).
1100082	r289594	19 de outubro de 2015	11.0-CURRENT after Linux KPI updates.
1100083	r289749	22 de outubro de 2015	11.0-CURRENT after renaming linuxapi.ko to linuxkpi.ko.
1100084	r290135	29 de outubro de 2015	11.0-CURRENT after moving the LinuxKPI module into the default kernel build.
1100085	r290207	30 de outubro de 2015	11.0-CURRENT after import of OpenSSL 1.0.2d.
1100086	r290275	2 de novembro de 2015	11.0-CURRENT after making figpar(3) macros more unique.
1100087	r290479	7 de novembro de 2015	11.0-CURRENT after changing sysctl_add_oid(9)'s ABI.
1100088	r290495	7 de novembro de 2015	11.0-CURRENT after string collation and locales rework.
1100089	r290505	7 de novembro de 2015	11.0-CURRENT after API change to sysctl_add_oid(9) (rev r290475).
1100090	r290715	10 de novembro de 2015	11.0-CURRENT after API change to callout_stop macro; (rev r290664).
1100091	r291537	30 de novembro de 2015	11.0-CURRENT after changing the interface between the nfsd.ko and nfscommon.ko modules in r291527.
1100092	r292499	19 de dezembro de 2015	11.0-CURRENT after removal of vm_pageout_grow_cache (rev r292469).

Valor	Revisão	Data	Release
1100093	r292966	30 de dezembro de 2015	11.0-CURRENT after removal of sys/crypto/sha2.h (rev r292782).
1100094	r294086	15 de janeiro de 2016	11.0-CURRENT after LinuxKPI PCI changes (rev r294086).
1100095	r294327	19 de janeiro de 2016	11.0-CURRENT after LRO optimizations.
1100096	r294505	21 de janeiro de 2016	11.0-CURRENT after LinuxKPI idr_* additions.
1100097	r294860	26 de janeiro de 2016	11.0-CURRENT after API change to dpv(3).
1100098	r295682	16 de fevereiro de 2016	11.0-CURRENT after API change to rman (rev r294883).
1100099	r295739	18 de fevereiro de 2016	11.0-CURRENT after allowing drivers to set the TCP ACK/data segment aggregation limit.
1100100	r296136	26 de fevereiro de 2016	11.0-CURRENT after bus_alloc_resource_any (9) API addition.
1100101	r296417	5 de março de 2016	11.0-CURRENT after upgrading our copies of clang, llvm, lldb and compiler-rt to 3.8.0 release.
1100102	r296749	12 de março de 2016	11.0-CURRENT after libelf cross-endian fix in rev r296685.
1100103	r297000	18 de março de 2016	11.0-CURRENT after using uintmax_t for rman ranges.
1100104	r297156	21 de março de 2016	11.0-CURRENT after tracking filemon usage via a proc.p_filemon pointer rather than its own lists.

Valor	Revisão	Data	Release
1100105	r297602	6 de abril de 2016	11.0-CURRENT after fixing sed functions `i` and `a` from discarding leading white space.
1100106	r298486	22 de abril de 2016	11.0-CURRENT after fixes for using IPv6 addresses with RDMA.
1100107	r299090	4 de maio de 2016	11.0-CURRENT after improving performance and functionality of the bitstring(3) api.
1100108	r299530	12 de maio de 2016	11.0-CURRENT after fixing handling of IOCTLs in the LinuxKPI.
1100109	r299933	16 de maio de 2016	11.0-CURRENT after implementing more Linux device related functions in the LinuxKPI.
1100110	r300207	19 de maio de 2016	11.0-CURRENT after adding support for managing Shingled Magnetic Recording (SMR) drives.
1100111	r300303	20 de maio de 2016	11.0-CURRENT after removing brk and sbrk from arm64.
1100112	r300539	23 de maio de 2016	11.0-CURRENT after adding bit_count to the bitstring(3) API.
1100113	r300701	26 de maio de 2016	11.0-CURRENT after disabling alignment faults on armv6.
1100114	r300806	26 de maio de 2016	11.0-CURRENT after fixing crunchgen(1) usage with `MAKEOBJDIRPREFIX`.
1100115	r300982	30 de maio de 2016	11.0-CURRENT after adding an mbuf flag for `M_HASHTYPE_`.

Valor	Revisão	Data	Release
1100116	r301011	31 de maio de 2016	11.0-CURRENT after SHA-512t256 (rev r300903) and Skein (rev r300966) where added to libmd, libcrypt, the kernel, and ZFS (rev r301010).
1100117	r301892	6 de junho de 2016	11.0-CURRENT after libpam was synced with stock r301602, bumping library version.
1100118	r302071	21 de junho de 2016	11.0-CURRENT after breaking binary compatibility of struct disk r302069.
1100119	r302150	23 de junho de 2016	11.0-CURRENT after switching geom_disk to using a pool mutex.
1100120	r302153	23 de junho de 2016	11.0-CURRENT after adding spares to struct ifnet.
1100121	r303979	12 de agosto de 2015	11-STABLE after `releng/11.0` branched from 11-STABLE (rev r303975).
1100500	r303979	12 de agosto de 2016	11.0-STABLE adding branched r303976.
1100501	r304609	22 de agosto de 2016	11.0-STABLE after adding C++11 thread_local support.
1100502	r304865	26 de agosto de 2016	11.0-STABLE after `LC*MASK` fix.
1100503	r305733	12 de setembro de 2016	11.0-STABLE after resolving a deadlock between `device_detach()` and usbd_do_request_flags(9).
1100504	r307330	14 de outubro de 2016	11.0-STABLE after ZFS merges.

Valor	Revisão	Data	Release
1100505	r307590	19 de outubro de 2016	11.0-STABLE after `struct fb_info` change.
1100506	r308048	28 de outubro de 2016	11.0-STABLE after installing header files required development with libzfs_core.
1100507	r310120	15 de dezembro de 2016	11.0-STABLE after adding the `ki_moretdname` member to `struct kinfo_proc` and `struct kinfo_proc32` to export the whole thread name to user-space utilities.
1100508	r310618	26 de dezembro de 2016	11.0-STABLE after upgrading our copies of clang, llvm, lldb, compiler-rt and libc++ to 3.9.1 release, and adding lld 3.9.1.
1100509	r311186	3 de janeiro de 2017	11.0-STABLE after crunchgen(1) META_MODE fix (rev r311185).
1100510	r315312	15 de março de 2017	11.0-STABLE after MFC of `fget_cap`, `getsock_cap`, and related changes.
1100511	r316423	2 de abril de 2017	11.0-STABLE after multiple MFCs updating clang, llvm, lld, lldb, compiler-rt and libc++ to 4.0.0 release.
1100512	r316498	4 de abril de 2017	11.0-STABLE after making CAM SIM lock optional (revs r315673, r315674).
1100513	r318197	11 de maio de 2017	11.0-STABLE after merging the addition of the <dev/mmc/mmc_ioctl.h> header.

Valor	Revisão	Data	Release
1100514	r319279	31 de maio de 2017	11.0-STABLE after multiple MFCs of `libpcap`, `WITHOUT_INET6`, and a few other minor changes.
1101000	r320486	30 de junho de 2017	`releng/11.1` branched from `stable/11`.
1101001	r320763	30 de junho de 2017	11.1-RC1 após de fundir a adição de flag `MAP_GUARD` mmap(2).
1101500	r320487	30 de junho de 2017	11-STABLE after `releng/11.1` branched.
1101501	r320666	5 de julho de 2017	11-STABLE after merging the `MAP_GUARD` mmap(2) flag addition.
1101502	r321688	29 de julho de 2017	11-STABLE after merging the NFS client forced dismount support `umount -N` addition.
1101503	r323431	11 de setembro de 2017	11-STABLE after merging changes making the WRFSBASE instruction operational on amd64.
1101504	r324006	26 de setembro de 2017	11-STABLE after merging libm from head, which adds cacoshl(3), cacosl(3), casinhl(3), casinl(3), catanl(3), catanhl(3), sincos(3), sincosf(3), and sincosl(3).
1101505	r324023	26 de setembro de 2017	11-STABLE after merging clang, llvm, lld, lldb, compiler-rt and libc++ 5.0.0 release.
1101506	r325003	25 de outubro de 2017	11-STABLE after merging r324281, adding the `value.u16` field to `struct diocgattr_arg`.

Valor	Revisão	Data	Release
1101507	r328379	24 de janeiro de 2018	11-STABLE after merging r325028, fixing `ptrace()` to always clear the correct thread event when resuming.
1101508	r328386	24 de janeiro de 2018	11-STABLE after merging r316648, renaming smp_no_rendevous_barrier() to smp_no_rendezvous_barrier().
1101509	r328653	1 de fevereiro de 2018	11-STABLE after an overwrite merge backport of the LinuxKPI from FreeBSD-head.
1101510	r329450	17 de fevereiro de 2018	11-STABLE after the cmpxchg() macro is now fully functional in the LinuxKPI.
1101511	r329981	25 de fevereiro de 2018	11-STABLE after concluding the recent LinuxKPI related updates.
1101512	r331219	19 de março de 2018	11-STABLE after merging retpoline support from the upstream llvm, clang and lld 5.0 branches.
1101513	r331838	31 de março de 2018	11-STABLE after merging clang, llvm, lld, lldb, compiler-rt and libc++ 6.0.0 release, and several follow-up fixes.
1101514	r332089	5 de abril de 2018	11-STABLE after merging r328331, adding a new and incompatible interpretation of ${name}_limits in rc scripts.

Valor	Revisão	Data	Release
1101515	r332363	10 de abril de 2018	11-STABLE after reverting r331880, removing the new and incompatible interpretation of ${name}_limits in rc scripts.
1101516	r334392	30 de maio de 2018	11-STABLE after dwatch(1) touch-ups.
1102000	r334459	1 de junho de 2018	`releng/11.2` branched from `stable/11`.
1102500	r334461	1 de junho de 2018	11-STABLE after releng/11.2 branched.
1102501	r335436	20 de junho de 2018	11-STABLE after LinuxKPI updates requiring recompilation of external kernel modules.
1102502	r338617	12 de setembro de 2018	11-STABLE after adding a socket option SO_TS_CLOCK and fixing recvmsg32() system call to properly down-convert layout of the 64-bit structures to match what 32-bit app(s) expect.
1102503	r338931	25 de setembro de 2018	11-STABLE after merging a TCP checksum fix to iflib(9) and adding new media types to if_media.h
1102504	r340309	9 de novembro de 2018	11-STABLE after several MFCs: updating objcopy(1) to properly handle little-endian MIPS64 object; correcting mips64el test to use ELF header; adding test for 64-bit ELF in _libelf_is_mips64el.

Valor	Revisão	Data	Release
1102505	r342804	6 de janeiro de 2019	11-STABLE após o merge do fix para o comportamento do linux_destroy_dev () quando ainda existem arquivos abertos a partir da cdev que esta sendo destruída.
1102506	r344220	17 de fevereiro de 2019	11-STABLE depois de mesclar vários commits para o lualoader.
1102507	r346296	16 de abril de 2019	11-STABLE after merging llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp 8.0.0 final release r356365.
1102508	r346784	27 de abril de 2019	11-STABLE after ether_gen_addr availability.
1102509	r347212	6 de maio de 2019	11-STABLE after MFC of r345303, r345658, and partially of r345305.
1102510	r347883	16 de maio de 2019	11-STABLE after bumping the Mellanox driver version numbers (mlx4en(4); mlx5en(4)).
1103000	r349026	14 de junho de 2019	`releng/11.3` branched from `stable/11`.
1103500	r349027	14 de junho de 2019	11-STABLE after releng/11.3 branched.
1103501	r354598	10 de novembro de 2019	11-STABLE after fixing a potential OOB read security issue in libc++.
1103502	r354614	11 de novembro de 2019	11-STABLE after adding sysfs create/remove functions that handles multiple files in one call to the LinuxKPI.
1103503	r354615	11 de novembro de 2019	11-STABLE after LinuxKPI sysfs improvements.

Valor	Revisão	Data	Release
1103504	r354616	11 de novembro de 2019	11-STABLE after enabling device class group attributes in the LinuxKPI.
1103505	r355899	19 de dezembro de 2019	11-STABLE after adding sigsetop extensions commonly found in musl libc and glibc.
1103506	r356395	6 de janeiro de 2020	11-STABLE after making USB statistics be per-device instead of per bus.
1103507	r356680	13 de janeiro de 2020	11-STABLE after adding own counter for cancelled USB transfers.
1103508	r357613	6 de fevereiro de 2020	11-STABLE after recent LinuxKPI changes.
1103509	r359958	15 de abril de 2020	11-STABLE after moving `id_mapped` to end of `bus_dma_impl` structure to preserve KPI.
1103510	r360658	5 de maio de 2020	11-STABLE after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 9.0.0 final release r372316.
1103511	r360784	7 de maio de 2020	11-STABLE after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 10.0.0 release.
1104000	r360804	8 de maio de 2020	`releng/11.4` branched from `stable/11`.

Valor	Revisão	Data	Release
1104001	r360822	8 de maio de 2020	11.4-BETA1 after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 10.0.0 release.
1104500	r360805	8 de maio de 2020	11-STABLE after releng/11.4 branched.
1104501	r362320	18 de junho de 2020	11-STABLE after implementing __is_constexpr() function macro in the LinuxKPI.
1104502	r362919	4 de julho de 2020	11-STABLE after making liblzma use libmd implementation of SHA256.
1104503	r363496	24 de julho de 2020	11-STABLE after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 10.0.1 release.
1104504	r363792	3 de agosto de 2020	11-STABLE after implementing the array_size() function in the LinuxKPI.
1104505	r364391	19 de agosto de 2020	11-STABLE after change to clone the task struct fields related to RCU.
1104506	r365471	8 de setembro de 2020	11-STABLE after adding atomic and bswap functions to libcompiler_rt.
1104507	r365661	12 de setembro de 2020	11-STABLE after followup commits to libcompiler_rt.
1104508	r366879	20 de outubro de 2020	11-STABLE after populating the acquire context field of a `ww_mutex` in the LinuxKPI.

Valor	Revisão	Data	Release
1104509	r366889	20 de outubro de 2020	11-STABLE after additions to LinuxKPI's `RCU` list.
1104510	r367513	9 de novembro de 2020	11-STABLE after the addition of `ptsname_r`.

18.4. Versões do FreeBSD 10

Tabela 56. Valores do `__FreeBSD_version` para o FreeBSD 10

Valor	Revisão	Data	Release
1000000	r225757	26 de setembro de 2011	10.0-CURRENT.
1000001	r227070	4 de novembro de 2011	10-CURRENT after addition of the posix_fadvise(2) system call.
1000002	r228444	12 de dezembro de 2011	10-CURRENT after defining boolean true/false in sys/types.h, sizeof(bool) may have changed (rev r228444). 10-CURRENT after xlocale.h was introduced (rev r227753).
1000003	r228571	16 de dezembro de 2011	10-CURRENT after major changes to carp(4), changing size of struct in_aliasreq, struct in6_aliasreq (rev r228571) and straitening arguments check of SIOCAIFADDR (rev r228574).
1000004	r229204	1 de janeiro de 2012	10-CURRENT after the removal of `skpc()` and the addition of memcchr(9) (rev r229200).

Valor	Revisão	Data	Release
1000005	r230207	16 de janeiro de 2012	10-CURRENT after the removal of support for SIOCSIFADDR, SIOCSIFNETMASK, SIOCSIFBRDADDR, SIOCSIFDSTADDR ioctls.
1000006	r230590	26 de janeiro de 2012	10-CURRENT after introduction of read capacity data asynchronous notification in the cam(4) layer.
1000007	r231025	5 de fevereiro de 2012	10-CURRENT after introduction of new tcp(4) socket options: TCP_KEEPINIT, TCP_KEEPIDLE, TCP_KEEPINTVL, and TCP_KEEPCNT.
1000008	r231505	11 de fevereiro de 2012	10-CURRENT after introduction of the new extensible sysctl(3) interface NET_RT_IFLISTL to query address lists.
1000009	r232154	25 de fevereiro de 2012	10-CURRENT after import of libarchive 3.0.3 (rev r232153).
1000010	r233757	31 de março de 2012	10-CURRENT after xlocale cleanup.
1000011	r234355	16 de abril de 2012	10-CURRENT import of LLVM/Clang 3.1 trunk r154661 (rev r234353).
1000012	r234924	2 de maio de 2012	10-CURRENT jemalloc import.
1000013	r235788	22 de maio de 2012	10-CURRENT after byacc import.
1000014	r237631	27 de junho de 2012	10-CURRENT after BSD sort becoming the default sort (rev r237629).

Valor	Revisão	Data	Release
1000015	r238405	12 de julho de 2012	10-CURRENT after import of OpenSSL 1.0.1c.
(Não mudou)	r238429	13 de julho de 2012	10-CURRENT after the fix for LLVM/Clang 3.1 regression.
1000016	r239179	8 de agosto de 2012	10-CURRENT after KBI change in ucom(4).
1000017	r239214	8 de agosto de 2012	10-CURRENT after adding streams feature to the USB stack.
1000018	r240233	8 de setembro de 2012	10-CURRENT after major rewrite of pf(4).
1000019	r241245	6 de outubro de 2012	10-CURRENT after pfil(9) KBI/KPI changed to supply packets in net byte order to AF_INET filter hooks.
1000020	r241610	16 de outubro de 2012	10-CURRENT after the network interface cloning KPI changed and struct if_clone becoming opaque.
1000021	r241897	22 de outubro de 2012	10-CURRENT after removal of support for non-MPSAFE filesystems and addition of support for FUSEFS (rev r241519).
1000022	r241913	22 de outubro de 2012	10-CURRENT after the entire IPv4 stack switched to network byte order for IP packet header storage.
1000023	r242619	5 de novembro de 2012	10-CURRENT after jitter buffer in the common USB serial driver code, to temporarily store characters if the TTY buffer is full. Add flow stop and start signals when this happens.

Valor	Revisão	Data	Release
1000024	r242624	5 de novembro de 2012	10-CURRENT after clang was made the default compiler on i386 and amd64.
1000025	r243443	17 de novembro de 2012	10-CURRENT after the sin6_scope_id member variable in struct sockaddr_in6 was changed to being filled by the kernel before passing the structure to the userland via sysctl or routing socket. This means the KAME-specific embedded scope id in sin6_addr.s6_addr[2] is always cleared in userland application.
1000026	r245313	11 de janeiro de 2013	10-CURRENT after install gained the -N flag. May also be used to indicate the presence of nmtree.
1000027	r246084	29 de janeiro de 2013	10-CURRENT after cat gained the -l flag (rev r246083).
1000028	r246759	13 de fevereiro de 2013	10-CURRENT after USB moved to the driver structure requiring a rebuild of all USB modules.
1000029	r247821	4 de março de 2013	10-CURRENT after the introduction of tickless callout facility which also changed the layout of struct callout (rev r247777).
1000030	r248210	12 de março de 2013	10-CURRENT after KPI breakage introduced in the VM subsystem to support read/write locking (rev r248084).

Valor	Revisão	Data	Release
1000031	r249943	26 de abril de 2013	10-CURRENT after the dst parameter of the ifnet `if_output` method was changed to take const qualifier (rev r249925).
1000032	r250163	1 de Maio de 2013	10-CURRENT after the introduction of the accept4(2) (rev r250154) and pipe2(2) (rev r250159) system calls.
1000033	r250881	21 de maio de 2013	10-CURRENT after flex 2.5.37 import.
1000034	r251294	3 de junho de 2013	10-CURRENT after the addition of these functions to libm: cacos(3), cacosf(3), cacosh(3), cacoshf(3), casin(3), casinf(3), casinh(3), casinhf(3), catan(3), catanf(3), catanh(3), catanhf(3), logl(3), log2l(3), log10l(3), log1pl(3), expm1l(3).
1000035	r251527	8 de junho de 2013	10-CURRENT after the introduction of the aio_mlock(2) system call (rev r251526).
1000036	r253049	9 de julho de 2013	10-CURRENT after the addition of a new function to the kernel GSSAPI module's function call interface.

Valor	Revisão	Data	Release
1000037	r253089	9 de julho de 2013	10-CURRENT after the migration of statistics structures to PCPU counters. Changed structures include: `ahstat`, `arpstat`, `espstat`, `icmp6_ifstat`, `icmp6stat`, `in6_ifstat`, `ip6stat`, `ipcompstat`, `ipipstat`, `ipsecstat`, `mrt6stat`, `mrtstat`, `pfkeystat`, `pim6stat`, `pimstat`, `rip6stat`, `udpstat` (rev r253081).
1000038	r253396	16 de julho de 2013	10-CURRENT after making `ARM EABI` the default ABI on arm, armeb, armv6, and armv6eb architectures.
1000039	r253549	22 de julho de 2013	10-CURRENT after `CAM` and mps(4) driver scanning changes.
1000040	r253638	24 de julho de 2013	10-CURRENT after addition of libusb pkgconf files.
1000041	r253970	5 de agosto de 2013	10-CURRENT after change from `time_second` to `time_uptime` in `PF_INET6`.
1000042	r254138	9 de agosto de 2013	10-CURRENT after VM subsystem change to unify soft and hard busy mechanisms.
1000043	r254273	13 de agosto de 2013	10-CURRENT after `WITH_ICONV` is enabled by default. A new src.conf(5) option, `WITH_LIBICONV_COMPAT` (disabled by default) adds `libiconv_open` to provide compatibility with the libiconv port.

Valor	Revisão	Data	Release
1000044	r254358	15 de agosto de 2013	10-CURRENT after `libc.so` conversion to an ld(1) script (rev r251668).
1000045	r254389	15 de agosto de 2013	10-CURRENT after devfs programming interface change by replacing the cdevsw flag `D_UNMAPPED_IO` with the struct cdev flag `SI_UNMAPPED`.
1000046	r254537	19 de agosto de 2013	10-CURRENT after addition of `M_PROTO[9-12]` and removal of `M_FRAG|M_FIRSTFRAG|M_LASTFRAG` mbuf flags (rev r254524, r254526).
1000047	r254627	21 de agosto de 2013	10-CURRENT after stat(2) update to allow storing some Windows/DOS and CIFS file attributes as stat(2) flags.
1000048	r254672	22 de agosto de 2013	10-CURRENT after modification of structure `xsctp_inpcb`.
1000049	r254760	24 de agosto de 2013	10-CURRENT after physio(9) support for devices that do not function properly with split I/O, such as sa(4).
1000050	r254844	24 de agosto de 2013	10-CURRENT after modifications of structure `mbuf` (rev r254780, r254799, r254804, r254807r254842).
1000051	r254887	25 de agosto de 2013	10-CURRENT after Radeon KMS driver import (rev r254885).

Valor	Revisão	Data	Release
1000052	r255180	3 de setembro de 2013	10-CURRENT after import of NetBSD `libexecinfo` is connected to the build.
1000053	r255305	6 de setembro de 2013	10-CURRENT after API and ABI changes to the Capsicum framework.
1000054	r255321	6 de setembro de 2013	10-CURRENT after `gcc` and `libstdc++` are no longer built by default.
1000055	r255449	6 de setembro de 2013	10-CURRENT after addition of `MMAP_32BIT` mmap(2) flag (rev r255426).
1000100	r259065	7 de dezembro de 2013	`releng/10.0` branched from `stable/10`.
1000500	r256283	10 de outubro de 2013	10-STABLE after branch from `head/`.
1000501	r256916	22 de outubro de 2013	10-STABLE after addition of first-boot rc(8) support.
1000502	r258398	20 de novembro de 2013	10-STABLE after removal of iconv symbols from `libc.so.7`.
1000510	r259067	7 de dezembro de 2013	`releng/10.0` __FreeBSD_version update to prevent the value from going backwards.
1000700	r259069	7 de dezembro de 2013	10-STABLE after `releng/10.0` branch.
1000701	r259447	15 de dezembro de 2013	10.0-STABLE after Heimdal encoding fix.
1000702	r260135	31 de dezembro de 2013	10-STABLE after MAP_STACK fixes.
1000703	r262801	5 de março de 2014	10-STABLE after upgrade of libc++ to 3.4 release.

Valor	Revisão	Data	Release
1000704	r262889	7 de março de 2014	10-STABLE after MFC of the vt(4) driver (rev r262861).
1000705	r263508	21 de março de 2014	10-STABLE after upgrade of llvm/clang to 3.4 release.
1000706	r264214	6 de abril de 2014	10-STABLE after GCC support for `__block` definition.
1000707	r264289	8 de abril de 2014	10-STABLE after FreeBSD-SA-14:06.openssl.
1000708	r265122	30 de abril de 2014	10-STABLE after FreeBSD-SA-14:07.devfs, FreeBSD-SA-14:08.tcp, and FreeBSD-SA-14:09.openssl.
1000709	r265946	13 de maio de 2014	10-STABLE after support for UDP-Lite protocol (RFC 3828).
1000710	r267465	13 de junho de 2014	10-STABLE after changes to strcasecmp(3), moving strcasecmp_l(3) and strncasecmp_l(3) from <string.h> to <strings.h> for POSIX 2008 compliance.
1000711	r268442	8 de julho de 2014	10-STABLE after FreeBSD-SA-14:17.kmem (rev r268432).
1000712	r269400	1 de agosto de 2014	10-STABLE after nfsd(8) 4.1 merge (rev r269398).
1000713	r269484	3 de agosto de 2014	10-STABLE after regex(3) library update to add ">" and "<" delimiters.

Valor	Revisão	Data	Release
1000714	r270174	3 de agosto de 2014	10-STABLE after `SOCK_DGRAM` bug fix (rev r269490).
1000715	r271341	9 de setembro de 2014	10-STABLE after FreeBSD-SA-14:18 (rev r269686).
1000716	r271686	16 de setembro de 2014	10-STABLE after FreeBSD-SA-14:19 (rev r271667).
1000717	r271816	18 de setembro de 2014	10-STABLE after i915 HW context support.
1001000	r272463	2 de outubro de 2014	10.1-RC1 after releng/10.1 branch.
1001500	r272464	2 de outubro de 2014	10-STABLE after releng/10.1 branch.
1001501	r273432	21 de outubro de 2014	10-STABLE after FreeBSD-SA-14:20, FreeBSD-SA-14:22, and FreeBSD-SA-14:23 (rev r273411).
1001502	r274162	4 de novembro de 2014	10-STABLE after FreeBSD-SA-14:23, FreeBSD-SA-14:24, and FreeBSD-SA-14:25.
1001503	r275040	25 de novembro de 2014	10-STABLE after merging new libraries/utilities (dpv(1), dpv(3), and figpar(3)) for data throughput visualization.
1001504	r275742	13 de dezembro de 2014	10-STABLE after merging an important fix to the LLVM vectorizer, which could lead to buffer overruns in some cases.
1001505	r276633	3 de janeiro de 2015	10-STABLE after merging some arm constants in r276312.

Valor	Revisão	Data	Release
1001506	r277087	12 de janeiro de 2015	10-STABLE after merging max table size update for yacc.
1001507	r277790	27 de janeiro de 2015	10-STABLE after changes to the UDP tunneling callback to provide a context pointer and the source sockaddr.
1001508	r278974	18 de fevereiro de 2015	10-STABLE after addition of the `CDAI_TYPE_EXT_INQ` request type.
1001509	r279287	25 de fevereiro de 2015	10-STABLE after FreeBSD-EN-15:01.vt, FreeBSD-EN-15:02.openssl, FreeBSD-EN-15:03.freebsd-update, FreeBSD-SA-15:04.igmp, and FreeBSD-SA-15:05.bind.
1001510	r279329	26 de fevereiro de 2015	10-STABLE after MFC of rev r278964.
1001511	r280246	19 de março de 2015	10-STABLE after sys/capability.h is renamed to sys/capsicum.h (rev r280224/).
1001512	r280438	24 de março de 2015	10-STABLE after addition of new mtio(4), sa(4) ioctls.
1001513	r281955	24 de abril de 2015	10-STABLE after starting the process of removing the use of the deprecated "M_FLOWID" flag from the network code.
1001514	r282275	30 de abril de 2015	10-STABLE after MFC of iconv(3) fixes.
1001515	r282781	11 de maio de 2015	10-STABLE after adding back `M_FLOWID`.

Valor	Revisão	Data	Release
1001516	r283341	24 de maio de 2015	10-STABLE after MFC of many USB things.
1001517	r283950	3 de junho de 2015	10-STABLE after MFC of sound related things.
1001518	r284204	10 de junho de 2015	10-STABLE after MFC of zfs vfs fixes (rev r284203).
1001519	r284720	23 de junho de 2015	10-STABLE after reverting bumping `MAXCPU` on amd64.
1002000	r285830	24 de julho de 2015	`releng/10.2` branched from 10-STABLE.
1002500	r285831	24 de julho de 2015	10-STABLE after `releng/10.2` branched from 10-STABLE.
1002501	r289005	8 de outubro de 2015	10-STABLE after merge of ZFS changes that affected the internal interface of zfeature_info structure (rev r288572).
1002502	r291243	24 de novembro de 2015	10-STABLE after merge of dump device changes that affected the arguments of `g_dev_setdumpdev()`(rev r291215).
1002503	r292224	14 de dezembro de 2015	10-STABLE after merge of changes to the internal interface between the nfsd.ko and nfscommon.ko modules, requiring them to be upgraded together (rev r292223).
1002504	r292589	22 de dezembro de 2015	10-STABLE after merge of xz 5.2.2 merge (multithread support) (rev r292588).
1002505	r292908	30 de dezembro de 2015	10-STABLE after merge of changes to pci(4) (rev r292907).

Valor	Revisão	Data	Release
1002506	r293476	9 de janeiro de 2016	10-STABLE after merge of utimensat(2) (rev r293473).
1002507	r293610	9 de janeiro de 2016	10-STABLE after merge of changes to linux(4) (rev r293477 through r293609).
1002508	r293619	9 de janeiro de 2016	10-STABLE after merge of changes to figpar(3) types/macros (rev r290275).
1002509	r295107	1 de fevereiro de 2016	10-STABLE after merge of API change to dpv(3).
1003000	r296373	4 de março de 2016	`releng/10.3` branched from 10-STABLE.
1003500	r296374	4 de março de 2016	10-STABLE after `releng/10.3` branched from 10-STABLE.
1003501	r298299	19 de junho de 2016	10-STABLE after adding kdbcontrol's -P option (rev r298297).
1003502	r299966	19 de junho de 2016	10-STABLE after libcrypto.so was made position independent.
1003503	r300235	19 de junho de 2016	10-STABLE after allowing MK_ overrides (rev r300233).
1003504	r302066	21 de junho de 2016	10-STABLE after MFC of filemon changes from 11-CURRENT.
1003505	r302228	27 de junho de 2016	10-STABLE after converting sed to use REG_STARTEND, fixing a Mesa issue.
1003506	r304611	22 de agosto de 2016	10-STABLE after adding C++11 thread_local support.
1003507	r304864	26 de agosto de 2016	10-STABLE after `LC*MASK` fix.

Valor	Revisão	Data	Release
1003508	r305734	12 de setembro de 2016	10-STABLE after resolving a deadlock between `device_detach()` and usbd_do_request_flags(9).
1003509	r307331	14 de outubro de 2016	10-STABLE after ZFS merges.
1003510	r308047	28 de outubro de 2016	10-STABLE after installing header files required development with libzfs_core.
1003511	r310121	15 de dezembro de 2016	10-STABLE after exporting whole thread name in `kinfo_proc` (rev r309676).
1003512	r315730	22 de março de 2017	10-STABLE after libmd changes (rev r314143).
1003513	r316499	4 de abril de 2017	10-STABLE after making CAM SIM lock optional (revs r315673, r315674).
1003514	r318198	11 de maio de 2017	10-STABLE after merging the addition of the <dev/mmc/mmc_ioctl.h> header.
1003515	r321222	19 de julho de 2017	10-STABLE after adding c++14 sized deallocation functions to libc++.
1003516	r321717	30 de julho de 2017	10-STABLE after merging the `MAP_GUARD` mmap(2) flag addition.
1004000	r323604	15 de setembro de 2017	`releng/10.4` branched from 10-STABLE.
1004500	r323605	15 de setembro de 2017	10-STABLE after `releng/10.4` branched from 10-STABLE.

Valor	Revisão	Data	Release
1004501	r328379	24 de janeiro de 2018	10-STABLE after merging r325028, fixing `ptrace()` to always clear the correct thread event when resuming.
1004502	r356396	6 de janeiro de 2020	10-STABLE after making USB statistics be per-device instead of per bus.
1004503	r356681	13 de janeiro de 2020	10-STABLE after adding own counter for cancelled USB transfers.

18.5. Versões do FreeBSD 9

Tabela 57. Valores do `__FreeBSD_version` para o FreeBSD 9

Valor	Revisão	Data	Release
900000	r196432	22 de agosto de 2009	9.0-CURRENT.
900001	r197019	8 de setembro de 2009	9.0-CURRENT after importing x86emu, a software emulator for real mode x86 CPU from OpenBSD.
900002	r197430	23 de setembro de 2009	9.0-CURRENT after implementing the EVFILT_USER kevent filter functionality.
900003	r200039	2 de dezembro de 2009	9.0-CURRENT after addition of sigpause(2) and PIE support in csu.
900004	r200185	6 de dezembro de 2009	9.0-CURRENT after addition of libulog and its libutempter compatibility interface.
900005	r200447	12 de dezembro de 2009	9.0-CURRENT after addition of sleepq_sleepcnt(9), which can be used to query the number of waiters on a specific waiting queue.

Valor	Revisão	Data	Release
900006	r201513	4 de janeiro de 2010	9.0-CURRENT after change of the scandir(3) and alphasort(3) prototypes to conform to SUSv4.
900007	r202219	13 de janeiro de 2010	9.0-CURRENT after the removal of utmp(5) and the addition of utmpx (see getutxent(3)) for improved logging of user logins and system events.
900008	r202722	20 de janeiro de 2010	9.0-CURRENT after the import of BSDL bc/dc and the deprecation of GNU bc/dc.
900009	r203052	26 de janeiro de 2010	9.0-CURRENT after the addition of SIOCGIFDESCR and SIOCSIFDESCR ioctls to network interfaces. These ioctl can be used to manipulate interface description, as inspired by OpenBSD.
900010	r205471	22 de março de 2010	9.0-CURRENT after the import of zlib 1.2.4.
900011	r207410	24 de abril de 2010	9.0-CURRENT after adding soft-updates journalling.
900012	r207842	10 de maio de 2010	9.0-CURRENT after adding liblzma, xz, xzdec, and lzmainfo.
900013	r208486	24 de maio de 2010	9.0-CURRENT after bringing in USB fixes for linux(4).
900014	r208973	10 de junho de 2010	9.0-CURRENT after adding Clang.
900015	r210390	22 de julho de 2010	9.0-CURRENT after the import of BSD grep.

Valor	Revisão	Data	Release
900016	r210565	28 de julho de 2010	9.0-CURRENT after adding mti_zone to struct malloc_type_internal.
900017	r211701	23 de agosto de 2010	9.0-CURRENT after changing back default grep to GNU grep and adding WITH_BSD_GREP knob.
900018	r211735	24 de agosto de 2010	9.0-CURRENT after the pthread_kill(3) -generated signal is identified as SI_LWP in si_code. Previously, si_code was SI_USER.
900019	r211937	28 de agosto de 2010	9.0-CURRENT after addition of the MAP_PREFAULT_READ flag to mmap(2).
900020	r212381	9 de setembro de 2010	9.0-CURRENT after adding drain functionality to sbufs, which also changed the layout of struct sbuf.
900021	r212568	13 de setembro de 2010	9.0-CURRENT after DTrace has grown support for userland tracing.
900022	r213395	2 de outubro de 2010	9.0-CURRENT after addition of the BSDL man utilities and retirement of GNU/GPL man utilities.
900023	r213700	11 de outubro de 2010	9.0-CURRENT after updating xz to git 20101010 snapshot.
900024	r215127	11 de novembro de 2010	9.0-CURRENT depois que libgcc.a foi trocado por libcompiler_rt.a.

Valor	Revisão	Data	Release
900025	r215166	12 de novembro de 2010	9.0-CURRENT after the introduction of the modularised congestion control.
900026	r216088	30 de novembro de 2010	9.0-CURRENT after the introduction of Serial Management Protocol (SMP) passthrough and the XPT_SMP_IO and XPT_GDEV_ADVINFO CAM CCBs.
900027	r216212	5 de dezembro de 2010	9.0-CURRENT after the addition of log2 to libm.
900028	r216615	21 de dezembro de 2010	9.0-CURRENT after the addition of the Hhook (Helper Hook), Khelp (Kernel Helpers) and Object Specific Data (OSD) KPIs.
900029	r216758	28 de dezembro de 2010	9.0-CURRENT after the modification of the TCP stack to allow Khelp modules to interact with it via helper hook points and store per-connection data in the TCP control block.
900030	r217309	12 de janeiro de 2011	9.0-CURRENT after the update of libdialog to version 20100428.
900031	r218414	7 de fevereiro de 2011	9.0-CURRENT after the addition of pthread_getthreadid_np(3).
900032	r218425	8 de fevereiro de 2011	9.0-CURRENT after the removal of the uio_yield prototype and symbol.
900033	r218822	18 de fevereiro de 2011	9.0-CURRENT after the update of binutils to version 2.17.50.

Valor	Revisão	Data	Release
900034	r219406	8 de março de 2011	9.0-CURRENT after the struct sysvec (sv_schedtail) changes.
900035	r220150	29 de março de 2011	9.0-CURRENT after the update of base gcc and libstdc++ to the last GPLv2 licensed revision.
900036	r220770	18 de abril de 2011	9.0-CURRENT after the removal of libobjc and Objective-C support from the base system.
900037	r221862	13 de maio de 2011	9.0-CURRENT after importing the libprocstat(3) library and fuser(1) utility to the base system.
900038	r222167	22 de maio de 2011	9.0-CURRENT after adding a lock flag argument to VFS_FHTOVP(9).
900039	r223637	28 de junho de 2011	9.0-CURRENT after importing pf from OpenBSD 4.5.
900040	r224217	19 de julho de 2011	Increase default MAXCPU for FreeBSD to 64 on amd64 and ia64 and to 128 for XLP (mips).
900041	r224834	13 de agosto de 2011	9.0-CURRENT after the implementation of Capsicum capabilities; fget(9) gains a rights argument.
900042	r225350	28 de agosto de 2011	Bump shared libraries' version numbers for libraries whose ABI has changed in preparation for 9.0.

Valor	Revisão	Data	Release
900043	r225350	2 de setembro de 2011	Add automatic detection of USB mass storage devices which do not support the no synchronize cache SCSI command.
900044	r225469	10 de setembro de 2011	Re-factor auto-quirk. 9.0-RELEASE.
900045	r229285	2 de janeiro de 2012	9-STABLE after MFC of true/false from 1000002.
900500	r229318	2 de janeiro de 2012	9.0-STABLE.
900501	r229723	6 de janeiro de 2012	9.0-STABLE after merging of addition of the posix_fadvise(2) system call.
900502	r230237	16 de janeiro de 2012	9.0-STABLE after merging gperf 3.0.3
900503	r231768	15 de fevereiro de 2012	9.0-STABLE after introduction of the new extensible sysctl(3) interface NET_RT_IFLISTL to query address lists.
900504	r232728	3 de março de 2012	9.0-STABLE after changes related to mounting of filesystem inside a jail.
900505	r232945	13 de março de 2012	9.0-STABLE after introduction of new tcp(4) socket options: TCP_KEEPINIT, TCP_KEEPIDLE, TCP_KEEPINTVL, and TCP_KEEPCNT.
900506	r235786	22 de maio de 2012	9.0-STABLE after introduction of the `quick_exit` function and related changes required for C++11.
901000	r239082	5 de agosto de 2012	9.1-RELEASE.

Valor	Revisão	Data	Release
901500	r239081	6 de agosto de 2012	9.1-STABLE after branching releng/9.1 (RELENG_9_1).
901501	r240659	11 de novembro de 2012	9.1-STABLE after LIST_PREV(3) added to queue.h (rev r242893) and KBI change in USB serial devices.
901502	r243656	28 de novembro de 2012	9.1-STABLE after USB serial jitter buffer requires rebuild of USB serial device modules.
901503	r247090	21 de fevereiro de 2013	9.1-STABLE after USB moved to the driver structure requiring a rebuild of all USB modules. Also indicates the presence of nmtree.
901504	r248338	15 de março de 2013	9.1-STABLE after install gained -l, -M, -N and related flags and cat gained the -l option.
901505	r251687	13 de junho de 2013	9.1-STABLE after fixes in ctfmerge bootstrapping (rev r249243).
902001	r253912	3 de agosto de 2013	`releng/9.2` branched from `stable/9`.
902501	r253913	2 de agosto de 2013	9.2-STABLE after creation of `releng/9.2` branch.
902502	r254938	26 de agosto de 2013	9.2-STABLE after inclusion of the `PIM_RESCAN` CAM path inquiry flag.
902503	r254979	27 de agosto de 2013	9.2-STABLE after inclusion of the `SI_UNMAPPED` cdev flag.
902504	r256917	22 de outubro de 2013	9.2-STABLE after inclusion of support for "first boot"rc(8) scripts.

Valor	Revisão	Data	Release
902505	r259448	12 de dezembro de 2013	9.2-STABLE after Heimdal encoding fix.
902506	r260136	31 de dezembro de 2013	9-STABLE after MAP_STACK fixes (rev r260082).
902507	r262801	5 de março de 2014	9-STABLE after upgrade of libc++ to 3.4 release.
902508	r263171	14 de março de 2014	9-STABLE after merge of the Radeon KMS driver (rev r263170).
902509	r263509	21 de março de 2014	9-STABLE after upgrade of llvm/clang to 3.4 release.
902510	r263818	27 de março de 2014	9-STABLE after merge of the vt(4) driver.
902511	r264289	27 de março de 2014	9-STABLE after FreeBSD-SA-14:06.openssl.
902512	r265123	30 de abril de 2014	9-STABLE after FreeBSD-SA-14:08.tcp.
903000	r267656	20 de junho de 2014	9-RC1 `releng/9.3` branch.
903500	r267657	20 de junho de 2014	9.3-STABLE `releng/9.3` branch.
903501	r268443	8 de julho de 2014	9-STABLE after FreeBSD-SA-14:17.kmem (rev r268433).
903502	r270175	19 de agosto de 2014	9-STABLE after `SOCK_DGRAM` bug fix (rev r269789).
903503	r271341	9 de setembro de 2014	9-STABLE after FreeBSD-SA-14:18 (rev r269687).
903504	r271686	16 de setembro de 2014	9-STABLE after FreeBSD-SA-14:19 (rev r271668).

Valor	Revisão	Data	Release
903505	r273432	21 de outubro de 2014	9-STABLE after FreeBSD-SA-14:20, FreeBSD-SA-14:21, and FreeBSD-SA-14:22 (rev r273412).
903506	r274162	4 de novembro de 2014	9-STABLE after FreeBSD-SA-14:23, FreeBSD-SA-14:24, and FreeBSD-SA-14:25.
903507	r275742	13 de dezembro de 2014	9-STABLE after merging an important fix to the LLVM vectorizer, which could lead to buffer overruns in some cases.
903508	r279287	25 de fevereiro de 2015	9-STABLE after FreeBSD-EN-15:01.vt, FreeBSD-EN-15:02.openssl, FreeBSD-EN-15:03.freebsd-update, FreeBSD-SA-15:04.igmp, and FreeBSD-SA-15:05.bind.
903509	r296219	29 de fevereiro de 2016	9-STABLE after bumping the default value of `compat.linux.osrelease` to `2.6.18` to support the linux-c6-* ports out of the box.
903510	r300236	19 de maio de 2016	9-STABLE after System Binary Interface (SBI) page was moved in latest version of Berkeley Boot Loader (BBL) due to code size increase in r300234.
903511	r305735	12 de setembro de 2016	9-STABLE after resolving a deadlock between `device_detach()` and usbd_do_request_flags(9).

18.6. Versões do FreeBSD 8

Tabela 58. Valores do `__FreeBSD_version` para o FreeBSD 8

Valor	Revisão	Data	Release
800000	r172531	11 de outubro de 2007	8.0-CURRENT. Separating wide and single byte ctype.
800001	r172688	16 de outubro de 2007	8.0-CURRENT after libpcap 0.9.8 and tcpdump 3.9.8 import.
800002	r172841	21 de outubro de 2007	8.0-CURRENT after renaming kthread_create(9) and friends to kproc_create(9) etc.
800003	r172932	24 de outubro de 2007	8.0-CURRENT after ABI backwards compatibility to the FreeBSD 4/5/6 versions of the PCIOCGETCONF, PCIOCREAD and PCIOCWRITE IOCTLs was added, which required the ABI of the PCIOCGETCONF IOCTL to be broken again
800004	r173573	12 de novembro de 2007	8.0-CURRENT after agp(4) driver moved from src/sys/pci to src/sys/dev/agp
800005	r174261	4 de dezembro de 2007	8.0-CURRENT after changes to the jumbo frame allocator (rev r174247).
800006	r174399	7 de dezembro de 2007	8.0-CURRENT after the addition of callgraph capture functionality to hwpmc(4).
800007	r174901	25 de dezembro de 2007	8.0-CURRENT after `kdb_enter()` gains a "why" argument.

Valor	Revisão	Data	Release
800008	r174951	28 de dezembro de 2007	8.0-CURRENT after LK_EXCLUPGRADE option removal.
800009	r175168	9 de janeiro de 2008	8.0-CURRENT after introduction of lockmgr_disown(9)
800010	r175204	10 de janeiro de 2008	8.0-CURRENT after the vn_lock(9) prototype change.
800011	r175295	13 de janeiro de 2008	8.0-CURRENT after the VOP_LOCK(9) and VOP_UNLOCK(9) prototype changes.
800012	r175487	19 de janeiro de 2008	8.0-CURRENT after introduction of lockmgr_recursed(9), BUF_RECURSED(9) and BUF_ISLOCKED(9) and the removal of `BUF_REFCNT()`.
800013	r175581	23 de janeiro de 2008	8.0-CURRENT after introduction of the "ASCII" encoding.
800014	r175636	24 de janeiro de 2008	8.0-CURRENT after changing the prototype of lockmgr(9) and removal of `lockcount()` and `LOCKMGR_ASSERT()`.
800015	r175688	26 de janeiro de 2008	8.0-CURRENT after extending the types of the fts(3) structures.
800016	r175872	1 de fevereiro de 2008	8.0-CURRENT after adding an argument to MEXTADD(9)
800017	r176015	6 de fevereiro de 2008	8.0-CURRENT after the introduction of LK_NODUP and LK_NOWITNESS options in the lockmgr(9) space.
800018	r176112	8 de fevereiro de 2008	8.0-CURRENT after the addition of m_collapse.

Valor	Revisão	Data	Release
800019	r176124	9 de fevereiro de 2008	8.0-CURRENT after the addition of current working directory, root directory, and jail directory support to the kern.proc.filedesc sysctl.
800020	r176251	13 de fevereiro de 2008	8.0-CURRENT after introduction of lockmgr_assert(9) and `BUF_ASSERT` functions.
800021	r176321	15 de fevereiro de 2008	8.0-CURRENT after introduction of lockmgr_args(9) and LK_INTERNAL flag removal.
800022	r176556	(backed out)	8.0-CURRENT after changing the default system ar to BSD ar(1).
800023	r176560	25 de fevereiro de 2008	8.0-CURRENT after changing the prototypes of lockstatus(9) and VOP_ISLOCKED(9), more specifically retiring the `struct thread` argument.
800024	r176709	1 de março de 2008	8.0-CURRENT after axing out the `lockwaiters` and `BUF_LOCKWAITERS` functions, changing the return value of `brelvp` from void to int and introducing new flags for lockinit(9).
800025	r176958	8 de março de 2008	8.0-CURRENT after adding F_DUP2FD command to fcntl(2).

Valor	Revisão	Data	Release
800026	r177086	12 de março de 2008	8.0-CURRENT after changing the priority parameter to cv_broadcastpri such that 0 means no priority.
800027	r177551	24 de março de 2008	8.0-CURRENT after changing the bpf monitoring ABI when zerocopy bpf buffers were added.
800028	r177637	26 de março de 2008	8.0-CURRENT after adding l_sysid to struct flock.
800029	r177688	28 de março de 2008	8.0-CURRENT after reintegration of the `BUF_LOCKWAITERS` function and the addition of lockmgr_waiters(9).
800030	r177844	1 de abril de 2008	8.0-CURRENT after the introduction of the rw_try_rlock(9) and rw_try_wlock(9) functions.
800031	r177958	6 de abril de 2008	8.0-CURRENT after the introduction of the `lockmgr_rw` and `lockmgr_args_rw` functions.
800032	r178006	8 de abril de 2008	8.0-CURRENT after the implementation of the openat and related syscalls, introduction of the O_EXEC flag for the open(2), and providing the corresponding linux compatibility syscalls.

Valor	Revisão	Data	Release
800033	r178017	8 de abril de 2008	8.0-CURRENT after added write(2) support for psm(4) in native operation level. Now arbitrary commands can be written to /dev/psm%d and status can be read back from it.
800034	r178051	10 de abril de 2008	8.0-CURRENT after introduction of the `memrchr` function.
800035	r178256	16 de abril de 2008	8.0-CURRENT after introduction of the `fdopendir` function.
800036	r178362	20 de abril de 2008	8.0-CURRENT after switchover of 802.11 wireless to multi-bss support (aka vaps).
800037	r178892	9 de maio de 2008	8.0-CURRENT after addition of multi routing table support (aka setfib(1), setfib(2)).
800038	r179316	26 de maio de 2008	8.0-CURRENT after removal of netatm and ISDN4BSD. Also, the addition of the Compact C Type (CTF) tools.
800039	r179784	14 de junho de 2008	8.0-CURRENT after removal of sgtty.
800040	r180025	26 de junho de 2008	8.0-CURRENT with kernel NFS lockd client.
800041	r180691	22 de julho de 2008	8.0-CURRENT after addition of arc4random_buf(3) and arc4random_uniform(3).
800042	r181439	8 de agosto de 2008	8.0-CURRENT after addition of cpuctl(4).

Valor	Revisão	Data	Release
800043	r181694	13 de agosto de 2008	8.0-CURRENT after changing bpf(4) to use a single device node, instead of device cloning.
800044	r181803	17 de agosto de 2008	8.0-CURRENT after the commit of the first step of the vimage project renaming global variables to be virtualized with a V_ prefix with macros to map them back to their global names.
800045	r181905	20 de agosto de 2008	8.0-CURRENT after the integration of the MPSAFE TTY layer, including changes to various drivers and utilities that interact with it.
800046	r182869	8 de setembro de 2008	8.0-CURRENT after the separation of the GDT per CPU on amd64 architecture.
800047	r182905	10 de setembro de 2008	8.0-CURRENT after removal of VSVTX, VSGID and VSUID.
800048	r183091	16 de setembro de 2008	8.0-CURRENT after converting the kernel NFS mount code to accept individual mount options in the nmount(2) iovec, not just one big struct nfs_args.
800049	r183114	17 de setembro de 2008	8.0-CURRENT after the removal of suser(9) and suser_cred(9).
800050	r184099	20 de outubro de 2008	8.0-CURRENT after buffer cache API change.

Valor	Revisão	Data	Release
800051	r184205	23 de outubro de 2008	8.0-CURRENT after the removal of the MALLOC(9) and FREE(9) macros.
800052	r184419	28 de outubro de 2008	8.0-CURRENT after the introduction of accmode_t and renaming of VOP_ACCESS 'a_mode' argument to 'a_accmode'.
800053	r184555	2 de novembro de 2008	8.0-CURRENT after the prototype change of vfs_busy(9) and the introduction of its MBF_NOWAIT and MBF_MNTLSTLOCK flags.
800054	r185162	22 de novembro de 2008	8.0-CURRENT after the addition of buf_ring, memory barriers and ifnet functions to facilitate multiple hardware transmit queues for cards that support them, and a lockless ring-buffer implementation to enable drivers to more efficiently manage queuing of packets.
800055	r185363	27 de novembro de 2008	8.0-CURRENT after the addition of Intel™ Core, Core2, and Atom support to hwpmc(4).
800056	r185435	29 de novembro de 2008	8.0-CURRENT after the introduction of multi-/no-IPv4/v6 jails.
800057	r185522	1 de dezembro de 2008	8.0-CURRENT after the switch to the ath hal source code.

Valor	Revisão	Data	Release
800058	r185968	12 de dezembro de 2008	8.0-CURRENT after the introduction of the VOP_VPTOCNP operation.
800059	r186119	15 de dezembro de 2008	8.0-CURRENT incorporates the new arp-v2 rewrite.
800060	r186344	19 de dezembro de 2008	8.0-CURRENT after the addition of makefs.
800061	r187289	15 de janeiro de 2009	8.0-CURRENT after TCP Appropriate Byte Counting.
800062	r187830	28 de janeiro de 2009	8.0-CURRENT after removal of minor(), minor2unit(), unit2minor(), etc.
800063	r188745	18 de fevereiro de 2009	8.0-CURRENT after GENERIC config change to use the USB2 stack, but also the addition of fdevname(3).
800064	r188946	23 de fevereiro de 2009	8.0-CURRENT after the USB2 stack is moved to and replaces dev/usb.
800065	r189092	26 de fevereiro de 2009	8.0-CURRENT after the renaming of all functions in libmp(3).
800066	r189110	27 de fevereiro de 2009	8.0-CURRENT after changing USB devfs handling and layout.
800067	r189136	28 de fevereiro de 2009	8.0-CURRENT after adding getdelim(), getline(), stpncpy(), strnlen(), wcsnlen(), wcscasecmp(), and wcsncasecmp().
800068	r189276	2 de março de 2009	8.0-CURRENT after renaming the ushub devclass to uhub.
800069	r189585	9 de março de 2009	8.0-CURRENT after libusb20.so.1 was renamed to libusb.so.1.

Valor	Revisão	Data	Release
800070	r189592	9 de março de 2009	8.0-CURRENT after merging IGMPv3 and Source-Specific Multicast (SSM) to the IPv4 stack.
800071	r189825	14 de março de 2009	8.0-CURRENT after gcc was patched to use C99 inline semantics in c99 and gnu99 mode.
800072	r189853	15 de março de 2009	8.0-CURRENT after the IFF_NEEDSGIANT flag has been removed; non-MPSAFE network device drivers are no longer supported.
800073	r190265	18 de março de 2009	8.0-CURRENT after the dynamic string token substitution has been implemented for rpath and needed paths.
800074	r190373	24 de março de 2009	8.0-CURRENT after tcpdump 4.0.0 and libpcap 1.0.0 import.
800075	r190787	6 de abril de 2009	8.0-CURRENT after layout of structs vnet_net, vnet_inet and vnet_ipfw has been changed.
800076	r190866	9 de abril de 2009	8.0-CURRENT after adding delay profiles in dummynet.
800077	r190914	14 de abril de 2009	8.0-CURRENT after removing VOP_LEASE() and vop_vector.vop_lease.

Valor	Revisão	Data	Release
800078	r191080	15 de abril de 2009	8.0-CURRENT after struct rt_weight fields have been added to struct rt_metrics and struct rt_metrics_lite, changing the layout of struct rt_metrics_lite. A bump to RTM_VERSION was made, but backed out.
800079	r191117	15 de abril de 2009	8.0-CURRENT after struct llentry pointers are added to struct route and struct route_in6.
800080	r191126	15 de abril de 2009	8.0-CURRENT after layout of struct inpcb has been changed.
800081	r191267	19 de abril de 2009	8.0-CURRENT after the layout of struct malloc_type has been changed.
800082	r191368	21 de abril de 2009	8.0-CURRENT after the layout of struct ifnet has changed, and with if_ref() and if_rele() ifnet refcounting.
800083	r191389	22 de abril de 2009	8.0-CURRENT after the implementation of a low-level Bluetooth HCI API.
800084	r191672	29 de abril de 2009	8.0-CURRENT after IPv6 SSM and MLDv2 changes.
800085	r191688	30 de abril de 2009	8.0-CURRENT after enabling support for VIMAGE kernel builds with one active image.
800086	r191910	8 de maio de 2009	8.0-CURRENT after adding support for input lines of arbitrarily length in patch(1).

Valor	Revisão	Data	Release
800087	r191990	11 de maio de 2009	8.0-CURRENT after some VFS KPI changes. The thread argument has been removed from the FSD parts of the VFS. `VFS_*` functions do not need the context any more because it always refers to `curthread`. In some special cases, the old behavior is retained.
800088	r192470	20 de maio de 2009	8.0-CURRENT after net80211 monitor mode changes.
800089	r192649	23 de maio de 2009	8.0-CURRENT after adding UDP control block support.
800090	r192669	23 de maio de 2009	8.0-CURRENT after virtualizing interface cloning.
800091	r192895	27 de maio de 2009	8.0-CURRENT after adding hierarchical jails and removing global securelevel.
800092	r193011	29 de maio de 2009	8.0-CURRENT after changing `sx_init_flags()` KPI. The SX_ADAPTIVESPIN is retired and a new SX_NOADAPTIVE flag is introduced to handle the reversed logic.
800093	r193047	29 de maio de 2009	8.0-CURRENT after adding mnt_xflag to struct mount.
800094	r193093	30 de maio de 2009	8.0-CURRENT after adding VOP_ACCESSX(9).

Valor	Revisão	Data	Release
800095	r193096	30 de maio de 2009	8.0-CURRENT after changing the polling KPI. The polling handlers now return the number of packets processed. A new IFCAP_POLLING_NOCOUNT is also introduced to specify that the return value is not significant and the counting should be skipped.
800096	r193219	1 de junho de 2009	8.0-CURRENT after updating to the new netisr implementation and after changing the way we store and access FIBs.
800097	r193731	8 de junho de 2009	8.0-CURRENT after the introduction of vnet destructor hooks and infrastructure.
(Não mudou)	r194012	11 de junho de 2009	8.0-CURRENT after the introduction of netgraph outbound to inbound path call detection and queuing, which also changed the layout of struct thread.
800098	r194210	14 de junho de 2009	8.0-CURRENT after OpenSSL 0.9.8k import.
800099	r194675	22 de junho de 2009	8.0-CURRENT after NGROUPS update and moving route virtualization into its own VImage module.
800100	r194920	24 de junho de 2009	8.0-CURRENT after SYSVIPC ABI change.
800101	r195175	29 de junho de 2009	8.0-CURRENT after the removal of the /dev/net/* per-interface character devices.

Valor	Revisão	Data	Release
800102	r195634	12 de julho de 2009	8.0-CURRENT after padding was added to struct sackhint, struct tcpcb, and struct tcpstat.
800103	r195654	13 de julho de 2009	8.0-CURRENT after replacing struct tcpopt with struct toeopt in the TOE driver interface to the TCP syncache.
800104	r195699	14 de julho de 2009	8.0-CURRENT after the addition of the linker-set based per-vnet allocator.
800105	r195767	19 de julho de 2009	8.0-CURRENT after version bump for all shared libraries that do not have symbol versioning turned on.
800106	r195852	24 de julho de 2009	8.0-CURRENT after introduction of OBJT_SG VM object type.
800107	r196037	2 de agosto de 2009	8.0-CURRENT after making the newbus subsystem Giant free by adding the newbus sxlock and 8.0-RELEASE.
800108	r199627	21 de novembro de 2009	8.0-STABLE after implementing EVFILT_USER kevent filter.
800500	r201749	7 de janeiro de 2010	8.0-STABLE after `__FreeBSD_version` bump to make `pkg_add -r` use packages-8-stable.

Valor	Revisão	Data	Release
800501	r202922	24 de janeiro de 2010	8.0-STABLE after change of the scandir(3) and alphasort(3) prototypes to conform to SUSv4.
800502	r203299	31 de janeiro de 2010	8.0-STABLE after addition of sigpause(2).
800503	r204344	25 de fevereiro de 2010	8.0-STABLE after addition of SIOCGIFDESCR and SIOCSIFDESCR ioctls to network interfaces. These ioctl can be used to manipulate interface description, as inspired by OpenBSD.
800504	r204546	1 de março de 2010	8.0-STABLE after MFC of importing x86emu, a software emulator for real mode x86 CPU from OpenBSD.
800505	r208259	18 de maio de 2010	8.0-STABLE after MFC of adding liblzma, xz, xzdec, and lzmainfo.
801000	r209150	14 de junho de 2010	8.1-RELEASE
801500	r209146	14 de junho de 2010	8.1-STABLE after 8.1-RELEASE.
801501	r214762	3 de novembro de 2010	8.1-STABLE after KBI change in struct sysentvec, and implementation of PL_FLAG_SCE/SCX/EXEC/SI and pl_siginfo for ptrace(PT_LWPINFO) .
802000	r216639	22 de dezembro de 2010	8.2-RELEASE
802500	r216654	22 de dezembro de 2010	8.2-STABLE after 8.2-RELEASE.

Valor	Revisão	Data	Release
802501	r219107	28 de fevereiro de 2011	8.2-STABLE after merging DTrace changes, including support for userland tracing.
802502	r219324	6 de março de 2011	8.2-STABLE after merging log2 and log2f into libm.
802503	r221275	1 de maio de 2011	8.2-STABLE after upgrade of the gcc to the last GPLv2 version from the FSF gcc-4_2-branch.
802504	r222401	28 de maio de 2011	8.2-STABLE after introduction of the KPI and supporting infrastructure for modular congestion control.
802505	r222406	28 de maio de 2011	8.2-STABLE after introduction of Hhook and Khelp KPIs.
802506	r222408	28 de maio de 2011	8.2-STABLE after addition of OSD to struct tcpcb.
802507	r222741	6 de junho de 2011	8.2-STABLE after ZFS v28 import.
802508	r222846	8 de junho de 2011	8.2-STABLE after removal of the schedtail event handler and addition of the sv_schedtail method to struct sysvec.
802509	r224017	14 de julho de 2011	8.2-STABLE after merging the SSSE3 support into binutils.
802510	r224214	19 de julho de 2011	8.2-STABLE after addition of RFTSIGZMB flag for rfork(2).

Valor	Revisão	Data	Release
802511	r225458	9 de setembro de 2011	8.2-STABLE after addition of automatic detection of USB mass storage devices which do not support the no synchronize cache SCSI command.
802512	r225470	10 de setembro de 2011	8.2-STABLE after merging of re-factoring of auto-quirk.
802513	r226763	25 de outubro de 2011	8.2-STABLE after merging of the MAP_PREFAULT_READ flag to mmap(2).
802514	r227573	16 de novembro de 2011	8.2-STABLE after merging of addition of posix_fallocate(2) syscall.
802515	r229725	6 de janeiro de 2012	8.2-STABLE after merging of addition of the posix_fadvise(2) system call.
802516	r230239	16 de janeiro de 2012	8.2-STABLE after merging gperf 3.0.3
802517	r231769	15 de fevereiro de 2012	8.2-STABLE after introduction of the new extensible sysctl(3) interface NET_RT_IFLISTL to query address lists.
803000	r232446	3 de março de 2012	8.3-RELEASE.
803500	r232439	3 de março de 2012	8.3-STABLE after branching releng/8.3 (RELENG_8_3).
803501	r247091	21 de fevereiro de 2013	8.3-STABLE after MFC of two USB fixes (rev r246616 and r246759).
804000	r248850	28 de março de 2013	8.4-RELEASE.
804500	r248819	28 de março de 2013	8.4-STABLE after 8.4-RELEASE.

Valor	Revisão	Data	Release
804501	r259449	16 de dezembro de 2013	8.4-STABLE after MFC of upstream Heimdal encoding fix.
804502	r265123	30 de abril de 2014	8.4-STABLE after FreeBSD-SA-14:08.tcp.
804503	r268444	9 de julho de 2014	8.4-STABLE after FreeBSD-SA-14:17.kmem.
804504	r271341	9 de setembro de 2014	8.4-STABLE after FreeBSD-SA-14:18 (rev r271305).
804505	r271686	16 de setembro de 2014	8.4-STABLE after FreeBSD-SA-14:19 (rev r271668).
804506	r273432	21 de outubro de 2014	8.4-STABLE depois de FreeBSD-SA-14:21 (rev r273413).
804507	r274162	4 de novembro de 2014	8.4-STABLE after FreeBSD-SA-14:23, FreeBSD-SA-14:24, and FreeBSD-SA-14:25.
804508	r279287	25 de fevereiro de 2015	8-STABLE after FreeBSD-EN-15:01.vt, FreeBSD-EN-15:02.openssl, FreeBSD-EN-15:03.freebsd-update, FreeBSD-SA-15:04.igmp, and FreeBSD-SA-15:05.bind.
804509	r305736	12 de setembro de 2016	8-STABLE after resolving a deadlock between `device_detach()` and usbd_do_request_flags(9).

18.7. Versões do FreeBSD 7

Tabela 59. Valores do `__FreeBSD_version` para o FreeBSD 7

Valor	Revisão	Data	Release
700000	r147925	11 de julho de 2005	7.0-CURRENT.

Valor	Revisão	Data	Release
700001	r148341	23 de julho de 2005	7.0-CURRENT after bump of all shared library versions that had not been changed since RELENG_5.
700002	r149039	13 de agosto de 2005	7.0-CURRENT after credential argument is added to dev_clone event handler.
700003	r149470	25 de agosto de 2005	7.0-CURRENT after memmem(3) is added to libc.
700004	r151888	30 de outubro de 2005	7.0-CURRENT after solisten(9) kernel arguments are modified to accept a backlog parameter.
700005	r152296	11 de novembro de 2005	7.0-CURRENT after IFP2ENADDR() was changed to return a pointer to IF_LLADDR().
700006	r152315	11 de novembro de 2005	7.0-CURRENT after addition of `if_addr` member to `struct ifnet` and IFP2ENADDR() removal.
700007	r153027	2 de dezembro de 2005	7.0-CURRENT after incorporating scripts from the local_startup directories into the base rcorder(8).
700008	r153107	5 de dezembro de 2005	7.0-CURRENT after removal of MNT_NODEV mount option.
700009	r153519	19 de dezembro de 2005	7.0-CURRENT after ELF-64 type changes and symbol versioning.

Valor	Revisão	Data	Release
700010	r153579	20 de dezembro de 2005	7.0-CURRENT after addition of hostb and vgapci drivers, addition of pci_find_extcap(), and changing the AGP drivers to no longer map the aperture.
700011	r153936	31 de dezembro de 2005	7.0-CURRENT after tv_sec was made time_t on all platforms but Alpha.
700012	r154114	8 de janeiro de 2006	7.0-CURRENT after ldconfig_local_dirs change.
700013	r154269	12 de janeiro de 2006	7.0-CURRENT after changes to /etc/rc.d/abi to support /compat/linux/etc/ld.so.cache being a symlink in a readonly filesystem.
700014	r154863	26 de janeiro de 2006	7.0-CURRENT after pts import.
700015	r157144	26 de março de 2006	7.0-CURRENT after the introduction of version 2 of hwpmc(4)'s ABI.
700016	r157962	22 de abril de 2006	7.0-CURRENT after addition of fcloseall(3) to libc.
700017	r158513	13 de maio de 2006	7.0-CURRENT after removal of ip6fw.
700018	r160386	15 de julho de 2006	7.0-CURRENT after import of snd_emu10kx.
700019	r160821	29 de julho de 2006	7.0-CURRENT after import of OpenSSL 0.9.8b.
700020	r161931	3 de setembro de 2006	7.0-CURRENT after addition of bus_dma_get_tag function

Valor	Revisão	Data	Release
700021	r162023	4 de setembro de 2006	7.0-CURRENT after libpcap 0.9.4 and tcpdump 3.9.4 import.
700022	r162170	9 de setembro de 2006	7.0-CURRENT after dlsym change to look for a requested symbol both in specified dso and its implicit dependencies.
700023	r162588	23 de setembro de 2006	7.0-CURRENT after adding new sound IOCTLs for the OSSv4 mixer API.
700024	r162919	28 de setembro de 2006	7.0-CURRENT after import of OpenSSL 0.9.8d.
700025	r164190	11 de novembro de 2006	7.0-CURRENT after the addition of libelf.
700026	r164614	26 de novembro de 2006	7.0-CURRENT after major changes on sound sysctls.
700027	r164770	30 de novembro de 2006	7.0-CURRENT after the addition of Wi-Spy quirk.
700028	r165242	15 de dezembro de 2006	7.0-CURRENT after the addition of sctp calls to libc
700029	r166259	26 de janeiro de 2007	7.0-CURRENT after the GNU gzip(1) implementation was replaced with a BSD licensed version ported from NetBSD.
700030	r166549	7 de fevereiro de 2007	7.0-CURRENT after the removal of IPIP tunnel encapsulation (VIFF_TUNNEL) from the IPv4 multicast forwarding code.

Valor	Revisão	Data	Release
700031	r166907	23 de fevereiro de 2007	7.0-CURRENT after the modification of bus_setup_intr() (newbus).
700032	r167165	2 de março de 2007	7.0-CURRENT after the inclusion of ipw(4) and iwi(4) firmware.
700033	r167360	9 de março de 2007	7.0-CURRENT after the inclusion of ncurses wide character support.
700034	r167684	19 de março de 2007	7.0-CURRENT after changes to how insmntque(), getnewvnode(), and vfs_hash_insert() work.
700035	r167906	26 de março de 2007	7.0-CURRENT after addition of a notify mechanism for CPU frequency changes.
700036	r168413	6 de abril de 2007	7.0-CURRENT after import of the ZFS filesystem.
700037	r168504	8 de abril de 2007	7.0-CURRENT after addition of CAM 'SG' peripheral device, which implements a subset of Linux SCSI SG passthrough device API.
700038	r169151	30 de abril de 2007	7.0-CURRENT after changing getenv(3), putenv(3), setenv(3) and unsetenv(3) to be POSIX conformant.
700039	r169190	1 de maio de 2007	7.0-CURRENT after the changes in 700038 were backed out.
700040	r169453	10 de maio de 2007	7.0-CURRENT after the addition of flopen(3) to libutil.

Valor	Revisão	Data	Release
700041	r169526	13 de maio de 2007	7.0-CURRENT after enabling symbol versioning, and changing the default thread library to libthr.
700042	r169758	19 de maio de 2007	7.0-CURRENT after the import of gcc 4.2.0.
700043	r169830	21 de maio de 2007	7.0-CURRENT after bump of all shared library versions that had not been changed since RELENG_6.
700044	r170395	7 de junho de 2007	7.0-CURRENT after changing the argument for vn_open()/VOP_OPEN() from file descriptor index to the struct file *.
700045	r170510	10 de junho de 2007	7.0-CURRENT after changing pam_nologin(8) to provide an account management function instead of an authentication function to the PAM framework.
700046	r170530	11 de junho de 2007	7.0-CURRENT after updated 802.11 wireless support.
700047	r170579	11 de junho de 2007	7.0-CURRENT after adding TCP LRO interface capabilities.
700048	r170613	12 de junho de 2007	7.0-CURRENT after RFC 3678 API support added to the IPv4 stack. Legacy RFC 1724 behavior of the IP_MULTICAST_IF ioctl has now been removed; 0.0.0.0/8 may no longer be used to specify an interface index. Use struct ipmreqn instead.

Valor	Revisão	Data	Release
700049	r171175	3 de julho de 2007	7.0-CURRENT after importing pf from OpenBSD 4.1
(Não mudou)	r171167		7.0-CURRENT after adding IPv6 support for FAST_IPSEC, deleting KAME IPSEC, and renaming FAST_IPSEC to IPSEC.
700050	r171195	4 de julho de 2007	7.0-CURRENT after converting setenv/putenv/etc. calls from traditional BSD to POSIX.
700051	r171211	4 de julho de 2007	7.0-CURRENT after adding new mmap/lseek/etc syscalls.
700052	r171275	6 de julho de 2007	7.0-CURRENT after moving I4B headers to include/i4b.
700053	r172394	30 de setembro de 2007	7.0-CURRENT after the addition of support for PCI domains
700054	r172988	25 de outubro de 2007	7.0-STABLE after MFC of wide and single byte ctype separation.
700055	r173104	28 de outubro de 2007	7.0-RELEASE, and 7.0-CURRENT after ABI backwards compatibility to the FreeBSD 4/5/6 versions of the PCIOCGETCONF, PCIOCREAD and PCIOCWRITE IOCTLs was MFCed, which required the ABI of the PCIOCGETCONF IOCTL to be broken again
700100	r174864	22 de dezembro de 2007	7.0-STABLE after 7.0-RELEASE

Valor	Revisão	Data	Release
700101	r176111	8 de fevereiro de 2008	7.0-STABLE after the MFC of m_collapse().
700102	r177735	30 de março de 2008	7.0-STABLE after the MFC of kdb_enter_why().
700103	r178061	10 de abril de 2008	7.0-STABLE after adding l_sysid to struct flock.
700104	r178108	11 de abril de 2008	7.0-STABLE after the MFC of procstat(1).
700105	r178120	11 de abril de 2008	7.0-STABLE after the MFC of umtx features.
700106	r178225	15 de abril de 2008	7.0-STABLE after the MFC of write(2) support to psm(4).
700107	r178353	20 de abril de 2008	7.0-STABLE after the MFC of F_DUP2FD command to fcntl(2).
700108	r178783	5 de maio de 2008	7.0-STABLE after some lockmgr(9) changes, which makes it necessary to include sys/lock.h to use lockmgr(9).
700109	r179367	27 de maio de 2008	7.0-STABLE after MFC of the memrchr(3) function.
700110	r181328	5 de agosto de 2008	7.0-STABLE after MFC of kernel NFS lockd client.
700111	r181940	20 de agosto de 2008	7.0-STABLE after addition of physically contiguous jumbo frame support.
700112	r182294	27 de agosto de 2008	7.0-STABLE after MFC of kernel DTrace support.
701000	r185315	25 de novembro de 2008	7.1-RELEASE
701100	r185302	25 de novembro de 2008	7.1-STABLE after 7.1-RELEASE.

Valor	Revisão	Data	Release
701101	r187023	10 de janeiro de 2009	7.1-STABLE after strndup(3) merge.
701102	r187370	17 de janeiro de 2009	7.1-STABLE after cpuctl(4) support added.
701103	r188281	7 de fevereiro de 2009	7.1-STABLE after the merge of multi-/no-IPv4/v6 jails.
701104	r188625	14 de fevereiro de 2009	7.1-STABLE after the store of the suspension owner in the struct mount, and introduction of vfs_susp_clean method into the struct vfsops.
701105	r189740	12 de março de 2009	7.1-STABLE after the incompatible change to the kern.ipc.shmsegs sysctl to allow allocating larger SysV shared memory segments on 64bit architectures.
701106	r189786	14 de março de 2009	7.1-STABLE after the merge of a fix for POSIX semaphore wait operations.
702000	r191099	15 de abril de 2009	7.2-RELEASE
702100	r191091	15 de abril de 2009	7.2-STABLE after 7.2-RELEASE.
702101	r192149	15 de maio de 2009	7.2-STABLE after ichsmb(4) was changed to use left-adjusted slave addressing to match other SMBus controller drivers.
702102	r193020	28 de maio de 2009	7.2-STABLE after MFC of the fdopendir(3) function.
702103	r193638	6 de junho de 2009	7.2-STABLE after MFC of PmcTools.

Valor	Revisão	Data	Release
702104	r195694	14 de julho de 2009	7.2-STABLE after MFC of the closefrom(2) system call.
702105	r196006	31 de julho de 2009	7.2-STABLE after MFC of the SYSVIPC ABI change.
702106	r197198	14 de setembro de 2009	7.2-STABLE after MFC of the x86 PAT enhancements and addition of d_mmap_single() and the scatter/gather list VM object type.
703000	r203740	9 de fevereiro de 2010	7.3-RELEASE
703100	r203742	9 de fevereiro de 2010	7.3-STABLE after 7.3-RELEASE.
704000	r216647	22 de dezembro de 2010	7.4-RELEASE
704100	r216658	22 de dezembro de 2010	7.4-STABLE after 7.4-RELEASE.
704101	r221318	2 de maio de 2011	7.4-STABLE after the gcc MFC in rev r221317.

18.8. Versões do FreeBSD 6

Tabela 60. Valores do `__FreeBSD_version` para o FreeBSD 6

Valor	Revisão	Data	Release
600000	r133921	18 de agosto de 2004	6.0-CURRENT
600001	r134396	27 de agosto de 2004	6.0-CURRENT after permanently enabling PFIL_HOOKS in the kernel.
600002	r134514	30 de agosto de 2004	6.0-CURRENT after initial addition of ifi_epoch to struct if_data. Backed out after a few days. Do not use this value.

Valor	Revisão	Data	Release
600003	r134933	8 de setembro de 2004	6.0-CURRENT after the re-addition of the ifi_epoch member of struct if_data.
600004	r135920	29 de setembro de 2004	6.0-CURRENT after addition of the struct inpcb argument to the pfil API.
600005	r136172	5 de outubro de 2004	6.0-CURRENT after addition of the "-d DESTDIR" argument to newsyslog.
600006	r137192	4 de novembro de 2004	6.0-CURRENT after addition of glibc style strftime(3) padding options.
600007	r138760	12 de dezembro de 2004	6.0-CURRENT after addition of 802.11 framework updates.
600008	r140809	25 de janeiro de 2005	6.0-CURRENT after changes to VOP_*VOBJECT() functions and introduction of MNTK_MPSAFE flag for Giantfree filesystems.
600009	r141250	4 de fevereiro de 2005	6.0-CURRENT after addition of the cpufreq framework and drivers.
600010	r141394	6 de fevereiro de 2005	6.0-CURRENT after importing OpenBSD's nc(1).
600011	r141727	12 de fevereiro de 2005	6.0-CURRENT after removing semblance of SVID2 `matherr()` support.
600012	r141940	15 de fevereiro de 2005	6.0-CURRENT after increase of default thread stacks' size.

Valor	Revisão	Data	Release
600013	r142089	19 de fevereiro de 2005	6.0-CURRENT after fixes in <src/include/stdbool.h> and <src/sys/i386/include/_types.h> for using the GCC-compatibility of the Intel C/C++ compiler.
600014	r142184	21 de fevereiro de 2005	6.0-CURRENT after EOVERFLOW checks in vswprintf(3) fixed.
600015	r142501	25 de fevereiro de 2005	6.0-CURRENT after changing the struct if_data member, ifi_epoch, from wall clock time to uptime.
600016	r142582	26 de fevereiro de 2005	6.0-CURRENT after LC_CTYPE disk format changed.
600017	r142683	27 de fevereiro de 2005	6.0-CURRENT after NLS catalogs disk format changed.
600018	r142686	27 de fevereiro de 2005	6.0-CURRENT after LC_COLLATE disk format changed.
600019	r142752	28 de fevereiro de 2005	Installation of acpica includes into /usr/include.
600020	r143308	9 de março de 2005	Addition of MSG_NOSIGNAL flag to send(2) API.
600021	r143746	17 de março de 2005	Addition of fields to cdevsw
600022	r143901	21 de março de 2005	Removed gtar from base system.
600023	r144980	13 de abril de 2005	LOCAL_CREDS, LOCAL_CONNWAIT socket options added to unix(4).

Valor	Revisão	Data	Release
600024	r145565	19 de abril de 2005	hwpmc(4) and related tools added to 6.0-CURRENT.
600025	r145565	26 de abril de 2005	struct icmphdr added to 6.0-CURRENT.
600026	r145843	3 de maio de 2005	pf updated to 3.7.
600027	r145966	6 de maio de 2005	Kernel libalias and ng_nat introduced.
600028	r146191	13 de maio de 2005	POSIX ttyname_r(3) made available through unistd.h and libc.
600029	r146780	29 de maio de 2005	6.0-CURRENT after libpcap updated to v0.9.1 alpha 096.
600030	r146988	5 de junho de 2005	6.0-CURRENT after importing NetBSD's if_bridge(4).
600031	r147256	10 de junho de 2005	6.0-CURRENT after struct ifnet was broken out of the driver softcs.
600032	r147898	11 de julho de 2005	6.0-CURRENT after the import of libpcap v0.9.1.
600033	r148388	25 de julho de 2005	6.0-STABLE after bump of all shared library versions that had not been changed since RELENG_5.
600034	r149040	13 de agosto de 2005	6.0-STABLE after credential argument is added to dev_clone event handler. 6.0-RELEASE.
600100	r151958	1 de novembro de 2005	6.0-STABLE after 6.0-RELEASE
600101	r153601	21 de dezembro de 2005	6.0-STABLE after incorporating scripts from the local_startup directories into the base rcorder(8).

Valor	Revisão	Data	Release
600102	r153912	30 de dezembro de 2005	6.0-STABLE after updating the ELF types and constants.
600103	r154396	15 de janeiro de 2006	6.0-STABLE after MFC of pidfile(3) API.
600104	r154453	17 de janeiro de 2006	6.0-STABLE after MFC of ldconfig_local_dirs change.
600105	r156019	26 de fevereiro de 2006	6.0-STABLE after NLS catalog support of csh(1).
601000	r158330	6 de maio de 2006	6.1-RELEASE
601100	r158331	6 de maio de 2006	6.1-STABLE after 6.1-RELEASE.
601101	r159861	22 de junho de 2006	6.1-STABLE after the import of csup.
601102	r160253	11 de julho de 2006	6.1-STABLE after the iwi(4) update.
601103	r160429	17 de julho de 2006	6.1-STABLE after the resolver update to BIND9, and exposure of reentrant version of netdb functions.
601104	r161098	8 de agosto de 2006	6.1-STABLE after DSO (dynamic shared objects) support has been enabled in OpenSSL.
601105	r161900	2 de setembro de 2006	6.1-STABLE after 802.11 fixups changed the api for the IEEE80211_IOC_STA_INFO ioctl.
602000	r164312	15 de novembro de 2006	6.2-RELEASE
602100	r162329	15 de setembro de 2006	6.2-STABLE after 6.2-RELEASE.
602101	r165122	12 de dezembro de 2006	6.2-STABLE after the addition of Wi-Spy quirk.

Valor	Revisão	Data	Release
602102	r165596	28 de dezembro de 2006	6.2-STABLE after pci_find_extcap() addition.
602103	r166039	16 de janeiro de 2007	6.2-STABLE after MFC of dlsym change to look for a requested symbol both in specified dso and its implicit dependencies.
602104	r166314	28 de janeiro de 2007	6.2-STABLE after MFC of ng_deflate(4) and ng_pred1(4) netgraph nodes and new compression and encryption modes for ng_ppp(4) node.
602105	r166840	20 de fevereiro de 2007	6.2-STABLE after MFC of BSD licensed version of gzip(1) ported from NetBSD.
602106	r168133	31 de março de 2007	6.2-STABLE after MFC of PCI MSI and MSI-X support.
602107	r168438	6 de abril de 2007	6.2-STABLE after MFC of ncurses 5.6 and wide character support.
602108	r168611	11 de abril de 2007	6.2-STABLE after MFC of CAM 'SG' peripheral device, which implements a subset of Linux SCSI SG passthrough device API.
602109	r168805	17 de abril de 2007	6.2-STABLE after MFC of readline 5.2 patchset 002.

Valor	Revisão	Data	Release
602110	r169222	2 de maio de 2007	6.2-STABLE after MFC of pmap_invalidate_cache(), pmap_change_attr(), pmap_mapbios(), pmap_mapdev_attr(), and pmap_unmapbios() for amd64 and i386.
602111	r170556	11 de junho de 2007	6.2-STABLE after MFC of BOP_BDFLUSH and caused breakage of the filesystem modules KBI.
602112	r172284	21 de setembro de 2007	6.2-STABLE after libutil(3) MFC's.
602113	r172986	25 de outubro de 2007	6.2-STABLE after MFC of wide and single byte ctype separation. Newly compiled binary that references to ctype.h may require a new symbol, __mb_sb_limit, which is not available on older systems.
602114	r173170	30 de outubro de 2007	6.2-STABLE after ctype ABI forward compatibility restored.
602115	r173794	21 de novembro de 2007	6.2-STABLE after back out of wide and single byte ctype separation.
603000	r173897	25 de novembro de 2007	6.3-RELEASE
603100	r173891	25 de novembro de 2007	6.3-STABLE after 6.3-RELEASE.
(Não mudou)	r174434	7 de dezembro de 2007	6.3-STABLE after fixing multibyte type support in bit macro.
603102	r178459	24 de abril de 2008	6.3-STABLE after adding l_sysid to struct flock.

Valor	Revisão	Data	Release
603103	r179367	27 de maio de 2008	6.3-STABLE after MFC of the memrchr(3) function.
603104	r179810	15 de junho de 2008	6.3-STABLE after MFC of support for `:u` variable modifier in make(1).
604000	r183583	4 de outubro de 2008	6.4-RELEASE
604100	r183584	4 de outubro de 2008	6.4-STABLE after 6.4-RELEASE.

18.9. Versões do FreeBSD 5

Tabela 61. Valores do `__FreeBSD_version` para o FreeBSD 5

Valor	Revisão	Data	Release
500000	r58009	13 de março de 2000	5.0-CURRENT
500001	r59348	18 de abril de 2000	5.0-CURRENT after adding addition ELF header fields, and changing our ELF binary branding method.
500002	r59906	2 de maio de 2000	5.0-CURRENT after kld metadata changes.
500003	r60688	18 de maio de 2000	5.0-CURRENT after buf/bio changes.
500004	r60936	26 de maio de 2000	5.0-CURRENT after binutils upgrade.
500005	r61221	3 de junho de 2000	5.0-CURRENT after merging libxpg4 code into libc and after TASKQ interface introduction.
500006	r61500	10 de junho de 2000	5.0-CURRENT after the addition of AGP interfaces.
500007	r62235	29 de junho de 2000	5.0-CURRENT after Perl upgrade to 5.6.0

Valor	Revisão	Data	Release
500008	r62764	7 de julho de 2000	5.0-CURRENT after the update of KAME code to 2000/07 sources.
500009	r63154	14 de julho de 2000	5.0-CURRENT after ether_ifattach() and ether_ifdetach() changes.
500010	r63265	16 de julho de 2000	5.0-CURRENT after changing mtree defaults back to original variant, adding -L to follow symlinks.
500011	r63459	18 de julho de 2000	5.0-CURRENT after kqueue API changed.
500012	r65353	2 de setembro de 2000	5.0-CURRENT after setproctitle(3) moved from libutil to libc.
500013	r65671	10 de setembro de 2000	5.0-CURRENT after the first SMPng commit.
500014	r70650	4 de janeiro de 2001	5.0-CURRENT after <sys/select.h> moved to <sys/selinfo.h>.
500015	r70894	10 de janeiro de 2001	5.0-CURRENT after combining libgcc.a and libgcc_r.a, and associated GCC linkage changes.
500016	r71583	24 de janeiro de 2001	5.0-CURRENT after change allowing libc and libc_r to be linked together, deprecating -pthread option.
500017	r72650	18 de fevereiro de 2001	5.0-CURRENT after switch from struct ucred to struct xucred to stabilize kernel-exported API for mountd et al.

Valor	Revisão	Data	Release
500018	r72975	24 de fevereiro de 2001	5.0-CURRENT after addition of CPUTYPE make variable for controlling CPU-specific optimizations.
500019	r77937	9 de junho de 2001	5.0-CURRENT after moving machine/ioctl_fd.h to sys/fdcio.h
500020	r78304	15 de junho de 2001	5.0-CURRENT after locale names renaming.
500021	r78632	22 de junho de 2001	5.0-CURRENT after Bzip2 import. Also signifies removal of S/Key.
500022	r83435	12 de julho de 2001	5.0-CURRENT after SSE support.
500023	r83435	14 de setembro de 2001	5.0-CURRENT after KSE Milestone 2.
500024	r84324	1 de outubro de 2001	5.0-CURRENT after d_thread_t, and moving UUCP to ports.
500025	r84481	4 de outubro de 2001	5.0-CURRENT after ABI change for descriptor and creds passing on 64 bit platforms.
500026	r84710	9 de outubro de 2001	5.0-CURRENT after moving to XFree86 4 by default for package builds, and after the new libc strnstr() function was added.
500027	r84743	10 de outubro de 2001	5.0-CURRENT after the new libc strcasestr() function was added.
500028	r87879	14 de dezembro de 2001	5.0-CURRENT after the userland components of smbfs were imported.

Valor	Revisão	Data	Release
(Não mudou)			5.0-CURRENT after the new C99 specific-width integer types were added.
500029	r89938	29 de janeiro de 2002	5.0-CURRENT after a change was made in the return value of sendfile(2).
500030	r90711	15 de fevereiro de 2002	5.0-CURRENT after the introduction of the type `fflags_t`, which is the appropriate size for file flags.
500031	r91203	24 de fevereiro de 2002	5.0-CURRENT after the usb structure element rename.
500032	r92453	16 de março de 2002	5.0-CURRENT after the introduction of Perl 5.6.1.
500033	r93722	3 de abril de 2002	5.0-CURRENT after the `sendmail_enable` rc.conf(5) variable was made to take the value `NONE`.
500034	r95831	30 de abril de 2002	5.0-CURRENT after mtx_init() grew a third argument.
500035	r96498	13 de maio de 2002	5.0-CURRENT with Gcc 3.1.
500036	r96781	17 de maio de 2002	5.0-CURRENT without Perl in /usr/src
500037	r97516	29 de maio de 2002	5.0-CURRENT after the addition of dlfunc(3)
500038	r100591	24 de julho de 2002	5.0-CURRENT after the types of some struct sockbuf members were changed and the structure was reordered.

Valor	Revisão	Data	Release
500039	r102757	1 de setembro de 2002	5.0-CURRENT after GCC 3.2.1 import. Also after headers stopped using *BSD_FOO_T* and started using _FOO_T_DECLARED. This value can also be used as a conservative estimate of the start of bzip2(1) package support.
500040	r103675	20 de setembro de 2002	5.0-CURRENT after various changes to disk functions were made in the name of removing dependency on disklabel structure internals.
500041	r104250	1 de outubro de 2002	5.0-CURRENT after the addition of getopt_long(3) to libc.
500042	r105178	15 de outubro de 2002	5.0-CURRENT after Binutils 2.13 upgrade, which included new FreeBSD emulation, vec, and output format.
500043	r106289	1 de novembro de 2002	5.0-CURRENT after adding weak pthread_XXX stubs to libc, obsoleting libXThrStub.so. 5.0-RELEASE.
500100	r109405	17 de janeiro de 2003	5.0-CURRENT after branching for RELENG_5_0
500101	r111120	19 de fevereiro de 2003	<sys/dkstat.h> is empty. Do not include it.
500102	r111482	25 de fevereiro de 2003	5.0-CURRENT after the d_mmap_t interface change.

Valor	Revisão	Data	Release
500103	r111540	26 de fevereiro de 2003	5.0-CURRENT after taskqueue_swi changed to run without Giant, and taskqueue_swi_giant added to run with Giant.
500104	r111600	27 de fevereiro de 2003	cdevsw_add() and cdevsw_remove() no longer exists. Appearance of MAJOR_AUTO allocation facility.
500105	r111864	4 de março de 2003	5.0-CURRENT after new cdevsw initialization method.
500106	r112007	8 de março de 2003	devstat_add_entry() has been replaced by devstat_new_entry()
500107	r112288	15 de março de 2003	Devstat interface change; see sys/sys/param.h 1.149
500108	r112300	15 de março de 2003	Token-Ring interface changes.
500109	r112571	25 de março de 2003	Addition of vm_paddr_t.
500110	r112741	28 de março de 2003	5.0-CURRENT after realpath(3) has been made thread-safe
500111	r113273	9 de abril de 2003	5.0-CURRENT after usbhid(3) has been synced with NetBSD
500112	r113597	17 de abril de 2003	5.0-CURRENT after new NSS implementation and addition of POSIX.1 getpw*_r, getgr*_r functions
500113	r114492	2 de maio de 2003	5.0-CURRENT after removal of the old rc system.
501000	r115816	4 de junho de 2003	5.1-RELEASE.

Valor	Revisão	Data	Release
501100	r115710	2 de junho de 2003	5.1-CURRENT after branching for RELENG_5_1.
501101	r117025	29 de junho de 2003	5.1-CURRENT after correcting the semantics of sigtimedwait(2) and sigwaitinfo(2).
501102	r117191	3 de julho de 2003	5.1-CURRENT after adding the lockfunc and lockfuncarg fields to bus_dma_tag_create(9).
501103	r118241	31 de julho de 2003	5.1-CURRENT after GCC 3.3.1-pre 20030711 snapshot integration.
501104	r118511	5 de agosto de 2003	5.1-CURRENT 3ware API changes to twe.
501105	r119021	17 de agosto de 2003	5.1-CURRENT dynamically-linked /bin and /sbin support and movement of libraries to /lib.
501106	r119881	8 de setembro de 2003	5.1-CURRENT after adding kernel support for Coda 6.x.
501107	r120180	17 de setembro de 2003	5.1-CURRENT after 16550 UART constants moved from <dev/sio/sioreg.h> to <dev/ic/ns16550.h>. Also when libmap functionality was unconditionally supported by rtld.
501108	r120386	23 de setembro de 2003	5.1-CURRENT after PFIL_HOOKS API update
501109	r120503	27 de setembro de 2003	5.1-CURRENT after adding kiconv(3)

Valor	Revisão	Data	Release
501110	r120556	28 de setembro de 2003	5.1-CURRENT after changing default operations for open and close in cdevsw
501111	r121125	16 de outubro de 2003	5.1-CURRENT after changed layout of cdevsw
501112	r121129	16 de outubro de 2003	5.1-CURRENT after adding kobj multiple inheritance
501113	r121816	31 de outubro de 2003	5.1-CURRENT after the if_xname change in struct ifnet
501114	r122779	16 de novembro de 2003	5.1-CURRENT after changing /bin and /sbin to be dynamically linked
502000	r123198	7 de dezembro de 2003	5.2-RELEASE
502010	r126150	23 de fevereiro de 2004	5.2.1-RELEASE
502100	r123196	7 de dezembro de 2003	5.2-CURRENT after branching for RELENG_5_2
502101	r123677	19 de dezembro de 2003	5.2-CURRENT after *cxa_atexit*/cxa_finalize functions were added to libc.
502102	r125236	30 de janeiro de 2004	5.2-CURRENT after change of default thread library from libc_r to libpthread.
502103	r126083	21 de fevereiro de 2004	5.2-CURRENT after device driver API megapatch.
502104	r126208	25 de fevereiro de 2004	5.2-CURRENT after getopt_long_only() addition.
502105	r126644	5 de março de 2004	5.2-CURRENT after NULL is made into ((void *)0) for C, creating more warnings.

Valor	Revisão	Data	Release
502106	r126757	8 de março de 2004	5.2-CURRENT after pf is linked to the build and install.
502107	r126819	10 de março de 2004	5.2-CURRENT after time_t is changed to a 64-bit value on sparc64.
502108	r126891	12 de março de 2004	5.2-CURRENT after Intel C/C++ compiler support in some headers and execve(2) changes to be more strictly conforming to POSIX.
502109	r127312	22 de março de 2004	5.2-CURRENT after the introduction of the bus_alloc_resource_any API
502110	r127475	27 de março de 2004	5.2-CURRENT after the addition of UTF-8 locales
502111	r128144	11 de abril de 2004	5.2-CURRENT after the removal of the getvfsent(3) API
502112	r128182	13 de abril de 2004	5.2-CURRENT after the addition of the .warning directive for make.
502113	r130057	4 de junho de 2004	5.2-CURRENT after ttyioctl() was made mandatory for serial drivers.
502114	r130418	13 de junho de 2004	5.2-CURRENT after import of the ALTQ framework.
502115	r130481	14 de junho de 2004	5.2-CURRENT after changing sema_timedwait(9) to return 0 on success and a non-zero error code on failure.

Valor	Revisão	Data	Release
502116	r130585	16 de junho de 2004	5.2-CURRENT after changing kernel dev_t to be pointer to struct cdev *.
502117	r130640	17 de junho de 2004	5.2-CURRENT after changing kernel udev_t to dev_t.
502118	r130656	17 de junho de 2004	5.2-CURRENT after adding support for CLOCK_VIRTUAL and CLOCK_PROF to clock_gettime(2) and clock_getres(2).
502119	r130934	22 de junho de 2004	5.2-CURRENT after changing network interface cloning overhaul.
502120	r131429	2 de julho de 2004	5.2-CURRENT after the update of the package tools to revision 20040629.
502121	r131883	9 de julho de 2004	5.2-CURRENT after marking Bluetooth code as non-i386 specific.
502122	r131971	11 de julho de 2004	5.2-CURRENT after the introduction of the KDB debugger framework, the conversion of DDB into a backend and the introduction of the GDB backend.
502123	r132025	12 de julho de 2004	5.2-CURRENT after change to make VFS_ROOT take a struct thread argument as does vflush. Struct kinfo_proc now has a user data pointer. The switch of the default X implementation to `xorg` was also made at this time.

Valor	Revisão	Data	Release
502124	r132597	24 de julho de 2004	5.2-CURRENT after the change to separate the way ports rc.d and legacy scripts are started.
502125	r132726	28 de julho de 2004	5.2-CURRENT after the backout of the previous change.
502126	r132914	31 de julho de 2004	5.2-CURRENT after the removal of kmem_alloc_pageable() and the import of gcc 3.4.2.
502127	r132991	2 de agosto de 2004	5.2-CURRENT after changing the UMA kernel API to allow ctors/inits to fail.
502128	r133306	8 de agosto de 2004	5.2-CURRENT after the change of the vfs_mount signature as well as global replacement of PRISON_ROOT with SUSER_ALLOWJAIL for the suser(9) API.
503000	r134189	23 de agosto de 2004	5.3-BETA/RC before the pfil API change
503001	r135580	22 de setembro de 2004	5.3-RELEASE
503100	r136595	16 de outubro de 2004	5.3-STABLE after branching for RELENG_5_3
503101	r138459	3 de dezembro de 2004	5.3-STABLE after addition of glibc style strftime(3) padding options.
503102	r141788	13 de fevereiro de 2005	5.3-STABLE after OpenBSD's nc(1) import MFC.

Valor	Revisão	Data	Release
503103	r142639	27 de fevereiro de 2005	5.4-PRERELEASE after the MFC of the fixes in <src/include/stdbool.h> and <src/sys/i386/include/_types.h> for using the GCC-compatibility of the Intel C/C++ compiler.
503104	r142835	28 de fevereiro de 2005	5.4-PRERELEASE after the MFC of the change of ifi_epoch from wall clock time to uptime.
503105	r143029	2 de março de 2005	5.4-PRERELEASE after the MFC of the fix of EOVERFLOW check in vswprintf(3).
504000	r144575	3 de abril de 2005	5.4-RELEASE.
504100	r144581	3 de abril de 2005	5.4-STABLE after branching for RELENG_5_4
504101	r146105	11 de maio de 2005	5.4-STABLE after increasing the default thread stacksizes
504102	r504101	24 de junho de 2005	5.4-STABLE after the addition of sha256
504103	r150892	3 de outubro de 2005	5.4-STABLE after the MFC of if_bridge
504104	r152370	13 de novembro de 2005	5.4-STABLE after the MFC of bsdiff and portsnap
504105	r154464	17 de janeiro de 2006	5.4-STABLE after MFC of ldconfig_local_dirs change.
505000	r158481	12 de maio de 2006	5.5-RELEASE.
505100	r158482	12 de maio de 2006	5.5-STABLE after branching for RELENG_5_5

18.10. Versões do FreeBSD 4

Tabela 62. Valores do `__FreeBSD_version` para o FreeBSD 4

Valor	Revisão	Data	Release
400000	r43041	22 de janeiro de 1999	4.0-CURRENT after 3.4 branch
400001	r44177	20 de fevereiro de 1999	4.0-CURRENT after change in dynamic linker handling
400002	r44699	13 de março de 1999	4.0-CURRENT after C++ constructor/destructor order change
400003	r45059	27 de março de 1999	4.0-CURRENT after functioning dladdr(3)
400004	r45321	5 de abril de 1999	4.0-CURRENT after __deregister_frame_info dynamic linker bug fix (also 4.0-CURRENT after EGCS 1.1.2 integration)
400005	r46113	27 de abril de 1999	4.0-CURRENT after suser(9) API change (also 4.0-CURRENT after newbus)
400006	r47640	31 de maio de 1999	4.0-CURRENT after cdevsw registration change
400007	r47992	17 de junho de 1999	4.0-CURRENT after the addition of so_cred for socket level credentials
400008	r48048	20 de junho de 1999	4.0-CURRENT after the addition of a poll syscall wrapper to libc_r
400009	r48936	20 de julho de 1999	4.0-CURRENT after the change of the kernel's `dev_t` type to `struct specinfo` pointer
400010	r51649	25 de setembro de 1999	4.0-CURRENT after fixing a hole in jail(2)

Valor	Revisão	Data	Release
400011	r51791	29 de setembro de 1999	4.0-CURRENT after the `sigset_t` datatype change
400012	r53164	15 de novembro de 1999	4.0-CURRENT after the cutover to the GCC 2.95.2 compiler
400013	r54123	4 de dezembro de 1999	4.0-CURRENT after adding pluggable linux-mode ioctl handlers
400014	r56216	18 de janeiro de 2000	4.0-CURRENT after importing OpenSSL
400015	r56700	27 de janeiro de 2000	4.0-CURRENT after the C++ ABI change in GCC 2.95.2 from -fvtable -thunks to -fno-vtable -thunks by default
400016	r57529	27 de fevereiro de 2000	4.0-CURRENT after importing OpenSSH
400017	r58005	13 de março de 2000	4.0-RELEASE
400018	r58170	17 de março de 2000	4.0-STABLE after 4.0-RELEASE
400019	r60047	5 de maio de 2000	4.0-STABLE after the introduction of delayed checksums.
400020	r61262	4 de junho de 2000	4.0-STABLE after merging libxpg4 code into libc.
400021	r62820	8 de julho de 2000	4.0-STABLE after upgrading Binutils to 2.10.0, ELF branding changes, and tcsh in the base system.
410000	r63095	14 de julho de 2000	4.1-RELEASE
410001	r64012	29 de julho de 2000	4.1-STABLE after 4.1-RELEASE
410002	r65962	16 de setembro de 2000	4.1-STABLE after setproctitle(3) moved from libutil to libc.
411000	r66336	25 de setembro de 2000	4.1.1-RELEASE

Valor	Revisão	Data	Release
411001			4.1.1-STABLE after 4.1.1-RELEASE
420000	r68066	31 de outubro de 2000	4.2-RELEASE
420001	r70895	10 de janeiro de 2001	4.2-STABLE after combining libgcc.a and libgcc_r.a, and associated GCC linkage changes.
430000	r73800	6 de março de 2001	4.3-RELEASE
430001	r76779	18 de maio de 2001	4.3-STABLE after wint_t introduction.
430002	r80157	22 de julho de 2001	4.3-STABLE after PCI powerstate API merge.
440000	r80923	1 de agosto de 2001	4.4-RELEASE
440001	r85341	23 de outubro de 2001	4.4-STABLE after d_thread_t introduction.
440002	r86038	4 de novembro de 2001	4.4-STABLE after mount structure changes (affects filesystem klds).
440003	r88130	18 de dezembro de 2001	4.4-STABLE after the userland components of smbfs were imported.
450000	r88271	20 de dezembro de 2001	4.5-RELEASE
450001	r91203	24 de fevereiro de 2002	4.5-STABLE after the usb structure element rename.
450002	r92151	12 de março de 2002	4.5-STABLE after locale changes.
450003			(Never created)
450004	r94840	16 de abril de 2002	4.5-STABLE after the `sendmail_enable` rc.conf(5) variable was made to take the value `NONE`.

Valor	Revisão	Data	Release
450005	r95555	27 de abril de 2002	4.5-STABLE after moving to XFree86 4 by default for package builds.
450006	r95846	1 de maio de 2002	4.5-STABLE after accept filtering was fixed so that is no longer susceptible to an easy DoS.
460000	r97923	21 de junho de 2002	4.6-RELEASE
460001	r98730	21 de junho de 2002	4.6-STABLE sendfile(2) fixed to comply with documentation, not to count any headers sent against the amount of data to be sent from the file.
460002	r100366	19 de julho de 2002	4.6.2-RELEASE
460100	r98857	26 de junho de 2002	4.6-STABLE
460101	r98880	26 de junho de 2002	4.6-STABLE after MFC of `sed -i`.
460102	r102759	1 de setembro de 2002	4.6-STABLE after MFC of many new pkg_install features from the HEAD.
470000	r104655	8 de outubro de 2002	4.7-RELEASE
470100	r104717	9 de outubro de 2002	4.7-STABLE
470101	r106732	10 de novembro de 2002	Start generated *std{in,out,err}p references rather than* sF. This changes std{in,out,err} from a compile time expression to a runtime one.
470102	r109753	23 de janeiro de 2003	4.7-STABLE after MFC of mbuf changes to replace m_aux mbufs by m_tag's
470103	r110887	14 de fevereiro de 2003	4.7-STABLE gets OpenSSL 0.9.7

Valor	Revisão	Data	Release
480000	r112852	30 de março de 2003	4.8-RELEASE
480100	r113107	5 de abril de 2003	4.8-STABLE
480101	r115232	22 de maio de 2003	4.8-STABLE after realpath(3) has been made thread-safe
480102	r118737	10 de agosto de 2003	4.8-STABLE 3ware API changes to twe.
490000	r121592	27 de outubro de 2003	4.9-RELEASE
490100	r121593	27 de outubro de 2003	4.9-STABLE
490101	r124264	8 de janeiro de 2004	4.9-STABLE after e_sid was added to struct kinfo_eproc.
490102	r125417	4 de fevereiro de 2004	4.9-STABLE after MFC of libmap functionality for rtld.
491000	r129700	25 de maio de 2004	4.10-RELEASE
491100	r129918	1 de junho de 2004	4.10-STABLE
491101	r133506	11 de agosto de 2004	4.10-STABLE after MFC of revision 20040629 of the package tools
491102	r137786	16 de novembro de 2004	4.10-STABLE after VM fix dealing with unwiring of fictitious pages
492000	r138960	17 de dezembro de 2004	4.11-RELEASE
492100	r138959	17 de dezembro de 2004	4.11-STABLE
492101	r157843	18 de abril de 2006	4.11-STABLE after adding libdata/ldconfig directories to mtree files.

18.11. Versões do FreeBSD 3

Tabela 63. Valores do `__FreeBSD_version` para o FreeBSD 3

Valor	Revisão	Data	Release
300000	r22917	19 de fevereiro de 1996	3.0-CURRENT before mount(2) change

Valor	Revisão	Data	Release
300001	r36283	24 de setembro de 1997	3.0-CURRENT after mount(2) change
300002	r36592	2 de junho de 1998	3.0-CURRENT after semctl(2) change
300003	r36735	7 de junho de 1998	3.0-CURRENT after ioctl arg changes
300004	r38768	3 de setembro de 1998	3.0-CURRENT after ELF conversion
300005	r40438	16 de outubro de 1998	3.0-RELEASE
300006	r40445	16 de outubro de 1998	3.0-CURRENT after 3.0-RELEASE
300007	r43042	22 de janeiro de 1999	3.0-STABLE after 3/4 branch
310000	r43807	9 de fevereiro de 1999	3.1-RELEASE
310001	r45060	27 de março de 1999	3.1-STABLE after 3.1-RELEASE
310002	r45689	14 de abril de 1999	3.1-STABLE after C++ constructor/destructor order change
320000			3.2-RELEASE
320001	r46742	8 de maio de 1999	3.2-RELEASE
320002	r50563	29 de agosto de 1999	3.2-STABLE after binary-incompatible IPFW and socket changes
330000	r50813	2 de setembro de 1999	3.3-RELEASE
330001	r51328	16 de setembro de 1999	3.3-RELEASE
330002	r53671	24 de novembro de 1999	3.3-STABLE after adding mkstemp(3) to libc
340000	r54166	5 de dezembro de 1999	3.4-RELEASE
340001	r54730	17 de dezembro de 1999	3.4-RELEASE
350000	r61876	20 de junho de 2000	3.5-RELEASE
350001	r63043	12 de julho de 2000	3.5-STABLE

18.12. Versões do FreeBSD 2.2

Tabela 64. Valores do `__FreeBSD_version` para o FreeBSD 2.2

Valor	Revisão	Data	Release
220000	r22918	19 de fevereiro de 1997	2.2-RELEASE
(Não mudou)			2.2.1-RELEASE
(Não mudou)			2.2-STABLE after 2.2.1-RELEASE
221001	r24941	15 de abril de 1997	2.2-STABLE after texinfo-3.9
221002	r25325	30 de abril de 1997	2.2-STABLE after top
222000	r25851	16 de maio de 1997	2.2.2-RELEASE
222001	r25921	19 de maio de 1997	2.2-STABLE after 2.2.2-RELEASE
225000	r30053	2 de outubro de 1997	2.2.5-RELEASE
225001	r31300	20 de novembro de 1997	2.2-STABLE after 2.2.5-RELEASE
225002	r32019	27 de dezembro de 1997	2.2-STABLE after ldconfig -R merge
226000	r34445	24 de março de 1998	2.2.6-RELEASE
227000	r37803	21 de julho de 1998	2.2.7-RELEASE
227001	r37809	21 de julho de 1998	2.2-STABLE after 2.2.7-RELEASE
227002	r39489	19 de setembro de 1998	2.2-STABLE after semctl(2) change
228000	r41403	29 de novembro de 1998	2.2.8-RELEASE
228001	r41418	29 de novembro de 1998	2.2-STABLE after 2.2.8-RELEASE

Note que o 2.2-STABLE às vezes é identificado como "2.2.5-STABLE" após o 2.2.5-RELEASE. O padrão costumava ser ano seguido do mês, mas decidimos mudá-lo para um sistema maior/menor mais simples a partir de 2.2. Isso aconteceu porque o desenvolvimento paralelo em várias branches inviabilizou a classificação dos lançamentos apenas por suas datas reais de lançamento. Não se preocupe com velhos -CURRENTs; eles estão listados aqui apenas para referência.

18.13. FreeBSD 2 Antes das Versões 2.2-RELEASE

Tabela 65. Valores do `__FreeBSD_version` para o FreeBSD 2 de antes da 2.2-RELEASE

Valor	Revisão	Data	Release
119411			2.0-RELEASE
199501	r7153	19 de março de 1995	2.1-CURRENT
199503	r7310	24 de março de 1995	2.1-CURRENT
199504	r7704	9 de abril de 1995	2.0.5-RELEASE
199508	r10297	26 de agosto de 1995	2.2-CURRENT before 2.1
199511	r12189	10 de novembro de 1995	2.1.0-RELEASE
199512	r12196	10 de novembro de 1995	2.2-CURRENT before 2.1.5
199607	r17067	10 de julho de 1996	2.1.5-RELEASE
199608	r17127	12 de julho de 1996	2.2-CURRENT before 2.1.6
199612	r19358	15 de novembro de 1996	2.1.6-RELEASE
199612			2.1.7-RELEASE