FreeBSD Porter’s Handbook

概要

このハンドブックは FreeBSD の port 作成者 (porter) 向けに、 具体的な port の作成方法や注意点などをまとめたものです。

日本語版の作成は FreeBSD 日本語ドキュメンテーション プロジェクト (FreeBSD doc-jp) が行なっています。 日本語訳および、日本語版のみに関することは FreeBSD 日本語ドキュメンテーションプロジェクト <doc-jp@jp.FreeBSD.org> において日本語で議論されています。

文書の日本語訳に関するお問い合わせや、
文書の原文に関する問い合わせをしたいが英語が得意でないという方は、 FreeBSD 日本語ドキュメンテーションプロジェクト <doc-jp@jp.FreeBSD.org> まで日本語でコメントをお寄せください。

目次

1. はじめに . . . 5
2. 自分で port を作成するには . . . 6
3. 分間 porting . . . 7
   3.1. Makefile の作成 . . . 7
   3.2. package 記述ファイルの作成 . . . 7
   3.3. チェックサムファイルの作成 . . . 9
   3.4. port のテスト . . . 9
   3.5. `portlint` によるチェック . . . 10
   3.6. port の提出 . . . 10
4. 本格的な port . . . 11
   4.1. port 構築の詳細 . . . 11
   4.2. オリジナルのソースの入手 . . . 12
   4.3. port の修正 . . . 13
   4.4. パッチの適用 . . . 13
   4.5. コンフィグレーション . . . 14
   4.6. ユーザからの入力の扱い . . . 14
5. Makefile の作成 . . . 15
   5.1. オリジナルのソース . . . 15
   5.2. 名前の付け方 . . . 15
   5.3. カテゴリ分類 . . . 20
   5.4. 配布ファイル . . . 26
   5.5. `MAINTAINER` . . . 35
   5.6. `COMMENT` . . . 35
   5.7. 依存関係 . . . 36
   5.8. 作業ディレクトリの指定 . . . 41
   5.9. `CONFLICTS` . . . 42
   5.10. ビルドのメカニズム . . . 42
6. 特別な配慮 . . . 43
   6.1. 共有ライブラリ . . . 43
   6.2. 配布制限がある ports . . . 43
   6.3. `perl` の利用 . . . 44
   6.4. X11 の利用 . . . 45
   6.5. `automake`, `autoconf` および `libtool` の利用 . . . 45
   6.6. GNOME の利用 . . . 47
   6.7. KDE の利用 . . . 47
   6.8. Bison の利用 . . . 47
   6.9. Java の利用 . . . 48
   6.10. Python の利用 . . . 50
   6.11. Emacs の利用 . . . 50

6.12. Ruby の利用 . . . 50
6.13. SDL の利用 . . . 50
7. MASTERDIR . . . 52
8. 共有ライブラリのバージョン . . . 53
9. マニュアルページ . . . 54
10. Motif を必要とする port . . . 56
10.1. USE_MOTIF . . . 56
10.2. MOTIFLIB . . . 56
11. X11 のフォント . . . 57
12. Info ファイル . . . 58
13. pkg-* ファイル . . . 59
13.1. pkg-message . . . 59
13.2. pkg-install . . . 59
13.3. pkg-deinstall . . . 59
13.4. pkg-req . . . 59
13.5. make の変数にあわせた pkg-plist の変更 . . . 60
13.6. pkg-* ファイルの名前変更 . . . 60
14. port のテスト . . . 62
14.1. portlint . . . 62
14.2. PREFIX . . . 62
14.3. FreshPorts 正当性テスト . . . 63
15. アップグレード . . . 64
16. やっていいことといけないこと . . . 65
16.1. はじめに . . . 65
16.2. バイナリの strip . . . 65
16.3. INSTALL_* マクロ . . . 65
16.4. WRKDIR . . . 66
16.5. WRKDIRPREFIX . . . 66
16.6. OS の種類やバージョンの識別 . . . 66
16.7. __FreeBSD_version の値 . . . 67
16.8. bsd.port.mk の後に書くこと . . . 76
16.9. 付加的な文書のインストール . . . 78
16.10. ディレクトリ構成 . . . 78
16.11. 空のディレクトリの削除 . . . 79
16.12. UID . . . 79
16.13. 合理的な port . . . 81
16.14. CC および CXX の尊重 . . . 81
16.15. CFLAGS の尊重 . . . 82
16.16. コンフィグレーション (設定) ファイル . . . 82
16.17. フィードバック . . . 83
16.18. README.html . . . 83

16.19. Port に BROKEN, FORBIDDEN などの印をつける 83
16.20. 必要な回避策 84
16.21. その他諸々 84
16.22. 困ったら... 84
17. Makefile のサンプル 85
18. パッキングリストの自動生成 87
19. この文書と ports システムの変更 88

Chapter 1. はじめに

FreeBSD Ports Collection は、ほとんどの人が FreeBSD でアプリケーション ("ports") をインストールする手段です。 FreeBSD に関する他のすべてと同じく、これも基本的にボランティア活動によるものです。
この文書を読む際には必ずこのことを念頭においてください。

FreeBSD では、誰もが新たな port を提出したり、 メンテナンスされていない既存の port をメンテナンスできます。 そのためには特にソースコードを管理する (commit) 権限は必要ありません。

Chapter 2. 自分で port を作成するには

自分で port を作ることや、既存の port の 更新作業に興味があるのですか。それはすばらしい!

ここでは FreeBSD 用の port を作る際の ガイドラインをいくつか示します。 既存の port を更新したいと考えている場合であっても、 まずこの章を読んでから、次に アップグレード を読むようにしてください。

この文書では充分に詳細がわからない場合には、 /usr/ports/Mk/bsd.port.mk を参照してください。 このファイルは、port の Makefile が例外なくインクルードしているものです。 これには細かくコメントが書かれていますので、Makefile を読むのに あまり慣れていない人でも、たくさんの情報を得ることができるでしょう。 それでも解決できないような質問は、FreeBSD ports メーリングリスト にポストしてみるのも 良いでしょう。

この文書では、上書き可能な 変数 (VAR) のうち 一部のものについてだけ述べています。 (すべてでは無いかもしれませんが、) ほとんどの変数は bsd.port.mk の先頭部分に記述されています。 それ以外のものも記述すべきかもしれません。 なお、このファイルは非標準のタブ設定を使用しています。 Emacs や Vim は、 この設定をファイルの読み込み時に認識するはずです。 vi(1) と ex(1) では、 一旦ファイルを読み込んでから :set tabstop=4 と タイプすることで、正しい値に設定することができます。

Chapter 3. 分間 porting

このセクションでは、簡単な port の作り方について説明します。
多くの場合、これだけでは不十分なので、 この文書の続きを読まなければならないでしょう。

まず、元の tar ファイルを DISTDIR に置きます。この変数の デフォルト値は /usr/ports/distfiles です。

以下の例では、そのソフトウェアが そのままコンパイル可能なものと仮定しています。
つまり、FreeBSD マシンで動かすために、 変更がまったく必要ないという意味です。
もし何か変更が必要な場合には、次のセクションも 参照する必要があるでしょう。

3.1. Makefile の作成

最小限の Makefile は 次のようなものになります。

```
# New ports collection makefile for:   oneko
# Date created:      5 December 1994
# Whom:              asami
#
# $FreeBSD$
#

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

MAINTAINER=     asami@FreeBSD.org
COMMENT=        A cat chasing a mouse all over the screen

MAN1=           oneko.1
MANCOMPRESSED=  yes
USE_IMAKE=      yes

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

おわかりでしょうか。$FreeBSD$ を 含む行の内容については、気にする必要はありません。
この行は、このファイルが FreeBSD の ports ツリーに 取り込まれる際に、CVS によって自動的に書き込まれます。 もっと詳しい例が見たい場合には、 Makefile のサンプルの セクションをご覧ください。

3.2. package 記述ファイルの作成

package にするしないに関わらず、どのような port でも 2 つの記述ファイルが必要です。それは pkg-descr と pkg-plist です。ファイル名が pkg- で始まっていることで 他のファイルと区別できるようになっています。

3.2.1. pkg-descr

このファイルには、その port についての少し長い説明を書きます。 その port が何をするのかについての、 数段落程度の簡潔な解説があれば充分です。

これはマニュアルでもなければ、使用方法やコンパイル方法についての細かい説明書でもありません。 *README* ファイルやマニュアルを引用するつもりなら注意が必要です。 これらは多くの場合、その port の簡潔な説明になっていなかったり、 扱いにくい形式になっていたりします。(マニュアルの場合、行を揃えるために空白が調整されていたりします。)
このソフトウェアに公式のウェブサイトがあるのなら、ここに書いてください。その際自動化ツールが正しく動作するように、ウェブサイトのうちの一つには、 先頭に `WWW:` をつけておいてください。

このファイルの最後に、あなたの名前を書くことが推奨されています。 たとえば、こんな具合です。

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

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

- Satoshi
asami@cs.berkeley.edu
```

3.2.2. pkg-plist

このファイルには、その port によってインストールされる すべてのファイルを列挙します。このファイルは package を作る際のリストとして使われるため、 "パッキングリスト (packing list)" とも呼ばれます。 ここに書くパス名は、インストール時のプレフィックス (通常 /usr/local または /usr/X11R6) からの相対パスです。 `MANn` 変数を 使用している場合 (使用することが推奨されています)、このリストに マニュアルは入れないようにしてください。

簡単な例を載せておきましょう。

```
bin/oneko
lib/X11/app-defaults/Oneko
lib/X11/oneko/cat1.xpm
lib/X11/oneko/cat2.xpm
lib/X11/oneko/mouse.xpm
@dirrm lib/X11/oneko
```

パッキングリストの詳細については、 pkg_create(1) のマニュアルを参照してください。

このリストには、すべてのファイルを列挙しなければなりませんが、ディレクトリそのものは列挙する必要がありません。 また、この port

がインストール時に独自のディレクトリを 作成する場合には、この port が削除されるときに そのディレクトリも削除されるよう、`@dirrm` の行を追加しておくのを忘れないでください。

このファイルでは、すべてのファイル名を アルファベット順にソートしておくことを推奨します。 そうすることで、port を更新する際の 変更点の確認が楽になります。

パッキングリストを手作業で作成するのは、 時にとても退屈な作業になります。 その port が非常に多数のファイルをインストールするとしたら、 パッキングリストの自動生成を行なえば、時間の節約になるかもしれません。

3.3. チェックサムファイルの作成

`make makesum` と入力するだけで、 (訳注: bsd.port.mk に書かれている) port 生成ルールに従い、自動的に distinfo ファイルが生成されます。

3.4. port のテスト

package 化も含め、その port が思った通りに 動くことを確認してください。確認の必要な重要ポイントは以下の通りです。

- その port がインストールしないものが pkg-plist に含まれていないこと。
- その port がインストールする、すべてのものが pkg-plist に含まれていること。
- `reinstall` ターゲットを使うことで、その port が 何度でもインストール可能なこと。
- その port が deintall される際には 後片付けをすること。

Procedure: 推奨されるテストの手順

1. `make install`
2. `make package`
3. `make deinstall`
4. `pkg_add package 名`
5. `make deinstall`
6. `make reinstall`
7. `make package`

`package` および `deinstall` の段階で、 どんな警告 (warning) も出力されないことを確認してください。ステップ 3 の後、(訳注: その port が作成した) すべての新しいディレクトリが正しく消去されていることを確認してください。 また、ステップ 4 の後にそのソフトウェアを使用してみて、 package からインストールされた場合にも正しく動作することを 確認してください。

3.5. portlint によるチェック

portlint を使い、その port が FreeBSD の ガイドラインに沿っているかどうかを確認してください。portlint プログラムは ports collection に 含まれています。 特に、Makefile が正しい形式になっているか、 package の 名前が正しいかどうかをチェックするのに良いでしょう。

3.6. port の提出

まず、やって良いこと悪いことの セクションを読んでください。

さて、満足のいく port が完成したら、残るは それを FreeBSD のメインの ports ツリーに置いて、他の人にも使ってもらうだけです。 work ディレクトリや pkgname.tgz といった package は必要ありませんから、まずこれらを消去してください。 あとは shar find port_dir の出力をバグレポートに入れ、send-pr(1) プログラムを使用して 送ってください (send-pr(1) についての詳細はバグ報告と一般的な論評を参照してください)。 もし、圧縮していない状態で 20KB 以上あるような port であれば、 それを ひとつの tar ファイルにまとめて圧縮し、バグレポートに入れる前に uuencode(1) を使用してください (20KB 以下のものを tar ファイルにして送っても良いのですが、 あまり歓迎されません)。 バクレポートの category は必ず ports, class は change-request としてください (レポートを confidential (機密) 指定には しないでください!)。また、port 化したプログラムの短い説明文を バグレポートの "Description" フィールドに追加して、"Fix" フィールドには shar したファイル、 もしくは uuencode した tar ファイルを追加するようにしてください。

障害報告の概要 (synopsis) 欄がよく書けていると、わたしたちが作業しやすくなります。 新しい port を提出するなら "New port: <カテゴリ>/<port 名> <短い port の概要>"、 port の更新なら、 "Update port: <カテゴリ>/<port 名> <短い更新の概要>" のような形が歓迎されます。この考え方に沿っていれば、
誰かがあなたの障害報告を時間をおかずに見てくれる可能性が高くなります。

もう一度、オリジナルのソースファイルや *work* ディレクトリ、 make package で作成した *package* が含まれていないことを確認してください。

port を提出したら、辛抱強くお待ちください。時には、ある port が FreeBSD に取り込まれるまで、数日しかかかりそうもないの に、数ヶ月かかることもあります。FreeBSD へのコミット待ちの ports の一覧が見られます。

わたしたちがひとたびその port をチェックしたら、必要なら あなたに確認して、それをツリーへ置きます。 あなたの名前はその他の FreeBSD への貢献者の一覧やその他のファイルにも載るでしょう。 う〜ん、素晴らしい。:-)

Chapter 4. 本格的な port

残念ながら移植がそう簡単ではなく、それを動かすために 多少の変更が必要になる場合もあるでしょう。このセクションでは、模範的な ports の作法に従い、 どのように変更を行なって動くようにするのかを順を追って説明します。

4.1. port 構築の詳細

まず、あなたが port のディレクトリで make と 入力してから起こる一連の出来事について、順を追って説明します。 ここを読むときには、別のウィンドウに bsd.port.mk を表示しておくと理解の助けになるかもしれません。

しかし、bsd.port.mk が何をしているのか 完全に理解できなくても 心配する必要はありません。それほど多くの人が理解している というわけでは ありませんから...。 *f(;)_*

1. まず、fetch という ターゲットが実行されます。 この fetch ターゲットは、配布ファイルがローカルの DISTDIR に 存在することを保証する役目を持っています。もし必要なファイルが DISTDIR に 存在しなければ、fetch ターゲットは Makefile で指定された MASTER_SITES 中の URL や、 FreeBSD のメイン FTP サイト ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/distfiles/ (ここにはバックアップとして、われわれ ports 管理者が確認した 配布ファイルを置いてあります) を探しにいきます。 make を実行するマシンがインターネットに 接続されていて、目的のファイルを FETCH で取ってこれた場合には、それを DISTDIR に 保存します。
2. 次に extract ターゲットが実行されます。 このターゲットは DISTDIR から 配布ファイル (普通は gzip された tar ファイル) を読み込み、 その内容を作業ディレクトリ WRKDIR (デフォルトでは work) に展開します。
3. 次に patch ターゲットが実行されます。 まず、PATCHFILES にパッチファイルが指定されていれば、そのパッチを適用します。 次に、PATCHDIR ディレクトリ (デフォルトでは files サブディレクトリ) に patch-* という 名前のパッチファイルが存在すれば、これらをアルファベット順に適用します。
4. 次に configure ターゲットが 実行されます。 これには、いろいろな場合があります。
   a. scripts/configure が 存在する場合には、そのスクリプトが実行されます。
   b. HAS_CONFIGURE または GNU_CONFIGURE がセットされていれば、 WRKSRC/configure が実行されます。
   c. USE_IMAKE がセットされていれば、 XMKMF (デフォルトでは xmkmf -a) が 実行されます。
5. 最後に build ターゲットが実行されます。 これは作業ディレクトリ (WRKSRC) に降りていき、ビルド (コンパイル) を実行するのが役目です。 USE_GMAKE がセットされていれば GNU make が使用され、 セットされていなければ FreeBSD の make が 使用されます。

上記はデフォルトの動作です。これに加えて pre-何とかや post-何とかというターゲットを定義したり、そのような名前のスクリプトを scripts サブディレクトリに置くことも可能で、 それぞれデフォルトの動作の前や後に実行されます。

たとえば、post-extract ターゲットが Makefile に定義されていて、 scripts サブディレクトリに pre-

build というファイルが置かれている場合、post-extract ターゲットは通常の展開動作の後に呼び出され、pre-build スクリプトはデフォルトのコンパイル動作の前に実行されます。 実行する動作が簡単であれば、スクリプトよりも Makefile のターゲットを使用することが 推奨されています。 なぜなら、その port ではどのような非標準の動作が必要とされるのか、
一箇所にまとめて書いてあった方が他の人に理解しやすいからです。

デフォルトの動作は bsd.port.mk の do-何とかという ターゲットで実行されます。 たとえば port を展開するコマンドは do-extract ターゲットに書かれています。
もしデフォルトのターゲットに不満があれば、 Makefile 中で do- 何とかというターゲットを再定義することにより、 好きなように変更することができます。

"メイン"のターゲット (たとえば extract, configure、その他) は、
すべての前段階が実行されていることを確認してから、
実際のターゲットやスクリプトを呼び出す以外のことは
行ないませんし、これらが変更されることも想定されていません。
もし展開の方法を変更したいときには do-extract の変更によって実現し、 extract の動作は絶対に変更しないでください。

これで、ユーザが make と 入力したときに何が起こるのかが理解できたと思います。 では、完璧な port を作成するための推奨手順を 順に見ていきましょう。

4.2. オリジナルのソースの入手

(通常の場合、) 圧縮された tar ファイルの形 (foo.tar.gz あるいは foo.tar.Z) で オリジナルのソースを入手して、 それを DISTDIR にコピーします。 できる限り、主流のソースを使用するようにしてください。

もとの tar ファイルがどこにあるかを示すために、変数 MASTER_SITES を設定する必要があります。 主なサイトのほとんどについては省略形が bsd.sites.mk で定義されています。 これらのサイト (と付随する定義) を、ソースコード内で同じ情報が繰り返されるのを避けるために、可能な限り使うようにしてください。
これらのサイトは時とともに変わってゆきますので、そうしないと、
関係者一同にとってメンテナンスの悪夢になってしまいます。

ネットワークへの接続の良好な FTP/HTTP サイトを見つけることができなかったり、頭にくるような非標準的な形式しか
置いていないサイトしか見つけられないときには、 自分の管理下にあり信頼できる FTP サーバや HTTP サーバ (たとえば、あなた自身のホームページ) に置くこともできます。

そのような便利かつ信頼のおける置き場所が見つからない場合、 我々が ftp.FreeBSD.org に "置き場所"を提供することもできますが、 これはなるべく避けたい解決法です。 配布ファイルは、誰かの freefall アカウントの ~/public_distfiles/ に置かれることでしょう。 その port をコミットする人に、置いてもらえるように頼んでください。
その人は配布ファイルを置いて、MASTER_SITES を MASTER_SITE_LOCAL にセットし、 MASTER_SITE_SUBDIR には 自分の freefall ユーザ名を 入れておいてくれるでしょう。

その port の配布ファイルが、 作者によるバージョン更新のようなことがなく変更されるなら、その配布ファイルを あなたのホームページに置いて、MASTER_SITES の最初に指定することも考えてみてください。できれば、その port

の作者にそういうことをしないようにお願いしてみてください。
そのためには、何かしらのソースコード管理を行うと役に立つでしょう。
あなたが独自のバージョンを置けば、ユーザが checksum mismatch エラーに悩まされることもなくなりますし、FreeBSD の FTP サイトの保守担当者の負担も減らすこともできます。 また、その port にマスターサイトが一つしか存在しない場合には、 あなたのサイトにバックアップを置き、 それを MASTER_SITES の 2 番目に 指定すると良いでしょう。

その port がインターネット上で入手できる追加パッチを 必要とするのなら、それも取ってきて DISTDIR に置いてください。 それらがメインのソースの tar ファイルとは別のサイトにあったとしても、心配する必要はありません。 そのような状況にもちゃんと対応できるようになっています (後述の PATCHFILES の記述を ご覧ください)。

4.3. port の修正

作業用のディレクトリに tar ファイルを展開し、 最新バージョンの FreeBSD 上で正しくコンパイルするために必要な、 あらゆる変更を行ないます。 この処理は最終的に自動化するわけですから、 何を行なったかを注意深く記録しておきましょう。 この port が完成した暁には、ファイルの削除、追加、 修正を含むすべての処理が自動化されたスクリプトやパッチファイルで行なえるようになっていなければなりません。

その port のコンパイルやインストールのために必要な手作業が あまりに多いようならば、Larry Wall の芸術的な Configure スクリプトを 参考にしたほうが良いかもしれません。 新しい ports collection は、エンドユーザにとって個々の port が 可能な限り"プラグ & プレイ"かつ 最小のディスク消費で make できることを目指しています。

明示的に記述されている場合を除き、あなたが作成して FreeBSD の ports collection に寄付したパッチファイル、 スクリプトおよびその他のファイルは、標準的な BSD の著作権条件によりカバーされているものと見なされます。

4.4. パッチの適用

port の準備段階で追加されたり変更されたりしたファイルは、 再帰的 diff(1) により後で patch(1) に与えられる形にすることができます。 パッチは適当にまとめて patch- という名前のファイルに入れてください。 はパッチが適用される順番を示します - これらはアルファベット順、 つまり aa が最初、 ab が その次といった順番で処理されます。 お望みなら、patch-Imakefile とか patch-src-config.h のように、パッチ対象のファイルのパス名を示す名前を使うこともできます。 これらのファイルは PATCHDIR に置いてください。 そうすれば自動的に適用されるようになっています。 すべてのパッチは WRKSRC からの相対パスにするべきです (通常、WRKSRC は port の tar ファイルが展開されるディレクトリで、make が実行されるところと同じです)。 修正やアップグレードを容易にするため、複数のパッチで同じファイルを修正するのは避けてください (たとえば、patch-aa と patch-ab が共に WRKSRC/foobar.c を修正するなど)。

RCS にとって特別な意味を持つ文字列をパッチ内に入れないようにしてください。ファイルを私たちのソースツリーに入れる時、 これらの文字列は CVS によって書き換えられてしまい、後でまたパッチを使おうとした時にうまくいかないことがあります。 RCS 文字列はドル記号 ($) で囲まれており、 $FreeBSD や $RCS などで始まります。

diff(1) の再帰 (-r) フラグを使って再帰的なパッチを作るのは大変結構なのですが、でき上がったパッチは必ず目でチェックして余計なゴミが入っていないことを確認してください。
よくあるのはバックアップファイル同士の変更点、あるいは Imake や GNU configure を使うソフトウェアの Makefile の変更点が入っている場合などです。 また configure.in を編集して autoconf を使って configure を作り直すときには、 configure の diff は含めずに (それらは良く数千行におよぶことがあります)、 USE_AUTOCONF=yes を定義して configure.in の diff をとってください。

ファイルをまるごと消す場合には、 パッチを使わずに post-extract ターゲットで消す方が簡単です。
できあがった差分に満足したら、
それらをソースのファイルごとに別々のパッチファイルに分割してください。

4.5. コンフィグレーション

カスタマイズのために追加したいコマンドがあれば、 configure という名前のスクリプトに入れて scripts サブディレクトリに置いてください。 上で述べたように、pre-configure あるいは post-configure という Makefile ターゲットや、 スクリプトで処理することもできます。

4.6. ユーザからの入力の扱い

もし、その port がビルド、コンフィグレーション、またはインストールの際にユーザからの入力を必要とするならば、 Makefile 中で IS_INTERACTIVE を設定しなければなりません。 これにより、ユーザが環境変数 BATCH を セットしている場合には、この port の処理がスキップされるので "夜間の無人ビルド" が実行可能になります。 (逆に環境変数 INTERACTIVE がセットされていると、 ユーザからの入力を必要とする port だけが コンパイルされます)。これは、連続して ports をビルドするマシン群で、 無駄になる時間を大きく減らします。

もし、適切なデフォルト設定が存在するのであれば、 PACKAGE_BUILDING 変数をチェックして、それが設定されている場合には ユーザ入力のスクリプトを起動しないようにしてください。こうすることによって、我々 ports 管理者が CDROM や FTP に 置く package を作成することができます。

Chapter 5. Makefile の作成

Makefile の作成は非常に単純です。繰り返しますが、始めるまえに既存の例を見ておくことを推奨します。 また、このハンドブックには Makefile のサンプルがあります。 それを見て、Makefile 内の変数の順番や空行を入れるところなどの参考にしてください。 そうすると他の人々にも読みやすいものとなります。

では、Makefile を設計するときに 問題となるところを順に追って見てみましょう。

5.1. オリジナルのソース

ソースは foozolix-1.2.tar.gz といった名前の 標準的な gzip された tar ファイルの形式で DISTDIR に置かれていますか? そうなっていれば、次のステップに進めます。 異なっている場合には、変数 DISTNAME, EXTRACT_CMD, EXTRACT_BEFORE_ARGS, EXTRACT_AFTER_ARGS, EXTRACT_SUFX, DISTFILES のうちいくつかを書き換える必要があります。 どれだけ変更しないといけないかは、その port の配布ファイルが どの程度標準からかけはなれているかによります (最もよくあるのは gzip ではなく普通の compress コマンドで tar ファイルが圧縮されている場合で、そのときは EXTRACT_SUFX=.tar.Z とするだけです)。

最悪の場合には、自分で do-extract ターゲットを作成して、 デフォルトを上書きすることもできます。しかし、そこまでする必要があることはめったにないでしょう。

5.2. 名前の付け方

Port の Makefile のはじめの部分で port に名前をつけ、バージョン番号を記述し、適切なカテゴリに載せます。

5.2.1. PORTNAME および PORTVERSION

PORTNAME には port の名前の基幹部分を入れ、 PORTVERSION には port のバージョン番号を入れます。

5.2.2. PORTREVISION および PORTEPOCH

5.2.2.1. PORTREVISION

PORTREVISION 変数は単調増加する値です。 PORTVERSION が増加した時 (つまり、新しいオフィシャルベンダーリリースが行なわれた時) には いつでも 0 にリセットされます。また、その値が 0 でない場合には package 名に追加されます。 PORTREVISION の変更は、(例えば pkg_version(1) 等の) 自動化ツールが、 新たな package が入手できることを示すのに使われます。

その port から作られる package の内容や構造に 大きな影響を与える変更を行なった時には、PORTREVISION を増やしてください。

PORTREVISION を上げる必要がある変更の例:

- セキュリティ上の脆弱性やバグを修正するため、または その port に新しい機能を追加するためのパッチの追加。
- package のコンパイル時オプションの有効化や 無効化のための port の Makefile の変更。

- パッキングリストの変更や、package のインストール時の挙動の変更 (たとえば、ssh のホストキーのような package の 初期データを生成するスクリプトの変更など)。
- その port が依存する共有ライブラリのバージョンを上げる場合 (新しいバージョンの共有ライブラリがインストールされた後に、そのライブラリに依存していた古い package をインストールを試みる場合、 その package は新しい libfoo.(x+1) ではなく 古い libfoo.x を探そうとするため、インストールに失敗します。 (訳注: そのため、PORTREVISION を上げた package を 作成する必要があるわけです))。
- ひそかに port 配布ファイルの変更が行なわれ、その機能に大きな変化があった場合。つまり、distinfo の修正を 必要とするような配布ファイルの変更が行なわれ、 新旧のバージョンの `diff -ru` を取ると 些細とは言えない変更が認められるにもかかわらず、オリジナルのバージョン番号が変更されていないことから `PORTVERSION` の変更は難しい場合。

PORTREVISION を上げる必要の無い変更の例:

- 生成される package に機能の変化が起らないような port スケルトンのスタイル変更。
- 生成される package に影響しないような `MASTER_SITES` その他の port に対する機能変更。
- 誤植の修正などの些細な変更で、その package のユーザがアップグレードを必要とするほどには重要でないパッチ。
- 以前にはコンパイルが通らなかった package を ビルド可能にするための修正 (その port が以前にビルド可能だった プラットフォームにおいて、その変更により何らかの機能的な違いが発生しない場合に限ります)。 `PORTREVISION` は package の内容を反映したものなので、その package が以前にビルド可能でなかったのなら、変更を示すために、 `PORTREVISION` を増やす必要はありません。

経験的な判断方法としては、ある port にコミットされた変更が (それが強化や修正によるものであれ、新しい package による実質的な効能であれ)、アップデートすることにより、 誰もが利益を受けるような何かかどうか、また定期的に ports ツリーを更新している人に更新を強制するということに値するか自問してみることです。
もし答がイエスであれば、 `PORTREVISION` を上げるべきでしょう。

5.2.2.2. `PORTEPOCH`

ソフトウェアのベンダや FreeBSD の port 作成者は、以前のものよりも小さい数字のバージョン番号をつけたソフトウェアをリリースするといった、何か馬鹿げたことをすることが時々あります。 例をあげると、ある port が foo-20000801 から foo-1.0 になるといった具合です (数字として見ると 20000801 は 1 よりも大きいため、間違って前者の方が新しいバージョンとして扱われてしまいます)。

このような場合には `PORTEPOCH` バージョンを増やしてください。 上のセクション 0 で説明したように、`PORTEPOCH` がゼロでない場合には、 それがパッケージ名の後ろにつけられます。絶対に `PORTEPOCH` を減らしたり、ゼロにリセットしてはいけません。 さもないと、以前に作成された package との比較に失敗する (つまり、その package が古くなっていることがわからない) ためです: 新しいバージョン番号 (上の例では`1.0,1`) は 依然として前のバージョン番号 (20000801) よりも数字としては小さいのですが、自動化ツールが サフィックス `,1` を特別扱いすることで、 以前の package には明示されていないサフィックス `,0` よりも新しいことがわかります。

誤って `PORTEPOCH` を削除したりリセットしたりすると、終わりのない悲劇に見舞われます。

上記の議論を理解できないなら、 わかるまで議論をたどるかメーリングリストで質問してください。

大多数の ports では、PORTEPOCH が 必要になることは まず無いものと考えられています。 また、注意深く PORTVERSION を使用することで、 そのソフトウェアの将来のリリースがバージョン構造を変更する必要が出てきた場合にも、 多くの場合前もって対応しておくことができるでしょう。 しかし、"スナップショット "リリースのように、 オフィシャルなバージョン番号を持たないベンダーリリースが行なわれた時には、 FreeBSD 版の port 作者によるケアが必要になります。 そういったリリースに対し、 リリース日付を使ったラベルを付けたいという誘惑にかられることがあるでしょうが、 そうすると新しい"オフィシャル"リリースが行なわれた時に、 上の例で示したような問題が起きることでしょう。

例えば、あるソフトウェアのスナップショットリリースが 20000917 に行なわれ、以前のバージョン番号が 1.2 だったとすると、 そのスナップショットの PORTVERSION には 20000917 ではなく 1.2.20000917 か何か、そのような番号を 指定するのが良いでしょう。 そうしておけば、例えばバージョン番号 1.3 として後続のリリースが 行なわれた場合にも、大小関係が崩されずにすむわけです。

5.2.2.3. PORTREVISION と PORTEPOCH の使い方の例

gtkmumble の port, バージョン 0.10 が ports collection にコミットされます。

```
PORTNAME=       gtkmumble
PORTVERSION=    0.10
```

PKGNAME は gtkmumble-0.10 になります。

ローカルな FreeBSD パッチを必要とする セキュリティホールが発見されました。 それに合わせて PORTREVISION を増やします。

```
PORTNAME=       gtkmumble
PORTVERSION=    0.10
PORTREVISION=   1
```

PKGNAME は gtkmumble-0.10_1 になります。

ベンダから 0.2 という番号が振られた 新バージョンがリリースされます (これにより、 作者は 0.10 という番号を "0.9 の次という意味ではなく"、 実際には 0.1.0 のつもりで 使用していたことがわかります - あらら、今さら遅すぎる)。 新しいマイナーバージョン 2 は数字として以前のバージョン番号 10 より小さいので、 強制的に新しい package "の方が新しい"と認識させるため PORTEPOCH を増やす必要があります。 これは新しいベンダーリリースなので、 PORTREVISION は 0 にリセット (または Makefile から削除) されます。

```
PORTNAME=       gtkmumble
PORTVERSION=    0.2
PORTEPOCH=      1
```

PKGNAME は gtkmumble-0.2,1 になります。

次のリリースは 0.3 です。 PORTEPOCH は減少することが無いため、
今度のバージョン変数は次のようになります:

```
PORTNAME=       gtkmumble
PORTVERSION=    0.3
PORTEPOCH=      1
```

PKGNAME は gtkmumble-0.3,1 になります。

もし、このアップグレードによって PORTEPOCH が 0 に リセットされたとすると、3 は数字として 10 よりも小さいため、 gtkmumble-0.10_1 の package をインストールした誰かは gtkmumble-0.3 の package の方が新しいことに気がつかないことになるでしょう。 これが、そもそも PORTEPOCH が導入された肝心な理由です。

5.2.3. PKGNAMEPREFIX および PKGNAMESUFFIX

二つのオプション変数 PKGNAMEPREFIX と PKGNAMESUFFIX は、 PORTNAME および PORTVERSION と結合され、PKGNAME を ${PKGNAMEPREFIX}${PORTNAME}${PKGNAMESUFFIX}-${PORTVERSION} として定義します。この時、適切な package 名を選ぶための ガイドラインに沿っているかどうかを確認してください。特に、PORTVERSION 中に ハイフン (-) を使用することは禁止されています。 また、package 名に *language-* もしくは *-compiled.specifics* 部分が 含まれる場合、それぞれ PKGNAMEPREFIX と PKGNAMESUFFIX を使用してください。 これらを PORTNAME の一部としてはいけません。

5.2.4. package 名についての規則

package の名前は以下のルールにしたがってつけてください。 これは package のディレクトリを見やすくするためで、 既に何千ものパッケージがありますし、目を痛めてしまうようだとユーザはそっぽを向くでしょう。

package の名前は以下のようにしてください。 言語-名前-オプションバージョン.番号

package 名は ${PKGNAMEPREFIX}${PORTNAME}${PKGNAMESUFFIX}-${PORTVERSION} というように定義されています。 変数がこの書式と適合していることを確認してください。

1. FreeBSD はユーザの慣れ親しんだ言語のサポートに力を入れています。 特定の言語のための port の package 名には 言語- に ISO-639 で定義されている言語名の略称を入れてください。たとえば日本語なら ja、 ロシア語なら ru、 ベトナム語なら vi、 中国語なら zh、 韓国語ならば ko、ドイツ語なら de といった具合です。

   port がある言語地域に特化したものである場合には、さらに二文字の国名コードを付加してください。 たとえば合衆国英語圏は en_US となり、スイスのフランス語圏は fr_CH となります。

   言語- 部分は、 PKGNAMEPREFIX 変数に 定義されなければなりません。

2. 名前の部分の最初の文字は 小文字でなければなりません。

(名前の残りの部分は大文字を含んでいても構わないため、
大文字を含んだソフトウェア名を変換する際の規則は、 あなた自身の裁量に任されています。) perl 5 のモジュールでは先頭に p5- を付け、 二重コロン (::) のセパレータをハイフン (-) に置きかえる習慣になっています。 たとえば Data::Dumper は p5-Data-Dumper になります。
また、そのソフトウェアの名前として通常使われるものに番号、
ハイフン、あるいは下線が入っている場合には、 それらを使うことも構いません (kinput2など)。

3. コンパイル時に環境変数や make の引数などで ハードコードされたデフォルトを変えてコンパイルできる場合、 *-compiled.specifics* にそのコンパイル時のデフォルトを入れてください (ハイフンはあってもなくてもかまいません)。
用紙のサイズ、あるいはフォントの解像度などがこれにあたります。

   -compiled.specifics 部分は、 PKGNAMESUFFIX 変数に定義されなければなりません。

4. バージョン番号は数字とアルファベットからなり、 ピリオド (.) で区切ります。
アルファベットは二文字以上続けてはいけません。 唯一の例外は"パッチレベル"を意味する文字列 pl で、 それ以外にバージョン番号がまったくついていない場合にのみ使うことができます。
もしソフトウェアのバージョンに "alpha", "beta", "rc" や "pre" といった文字列が含まれるなら、ピリオドの後に最初の一文字をとってください。
これらの後に、さらにバージョン文字列が続く場合には、
一文字のアルファベットの後にピリオドをつけずに番号を続けます。

   この考え方は、 バージョン文字列を見て簡単に ports を並べられるようにするためのものです。
特に、バージョン番号の各部分が必ずピリオドで区切られていること、
また日付の部分がバージョン文字列の一部となっている場合には yyyy.mm.dd という書式を使っていることを確認してください。 dd.mm.yyyy や、2000 年問題に対応していない yy.mm.dd という書式を使ってはいけません。

では、DISTNAMEを正しい PKGNAME に直す例を見てみましょう:

以下は、ソフトウェアの作者が決めた名前から 適切な package 名に変換する方法を示した (実際の) 例です。

配布名	PKGNAMEPREFIX	PORTNAME	PKGNAMESUFFIX	PORTVERSION	理由
mule-2.2.2	(空)	mule	(空)	2.2.2	変更の必要はありません
XFree86-3.3.6	(空)	XFree86	(空)	3.3.6	変更の必要はありません
EmiClock-1.0.2	(空)	emiclock	(空)	1.0.2	プログラム一つだけの時は小文字のみ
rdist-1.3alpha	(空)	rdist	(空)	1.3.a	alpha のような文字列は使えない
es-0.9-beta1	(空)	es	(空)	0.9.b1	alpha のような文字列は使えない

配布名	PKGNAMEPREFIX	PORTNAME	PKGNAMESUFFIX	PORTVERSION	理由
mailman-2.0rc3	(空)	mailman	(空)	2.0.r3	rc のような文字列は使えない
v3.3beta021.src	(空)	tiff	(空)	3.3	なんなんでしょう ;)
tvtwm	(空)	tvtwm	(空)	pl11	バージョン番号は必ず必要
piewm	(空)	piewm	(空)	1.0	同上
xvgr-2.10pl1	(空)	xvgr	(空)	2.10.1	pl が使えるのは、他にメジャー/マイナーバージョン番号がない場合のみ
gawk-2.15.6	ja-	gawk	(空)	2.15.6	日本語バージョン
psutils-1.13	(空)	psutils	-letter	1.13	コンパイル時に用紙のサイズを指定
pkfonts	(空)	pkfonts	300	1.0	300dpiフォント用の package

オリジナルのソースにまったくバージョン情報が見当たらず、
また原作者が新しいバージョンをリリースする可能性が低いときには、 バージョン番号として 1.0 を使えばいいでしょう (上記の piewm の例がこれにあたります)。
そうでない場合には原作者に聞くか、日付 (yyyy.mm.dd) を使うなどしてください。

5.3. カテゴリ分類

5.3.1. CATEGORIES

パッケージが作成されると /usr/ports/packages/All に置かれ、一つ以上の /usr/ports/packages のサブディレクトリからリンクが張られます。 これらのサブディレクトリの名称は、CATEGORIES 変数で指定されます。これは、ユーザが FTP サイトや CDROM のパッケージの山から探し出すのを容易にするためのものです。 既存のカテゴリを参照して、 あなたの port にふさわしいものを選んでください。

また、このリストは、その port が ports ツリーのどこにインポートされるかも決定します。
ここに複数のカテゴリを指定すると、port
のファイルは最初のカテゴリ名のサブディレクトリに置かれることになります。
適切なカテゴリの選択方法についてはカテゴリ節をご覧ください。

あなたが作成した port が、本当に既存のどのカテゴリにも当てはまらない場合には、新たにカテゴリ名を作成することもできます。 その場合、新しいカテゴリを提案するメールを FreeBSD ports メーリングリスト 宛に送ってください。

しかし、一般的にはあなたが提案したカテゴリにあてはまる ports
が一握りではすまない場合でなければ、 あなたの提案は却下されるでしょう。

時々、カテゴリを 2 階層構造や、
何か他のキーワードを利用した構造に再構成することを提案する人がいます。
今日まで、その提案はどれも実現しませんでした。
なぜなら、その構成を実現することは簡単なのですが、既存の Ports Collection
全体を構成しなおしたものに合わせて改修する労力は、
控え目にいっても気が遠くなるものだからです。 こういうアイディアを送る前に、
それらの提案の歴史をメーリングリストのアーカイブで調べてください。
さらに、動作するプロトタイプを示せと言われるのに対する準備をしておきましょう。

5.3.2. 現在のカテゴリのリスト

ここに現在の port のカテゴリの一覧を示します。 アスタリスク(*) が付いているものは仮想 (*virtual*) カテゴリです - これらには対応するサブディレクトリが port ツリーにはありません。 これらは第 2 の補助的なカテゴリとして、 検索目的にしか使われません。

仮想カテゴリでないものは、 そのサブディレクトリ内の pkg/COMMENT
に一行の記述があります (例: archivers/pkg/COMMENT)。

カテゴリ	説明	**Notes**
accessibility	障害を持ったユーザの役に立つ ports	
afterstep*	AfterStep ウィンドウマネージャをサポートする ports	
arabic	アラビア語サポート	
archivers	アーカイブ用ツール	
astro	天文学関連の ports	
audio	サウンドをサポートする ports	
benchmarks	ベンチマークユーティリティ	
biology	生物学関連のソフトウェア	
cad	CAD ツール	
chinese	中国語サポート	
comms	通信ソフトウェア	ほとんどはシリアルポート用のソフトウェア
converters	文字コード変換	
databases	データベース	
deskutils	コンピュータが発明される以前に机上で使われていた道具	(訳注: いわゆるデスクトップユーティリティのこと)

カテゴリ	説明	Notes
devel	開発ユーティリティ	単にライブラリだからというだけで、 どうしてもここに置かなければならない理由があるのでない限り、 ライブラリをここに含めないでください。
dns	DNS 関連ソフトウェア	
editors	一般的なエディタ	特殊なエディタはそれぞれふさわしいセクションに入れます (たとえば数式エディタは math です)。
elisp	Emacs-lisp の ports	
emulators	他のオペレーティングシステム用のエミュレータ	端末エミュレータはここに含まれません - X ベースのものは x11 に、 テキストベースのものは機能によって comms か misc に分類されます。
finance	金融や財務会計関連のアプリケーション。	
french	フランス語サポート	
ftp	FTP クライアントとサーバユーティリティ	port が FTP と HTTP の両方に対応していれば、 ftp に入れ、第 2 カテゴリを www とします。
games	ゲーム	
german	ドイツ語サポート	
gnome*	GNOME プロジェクトの ports	
graphics	グラフィックユーティリティ	
haskell*	Haskell 言語関連のソフトウェア。	
hebrew	ヘブライ語サポート	
hungarian	ハンガリー語サポート	
ipv6*	IPv6 関連のソフトウェア	
irc	インターネットリレーチャット (IRC) 用ユーティリティ	
japanese	日本語サポート	
java	Java 言語関連のソフトウェア	
kde*	K Desktop Environment (kde) プロジェクトの ports	

カテゴリ	説明	Notes
korean	韓国語サポート	
lang	プログラミング言語	
linux*	Linux アプリケーションとサポートユーティリティ	
lisp*	Lisp 言語関連のソフトウェア	
mail	メールソフトウェア	
math	数値計算ソフトウェアやその他の数学ソフトウェア	
mbone	MBone アプリケーション	
misc	種々のユーティリティ	基本的に他のカテゴリに属さないものです。 これは他の仮想でないカテゴリを伴わない、唯一のカテゴリです。 `misc` と他のカテゴリが `CATEGORIES` 行に書かれている場合、 `misc` を削除して他のサブディレクトリにおいて良いという意味になります。 このカテゴリに置かれた ports は見落とされやすいので、可能な限り `misc` よりふさわしいカテゴリを探してください。
multimedia	マルチメディアソフトウェア	
net	種々のネットワークソフトウェア	
net-mgmt	ネットワーク管理ソフトウェア	
news	USENET ニュースソフトウェア	
offix*	OffiX suite の ports	
palm	Palm™ シリーズをサポートするソフトウェア	
parallel*	並列計算を行うアプリケーション	
pear*	Pear PHP フレームワーク関連の ports	
perl5*	実行に Perl バージョン 5 を必要とする ports	
picobsd	PicoBSD をサポートするための ports	
plan9*	Plan9 に由来するさまざまなソフトウェア	

カテゴリ	説明	Notes
polish	ポーランド語サポート	
portuguese	ポルトガル語サポート	
print	印刷ソフトウェア	DTP 用ツール (プレビューアなど) もここに分類されます。
python*	Python 言語関連のソフトウェア	
ruby*	Ruby 言語関連のソフトウェア	
russian	ロシア語サポート	
science	astro や biology, math 等、他のカテゴリにはあてはまらない科学関連の ports	
security	セキュリティ関連のユーティリティ	
shells	コマンドラインシェル	
sysutils	システムユーティリティ	
tcl76*	実行に Tcl バージョン 7.6 を必要とする ports	
tcl80*	実行に Tcl バージョン 8.0 を必要とする ports	
tcl81*	実行に Tcl バージョン 8.1 を必要とする ports	
tcl82*	実行に Tcl バージョン 8.2 を必要とする ports	
tcl83*	実行に Tcl バージョン 8.3 を必要とする ports	
textproc	テキスト処理ユーティリティ	DTP ツールはここではなく、print に分類されます。
tk42*	実行に Tk バージョン 4.2 を必要とする ports	
tk80*	実行に Tk バージョン 8.0 を必要とする ports	
tk81*	実行に Tk バージョン 8.1 を必要とする ports	
tk82*	実行に Tk バージョン 8.2 を必要とする ports	
tk83*	実行に Tk バージョン 8.3 を必要とする ports	
tkstep80*	実行に TkSTEP バージョン 8.0 を必要とする ports	

カテゴリ	説明	Notes
ukrainian	ウクライナ語サポート	
vietnamese	ベトナム語サポート	
windowmaker*	WindowMaker ウィンドウマネージャをサポートする ports	
www	World Wide Web 関連のソフトウェア	HTML 言語サポートもここに分類されます。
x11	X ウィンドウシステムとその関連ソフトウェア	このカテゴリは、 直接ウィンドウシステムをサポートするソフトウェアのみを対象とするものです。 通常の X アプリケーションをここに分類しないでください。 ほとんどは他の x11-* カテゴリ (下記参照) に分類されるべきです。 あなたの port が X アプリケーションで、 USE_XLIB を定義し (USE_IMAKE を定義すると自動的に定義されます)、 適切なカテゴリに分類してください。
x11-clocks	X11 用時計	
x11-fm	X11 用ファイルマネージャ	
x11-fonts	X11 フォントとフォントユーティリティ	
x11-servers	X11 サーバ	
x11-toolkits	X11 ツールキット	
x11-wm	X11 ウィンドウマネージャ	
zope*	Zope サポート	

5.3.3. 適切なカテゴリの選択

多くのカテゴリに重なるので、どれを"第一
"カテゴリにするかを決めなければならないことがたびたびあるでしょう。
これをうまく決めるルールがいくつかあります。
以下はその優先順のリストで、優先度の高いものから低いものの順に書いてあります。

- 言語特有のカテゴリがまず最初です。 たとえば日本語の X11 のフォントをインストールする port の場合、 CATEGORIES 行は japanese x11-fonts となるでしょう。
- より特徴的なカテゴリが、 一般的なカテゴリより優先されます。 たとえば、HTML エディタの場合は www editors となります。 これを逆順にはしないでください。 また、 port が irc, mail, mbone, news, security, www のいずれかに属する場合には net

は暗黙のうちに含まれますので、入れるべきではありません。

- x11 を第二カテゴリにするのは第一カテゴリが自然言語の場合のみにしてください。 特に X のアプリケーションには x11 を指定しないでください。
- Emacs のモードは、 そのモードで対応しているアプリケーションと同じ ports カテゴリに置くようにして、 editors には置かないでください。 例えば、あるプログラミング言語のソースファイルを編集するための Emacs モードは、 lang に置くべきです。
- もし、あなたの port が他のどのカテゴリにも属しない場合には misc にしてください。

もし、あなたがカテゴリについて自信が持てない場合には、 そのことを send-pr(1) する時に書き加えてください。 そうすればインポートする前にそれについて議論できます (もしあなたがコミッターであれば、 そのことを FreeBSD ports メーリングリスト に送って先に議論するようにしてください。 新しい port が間違ったカテゴリに import されて、 すぐ移動されることがあまりに多いのです。そうなると、
ソースリポジトリのマスターが不要で好ましくない膨れ方をしてしまいます。

5.4. 配布ファイル

Makefile の第二の部分では、 その port をビルドするためにダウンロードしなければならないファイルと、
それをどこからダウンロードできるか説明しています。

5.4.1. DISTNAME

DISTNAME は製作者が決めたソフトウェアの名前です。 デフォルトでは DISTNAME は ${PORTNAME}-${PORTVERSION} になりますので、 必要にな場合だけ書き換えるようにしてください。 DISTNAME は二つの場所でしか使われません。 一つ目は配布ファイルリスト (DISTFILES) のデフォルト ${DISTNAME}${EXTRACT_SUFX} で、二つ目は配布ファイルが展開されるサブディレクトリ WRKSRC のデフォルト work/${DISTNAME} です。

PKGNAMEPREFIX や PKGNAMESUFFIX は DISTNAME に影響を与えません。 また、元のソースアーカイブが ${PORTNAME}-${PORTVERSION}${EXTRACT_SUFX} という名前ではないのに、WRKSRC を work/${PORTNAME}-${PORTVERSION} と設定しているなら、おそらく DISTNAME はそのままにしておく必要があることに注意してください - DISTNAME と WRKSRC の両方を (そして おそらく EXTRACT_SUFX も) セットするよりは、DISTFILES を定義する方が楽でしょう。

5.4.2. MASTER_SITES

元になる配布ファイルを指し示す、FTP/HTTP の URL のファイル名を除いた部分を MASTER_SITES に設定します。 最後にスラッシュ (/) をつけることをお忘れなく!

このシステム上に配布ファイルが見つからなかった場合、 make マクロは FETCH を使ってこの変数に指定されたサイトから配布ファイルを取得しようとします。

このリストには、 できれば異なる大陸に存在する複数のサイトを入れておくことが推奨されています。 これにより、広域ネットワークのトラブルに対する耐性を高めることができます。

さらに私たちは、自動的に最も近いマスタサイトを判断して、
そこから取ってくるメカニズムの導入を計画しています。
複数のサイトがあれば、この取組を大きく助けることになります。

元になる tar ファイルが X-contrib や GNU, Perl CPAN 等の有名なアーカイブサイトに置かれている場合には、 MASTER_SITE_* を使ってこれらのサイトを簡潔に (例えば MASTER_SITE_XCONTRIB とか、 MASTER_SITE_PERL_CPAN のように) 指定することができます。 MASTER_SITES を これらの変数の一つにセットし、 サイト内でのパスを MASTER_SITE_SUBDIR に指定するだけです。 以下に例を示します。

```
MASTER_SITES=         ${MASTER_SITE_XCONTRIB}
MASTER_SITE_SUBDIR=   applications
```

これらの変数は /usr/ports/Mk/bsd.sites.mk で定義されています。
いつでも新しい項目が追加されて行きますので、 port を提出する前に このファイルの最新版をチェックするように心掛けてください。

ユーザは /etc/make.conf 中で MASTER_SITE_* 変数を上書きすることもできます。
そうすることで、これらの有名なアーカイブそのものではなく、
好みのミラーサイトを使用することができます。

5.4.3. EXTRACT_SUFX

配布ファイルが 1 つで、 圧縮方式を示すのに普通と異なる接尾辞を使っていたら、 EXTRACT_SUFX を設定してください。

例えば、配布ファイルがより一般的な foo.tar.gz ではなく、 foo.tgz となっていたら、次のように書きます。

```
DISTNAME=      foo
EXTRACT_SUFX=  .tgz
```

USE_BZIP2 と USE_ZIP 変数を設定すると、EXTRACT_SUFX は必要に応じて自動的に .bz2 または .zip に設定されます。 どちらも設定されていなければ、EXTRACT_SUFX は .tar.gz に設定されます。

EXTRACT_SUFX と DISTFILES を両方設定する必要はありません。

5.4.4. DISTFILES

時々、ダウンロードするファイルの名称が port の名称とまったく似ていないことがあります。たとえば、 source.tar.gz などと名づけられていることもあるでしょう。
ほかに、ソースコードがいくつかのアーカイブに分かれていて、
そのすべてをダウンロードしなければならないならないこともあります。

この場合、DISTFILES に、ダウンロードしなければならないファイルすべてのリストを、スペースで区切って設定してください。

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

明示的に設定されていない場合、 DISTFILES は ${DISTNAME}${EXTRACT_SUFX} に設定されます。

5.4.5. EXTRACT_ONLY

DISTFILES の一部だけを展開すべき
(例えば、一方がソースコードで、もう一方は圧縮されていない文書という)
場合、展開しなければならないファイル名を EXTRACT_ONLY に設定してください。

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

どのDISTFILES も展開すべきではないなら、 EXTRACT_ONLY に空文字列を設定してください。

```
EXTRACT_ONLY=
```

5.4.6. PATCHFILES

その port が配布ファイルの他に FTP や HTTP で手に入る追加パッチを必要とする場合には、 PATCHFILES にはそのパッチのファイル名を、 PATCH_SITES にはそのファイルが置かれているディレクトリの URL をセットしてください。(書き方は MASTER_SITES と同じです。)

そのパッチに記録されているファイル名に余計なパス名がついていて、
ソースツリーのトップディレクトリ (つまり WRKSRC) からの相対パスになっていない場合には、
それに応じた PATCH_DIST_STRIP を指定してください。
たとえば、パッチ内のすべてのファイル名の先頭に、余計な foozolix-1.0/ がついている場合には、
PATCH_DIST_STRIP=-p1 としてください。

これらのパッチは圧縮されていても大丈夫です。 ファイル名が .gz や .Z
で終わる場合には、自動的に展開されるようになっています。

もしパッチが、ドキュメント等その他のファイルと一緒に gzip された tar
ファイルで配布されている場合には、単に PATCHFILES を使うだけではうまくいきません。
このような場合には、このパッチの tar ファイルの名前と場所を DISTFILES と MASTER_SITES
に追加しておきます。 それから、EXTRA_PATCHES 変数にそれらのパッチを指定すれば、 bsd.port.mk が
自動的にパッチを適用してくれます。 特に注意が必要なのは、パッチファイルを PATCHDIR
ディレクトリにコピーしてはならないことです - (訳注: port が CD-ROM
上に置かれている等の場合には、) そのディレクトリには書き込みができないかもしれません。

それが普通の gzip か compress された tar ファイルであれば、
通常のソースファイルと一緒にパッチ適用時までに展開されていますので、
明示的に展開する必要はありません。 もしパッチを DISTFILES
に追加した場合には、パッチを含むファイルが展開される際に、
そのディレクトリにある何かを上書きしないように注意してください。
さらに、コピーされたパッチファイルを削除するコマンドを pre-clean

> ターゲットに追加することを忘れないでください。

5.4.7. 異なるサイトやサブディレクトリからの複数の配布ファイルまたはパッチ (MASTER_SITES:n)

(これはいささか"高度な話題"です。この文書を初めて読む人は、最初はこの節を飛ばしてもよいでしょう)。

この節は MASTER_SITES:n や MASTER_SITES_NN と呼ばれる取得方法について説明しています。ここでは、この方式を MASTER_SITES:n と呼びます。

まず、背景を少し説明しておきましょう。OpenBSD には、DISTFILES と PATCHFILES 変数の両方に素敵な機能があります。ファイル、パッチの両方とも、後ろに :n (n は [0-9] のどれかになります) をつけてグループを指示できます。たとえば、

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

OpenBSD では、配布ファイル alpha は、通常の MASTER_SITES ではなく MASTER_SITES0 に、 beta は MASTER_SITES1 に結び付けられます。

これは、正しいダウンロードサイトを際限なく探す羽目になるのを減らせる、 興味深い機能です。

DISTFILES にファイルが 2 つ指定され、MASTER_SITES が 20 サイトあって、サイトはものすごく遅く、beta は MASTER_SITES 中のすべてのサイトに置かれていますが、alpha は 20 番目のサイトにしかないという場合を考えてください。 メンテナがあらかじめそのことを知っていたら、すべてのサイトを確認するのは無駄だと思いませんか?
楽しい週末のはじまりというわけにはゆきませんね。

イメージできたら、今度は DISTFILES や MASTER_SITES がもっと沢山あるのを想像してください。"distfiles 調査マイスタ"は、 ネットワーク負荷が緩和されることを喜ぶに違いありません。

次節からは、FreeBSD におけるこのアイディアの実装について説明します。OpenBSD の考え方を多少改良しています。

5.4.8. 簡単な説明

この節では、複数の配布ファイルやパッチを、
異なるサイトやサブディレクトリから細かく分けて取得する簡単な設定を示します。
ここでは、単純化した MASTER_SITES:n の使い方を説明します。ほとんどの場面ではこれで十分です。さらに詳しいことを知りたければ、次の節をお読みください。

アプリケーションによっては、
いくつもの異なるサイトからダウンロードする複数の配布ファイルからなっているものがあります。
たとえば、Ghostscript は、中核部のプログラムと、
ユーザのプリンタに応じて使い分けられる多数のドライバファイルからなっています。
このドライバファイルの一部は中核部と共に配布されますが、
多くはさまざまなサイトからダウンロードしなければなりません。

これに対応するため、DISTFILES の各項目の後ろには、コロンと"タグ名"をつけられるようになっています。MASTER_SITES

に設定されているそれぞれのサイトの末尾にも、コロンと、
そのサイトからダウンロードすべきファイルを示すためのタグを加えます。

たとえば、ソースコードが source1.tar.gz と source2.tar.gz の 2 つに分けられていて、2 つの別のサイトからダウンロードしなければならないアプリケーションを考えてみましょう。その port の Makefile には、各サイトに 1 つファイルがある場合の、簡単な MASTER_SITES:n の使用法 のような行があるとします。

例 *1.* 各サイトに *1* つファイルがある場合の、簡単な MASTER_SITES:n の使用法

```
MASTER_SITES=   ftp://ftp.example1.com/:source1 \
                ftp://ftp.example2.com/:source2
DISTFILES=      source1.tar.gz:source1 \
                source2.tar.gz:source2
```

複数の配布ファイルに同じタグがついていてもかまいません。先ほどの例に続いて、3 番目の配布ファイル source3.tar.gz があって、ftp.example2.com からダウンロードすべきだとしましょう。Makefile は 各サイトに 1 つ以上ファイルがある場合の、簡単な MASTER_SITES:n の使用法 のようになります。

例 *2.* 各サイトに *1* つ以上ファイルがある場合の、簡単な MASTER_SITES:n の使用法

```
MASTER_SITES=   ftp://ftp.example1.com/:source1 \
                ftp://ftp.example2.com/:source2
DISTFILES=      source1.tar.gz:source1 \
                source2.tar.gz:source2 \
                source3.tar.gz:source2
```

5.4.9. 詳しい説明

分かりました。前節の例ではあなたの要求を満足できなかったわけですね。
この節では、ファイルの取得を細かく制御する仕組み MASTER_SITES:n がどう働くかと、これを利用するために ports をどう変更すればよいかを詳しく説明します。

1. 要素の末尾に :n をつけることができます。ここで、*n* は `
   つまり、概念上はいかなる文字と数字からなる文字列でもよいのですが、われわれとしては、当面は `[a-zA-Z_][0-9a-zA-Z_] に制限します。

   さらに、文字列のマッチは大文字と小文字を区別します。つまり、n と N は別の文字として扱われます。

   しかし、default, all, ALL は特別な意味を与えられているので、末尾に付加するのには使えません (これは、ii 項で内部的に利用されています)。さらに、DEFAULT は特別な意味を持つ単語です (3 の項を確認してください)。

2. :n がついた要素は、グループ n に属し、:m がついた要素は、グループ m に属するということになります。

3. 接尾辞がついていない要素はグループに属しません。これは、特別なグループ DEFAULT に属しているとして扱われます。要素の後ろに DEFAULT をつけるのは、その要素を DEFAULT とそれ以外のグループに同時に割り当てたいのでなければ、冗長に過ぎません (5 の項を確認してください)。

   次の例はどちらも同じ意味ですが、最初の方が好ましいです。

   ```
   MASTER_SITES=   alpha

   MASTER_SITES=   alpha:DEFAULT
   ```

4. グループは相互排他ではありません。ひとつの要素が同時に複数のグループに属することができ、ひとつのグループには複数の要素が属することも、何も割り当てないこともできます。同じグループで何回も指定された要素は、単に複数回指定された要素ということになります。

5. ある要素を同時にいくつものグループに所属させたい時は、カンマ演算子 (,) が使えます。

   その都度別の接尾辞をつけて繰り返すかわりに、
   一度だけ接尾辞を指定して複数のグループを指定できます。たとえば、:m,n,o と書くと、その要素はグループ m, n および o に属することを示します。

   以下の例はすべて同等ですが、最後の形式がもっともよいでしょう。

   ```
   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. 任意のグループ内のサイトは、MASTER_SORT_AWK によって整列されます。MASTER_SITES と PATCH_SITES 内のすべてのグループについても同様に整列されます。

1. グループの概念は、変数 MASTER_SITES, PATCH_SITES, MASTER_SITE_SUBDIR, PATCH_SITE_SUBDIR, DISTFILES および PATCHFILES においても、下記の文法に従って使えます。

   a. MASTER_SITES, PATCH_SITES, MASTER_SITE_SUBDIR および PATCH_SITE_SUBDIR のすべての要素はスラッシュ / 記号で終端されていなければなりません。ある要素がどれかのグループに属しているなら、グループの接尾辞 :n は、終端記号 / のすぐ後にこなければなりません。MASTER_SITES:n の仕組みでは、終端記号 / があることで、:n が要素の有効な一部である場合と、:n がグループ n を示す場合の混同を避けることができます。以前は、MASTER_SITE_SUBDIR と PATCH_SITE_SUBDIR 要素のいずれにおいても終端記号 / は不要だったので、互換性を保つために、接尾辞の直前の文字が / でなければ、要素の接尾辞が :n であっても、グループの接尾語ではなく、要素の有効な一部分として扱われます。MASTER_SITE_SUBDIR における MASTER_SITES:n の詳細な使用法 と カンマ演算子、複数のファイル、複数のサイト、複数のサブディレクトリと合わせた MASTER_SITES:n の詳細な使用法 の両方をご覧ください。

例 3. MASTER_SITE_SUBDIR における MASTER_SITES:n の詳細な使用法

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

- グループ DEFAULT に属するディレクトリ → old:n
- グループ NEW に属するディレクトリ → new

例 4. カンマ演算子、複数のファイル、複数のサイト、 複数のサブディレクトリと合わせた MASTER_SITES:n の詳細な使用法

```
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
```

上の例は、次のような細かく分けた取得を実現します。
サイトは、利用される順番で挙げられています。

- file1 は次のサイトから取得されます。
  - MASTER_SITE_OVERRIDE
  - http://site1/directory/
  - http://site1/directory-one/
  - http://site1/directory-trial:1/
  - http://site2/
  - http://site7/
  - MASTER_SITE_BACKUP
- file2 は、file1 と同じグループに属しているので、 まったく同じように取得されます。
  - MASTER_SITE_OVERRIDE
  - http://site1/directory/
  - http://site1/directory-one/
  - http://site1/directory-trial:1/
  - http://site2/
  - http://site7/

        - MASTER_SITE_BACKUP
    - file3 は次のサイトから取得されます。
        - MASTER_SITE_OVERRIDE
        - http://site3/
        - MASTER_SITE_BACKUP
    - file4 は次のサイトから取得されます。
        - MASTER_SITE_OVERRIDE
        - http://site4/
        - http://site5/
        - http://site6/
        - http://site7/
        - http://site8/directory-one/
        - MASTER_SITE_BACKUP
    - file5 は次のサイトから取得されます。
        - MASTER_SITE_OVERRIDE
        - MASTER_SITE_BACKUP
    - file6 は次のサイトから取得されます。
        - MASTER_SITE_OVERRIDE
        - http://site8/directory-one/
        - MASTER_SITE_BACKUP

2. MASTER_SITE_SOURCEFORGE のように、bsd.sites.mk で定義される特別な変数をグループに割り当てるにはどうすればよいですか?

   MASTER_SITE_SOURCEFORGE と合わせた MASTER_SITES:n の詳しい使用法 をご覧ください。

   例 *5.* MASTER_SITE_SOURCEFORGE と合わせた MASTER_SITES:n の詳しい使用法

   ```
   MASTER_SITES=   http://site1/ ${MASTER_SITE_SOURCEFORGE:S/$/:sourceforge,TEST/}
   DISTFILES=      something.tar.gz:sourceforge
   ```

   something.tar.gz は、MASTER_SITE_SOURCEFORGE に含まれるあらゆるサイトから取得されます。

3. これを PATCH* 変数と組み合わせて使うにはどうすればよいでしょうか?

   すべての例で MASTER* 変数を使っていますが、PATCH_SITES と合わせた MASTER_SITES:n の簡単な使用法 にあるように、PATCH* 変数に対してもまったく同じように働きます。

例 6. PATCH_SITES と合わせた MASTER_SITES:n の簡単な使用法

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

5.4.10. ports について何が変更され、何が変わらないのでしょうか?

i. 現在のすべての ports はそのまま変わりません。 MASTER_SITES:n 機能のコードは、7 で述べた文法に従う :n のような形式が後ろについた要素がある場合だけ動作します。

1. port を make する際のターゲットにも変更はありません。 checksum, makesum, patch, configure, build 等です。 もちろん、do-fetch, fetch-list, master-sites それから patch-sites は例外です。
   - do-fetch は、新しくグループ分けの接尾辞のついた DISTFILES と PATCHFILES を設定します。それぞれが、対応する MASTER_SITES と PATCH_SITES を利用し、さらに対応する MASTER_SITE_SUBDIR と PATCH_SITE_SUBDIR を利用します。カンマ演算子、複数のファイル、複数のサイト、複数のサブディレクトリと合わせた MASTER_SITES:n の詳細な使用法 をご覧ください。
   - fetch-list は、do-fetch と同じようにグループを利用するということを除いて、以前の fetch-list のように動作します。
   - master-sites および patch-sites は、 (古いバージョンと互換性がなくなり) DEFAULT グループの要素を返すだけになっています。 実際は、それぞれ master-sites-default および patch-sites-default というターゲットを実行します。

     さらに、 MASTER_SITES や PATCH_SITES を直接確認するよりも、 master-sites-all または patch-sites-all のどちらかのターゲットを使う方がよいです。 また、将来のバージョンでも直接確認ができるかどうかは保証されていません。 これら新規 port ターゲットについては、B の項をご確認ください。

2. 新規の port ターゲット
   a. MASTER_SITES および PATCH_SITES のそれぞれについて、 グループ *n* の要素を表示する master-sites-n および patch-sites-n ターゲットがあります。たとえば、 master-sites-DEFAULT および patch-sites-DEFAULT のいずれも DEFAULT グループの要素を返し、 master-sites-test および patch-sites-test は test グループの要素を返します。

a. 以前の master-sites および patch-sites が行っていた作業を行う master-sites-all および patch-sites-all という新たなターゲットがあります。 これらのターゲットは、すべてのグループの要素をすべてが同じグループに属しているかのように返します。 ただし、master-sites-all および patch-sites-all のそれぞれについて、 DISTFILES や PATCHFILES で定義されているグループと同じ数だけ MASTER_SITE_BACKUP と MASTER_SITE_OVERRIDE を表示します。

5.4.11. DIST_SUBDIR

/usr/ports/distfiles ディレクトリ内をあまり散らかさないようにしてください。 たくさんのファイルを取ってくる port や、他の port と名前の衝突が起きる恐れのあるファイル

(Makefile など) がある場合には、 DIST_SUBDIR に port の名前 (${PORTNAME} か ${PKGNAMEPREFIX}${PORTNAME} を使うといいでしょう) を入れてください。すると DISTDIR がデフォルトの /usr/ports/distfiles から /usr/ports/distfiles/DIST_SUBDIR に変更され、取ってきたファイルはすべてそのサブディレクトリの中に置かれるようになります。

また、 ファイルを取ってくるときにバックアップサイトとして使われる ftp.FreeBSD.org のディレクトリ名にもこの変数の値が使われます (Makefile の中で DISTDIR を明示的に指定した場合、ローカルのファイルを置くところは変わりますが、 このサイトのディレクトリ名は変わりません。 必ず DIST_SUBDIR を使うようにしてください)。

この変数は Makefile 中で明示的に指定された MASTER_SITES には影響しません。

5.5. MAINTAINER

あなたのメールアドレスをここに入れてください。 お願いします。 :-)

MAINTAINER の値は、コメント部のない単一のアドレスしか受け付けられません。 user@hostname.domain という形式を利用してください。この項目には、
あなたの本名などの説明用のテキストは一切いれないでください。 (そうしても、ただ bsd.port.mk が混乱するだけです)。そういう情報は pkg-descr に書いてください。

保守担当者 (maintainer) の責任に関する詳細説明は、 Makefile 中の MAINTAINER のセクションを参照してください。

Port のメンテナがユーザからの更新要求に (主な公休日を除いて) 2 週間返答しなかったら、保守担当者の持ち時間が切れたとみなして、 保守担当者の明示的な了承なしに更新して構いません。保守担当者が 3 ヶ月以内に返答しない場合は、 その保守担当者は無断で不在にしているとみなして、問題となっている port の保守担当者を入れ替えても構いません。 例外となるは、ports 管理チーム <portmgr@FreeBSD.org> または セキュリティオフィサチーム <security-officer@FreeBSD.org> が保守しているものです。このグループが保守している port に対しては許可を得ずにコミットしてはいけません。

ports 管理チーム <portmgr@FreeBSD.org> は、何か理由があれば、誰かを保守担当から外したり、別の方を担当者にする権利を持ちます。 セキュリティオフィサチーム <security-officer@FreeBSD.org> は、セキュリティ上の理由で、保守担当者の権限を剥奪したり担当者を変更する権利を持ちます。

5.6. COMMENT

その port の 1 行の説明です。 コメントにはパッケージ名 (やソフトウェアのバージョン) を入れないでください。 コメントは大文字で始め、最後にピリオドは付けないでください。たとえば、こんな具合です。

```
COMMENT=	A cat chasing a mouse all over the screen
```

Makefile 中で、 COMMENT 変数は MAINTAINER 変数の直後においてください。

COMMENT 行は、port の 1 行の要約としてユーザに示されるので、 70 文字未満に抑えるようにしてください。

5.7. 依存関係

多くの port は他の port に依存しています。 必要なものすべてがユーザのマシン上に存在することを保証するために使用可能な、7 つの変数が用意されています。
よくあるケースのためにあらかじめ設定された依存変数に加え、
いくつかの依存関係の制御のための変数があります。

5.7.1. LIB_DEPENDS

その port が必要とする共有ライブラリを、この変数で指定します。 (訳注: libc 等、標準のライブラリは指定する必要がありません。) これは *lib:dir*:target という 組のリストです。 *lib* が共有ライブラリの名前、 *dir* が そのライブラリが見つからない場合に インストールされる port のディレクトリ、 *target*が そのディレクトリで呼ばれるターゲットです。 たとえば、

```
LIB_DEPENDS= jpeg.9:${PORTSDIR}/graphics/jpeg:install
```

と指定されていた場合、まずメジャーバージョンが 9 の jpeg 共有ライブラリがインストールされているかどうかを確認します。
インストールされていない場合には、ports ツリーの graphics/jpeg サブディレクトリに移動し、 *target* のコンパイルとインストールを行ないます。 *target* の部分は、 それが DEPENDS_TARGET (デフォルトでは install) と 等しいときには省略することができます。

先頭の *lib* の部分は ldconfig -r | grep -wF への引数になります。
この変数には正規表現を入れないようにしてください。

この依存関係のチェックは、 extract ターゲットと install ターゲットの中で、2 回行なわれます。 (訳注: これは、その port をビルドするマシンとインストールされるマシンが違う場合、どちらのマシンでもそのライブラリが利用できることを確認するためです)。
同様に、依存するライブラリの名前は package 中にも書き込まれていて、 pkg_add(1) 実行時にそのライブラリがユーザのシステムに存在していなければ、 自動的にインストールされます。

5.7.2. RUN_DEPENDS

この port の実行時に必要となるプログラム、またはファイルがあるときにはこの変数で指定します。これは *path:dir*:target という組のリストです。 *path* がファイルまたはプログラムの名前、 *dir* が それが見つからない場合にインストールされる port のディレクトリ、 *target* が そのディレクトリで呼ばれるターゲットです。 *path* の最初の文字がスラッシュ (/) の場合にはファイルかディレクトリとみなし、 存在するかどうか test -e を使ってチェックします。 そうでない場合には実行可能ファイルであると考えて、そのプログラムがユーザのサーチパス上にあるかどうか which -s を使って確認します。

たとえば Makefile に以下のように書いてあるとします。

```
RUN_DEPENDS=   ${LOCALBASE}/etc/innd:${PORTSDIR}/news/inn \
               wish8.0:${PORTSDIR}/x11-toolkits/tk80
```

まず、/usr/local/etc/innd というファイルかディレクトリが存在するか確認します。

存在しない場合には、ports ツリーの news/inn というサブディレクトリでビルドとインストールを行ないます。 さらに、wish8.0 というプログラムがユーザのサーチパス中にあるかどうか探します。 ない場合には同じく ports ツリーの x11-toolkits/tk80 というサブディレクトリでコンパイルとインストールを行ないます。

この例で、innd は実際にはプログラムです。
このように、プログラムであっても一般ユーザのサーチパスに
含まれているとは考えにくいところに置かれているものの場合には、
絶対パスで指定してください。

この依存関係は install ターゲット中でチェックされます。 また、pkg_add(1) によるインストールの際に、その package が依存するものがユーザのシステムに存在しない場合には自動的に追加インストールできるように、依存するものの名前も package 中に記録されます。 *target* の部分が DEPENDS_TARGET と同じ場合には、*target* の部分を省略することができます。

5.7.3. BUILD_DEPENDS

この port のビルド時に必要となるプログラム、 またはファイルがあるときにはこの変数で指定します。 RUN_DEPENDS と同様に、これは *path:dir*:target という組のリストです。たとえば、

```
BUILD_DEPENDS=unzip:${PORTSDIR}/archivers/unzip
```

と指定されていた場合、まず unzip という名前のプログラムがインストールされているかどうかを確認します。
インストールされていない場合には ports ツリーの archivers/unzip サブディレクトリに移動し、ビルドとインストールを行ないます。

ここで言う"ビルド"とは、
ファイルの展開からコンパイルまでのすべての処理を意味します。
この依存関係は、extract ターゲットの中でチェックされます。 *target* の部分は、DEPENDS_TARGET と同じ場合には省略することができます。

5.7.4. FETCH_DEPENDS

この port を取ってくるのに必要となるプログラム、またはファイルがあるときにはこの変数で指定します。 上の二つと同様に、これは *path:dir*:target という組のリストです。たとえば、

```
FETCH_DEPENDS=ncftp2:${PORTSDIR}/net/ncftp2
```

と指定されていれば、ncftp2 という名前のプログラムを探します。 見つからない場合には、ports ツリーの net/ncftp2 サブディレクトリでビルドとインストールを行ないます。

この依存関係は fetch ターゲット中でチェックされます。 *target* の部分は、 DEPENDS_TARGET と同じ場合には省略することができます。

5.7.5. EXTRACT_DEPENDS

この変数には、この port の展開に必要な実行ファイルや、他のファイルを指定します。
前の変数と同じく、これは *path:dir*:target のタプルの一覧です。たとえば、

```
EXTRACT_DEPENDS=
        unzip:${PORTSDIR}/archivers/unzip
```

は、unzip という実行形式のファイルがあるかどうか確認し、見つからなければ、ports ツリーの archivers/unzip サブディレクトリに降りてビルドおよびインストールを行います。

依存関係は extract ターゲットにおいて確認されます。*target* 部分が DEPENDS_TARGET と同じなら、省いて構いません。

この変数は、展開が働いておらず (デフォルトでは gzip を仮定しています)、USE_* で説明されている USE_ZIP や USE_BZIP2 を使っても動かない場合にだけ使ってください。

5.7.6. PATCH_DEPENDS

この変数は、この port がパッチを当てる際に必要とする実行ファイルや他のファイルを指定します。
前の変数と同じく、これは *path:dir*:target のタプルの一覧です。たとえば、

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

は、ports ツリーの java/jfc サブディレクトリに移動して、ビルドおよびインストールを行います。

依存関係は、patch ターゲットにおいて確認されます。*target* 部分が DEPENDS_TARGET と同じなら省略して構いません。

5.7.7. DEPENDS

上記のいずれにもあてはまらないような依存関係がある場合、または他の port がインストールされているだけではなく
ソースが展開されている必要がある場合には、この変数を使います。これは上記の四つと違い、特に "確認"するものが ありませんので、*dir*:target という形式のリストになります。*target* の部分は DEPENDS_TARGET と同じ場合には省略することができます。

5.7.8. USE_*

多くの ports に共通の依存関係をカプセル化するために、いくつもの変数が存在しています。

表 *1.* USE_* 変数

変数	意味
USE_BZIP2	その port の tarball は bzip2 で圧縮されています。
USE_ZIP	その port の tarball は zip で圧縮されています。

変数	意味
USE_GMAKE	その port をビルドするのに gmake が必要です。
USE_PERL5	その port をビルドしてインストールするのに perl 5 が必要です。 perl に関連して設定可能な他の変数については perl の利用 をご覧ください。
USE_X_PREFIX	その port は PREFIX ではなく X11BASE にインストールされます。 X11 に関連して設定可能な他の変数については、 X11 の利用 をご覧ください。
USE_AUTOMAKE	その port のビルドに GNU automake が使われます。automake に関わる他に設定可能な変数については、 automake - autoconf および libtool の利用 をご覧ください。
USE_AUTOCONF	その port のビルドに GNU autoconf が使われます。autoconf に関わる他に設定可能な変数については、 automake - autoconf および libtool の利用 をご覧ください。
USE_LIBTOOL	その port のビルドに GNU libtool が使われます。libtool に関わる他に設定可能な変数については、 automake - autoconf および libtool の利用 をご覧ください。
GMAKE	gmake が PATH に入っていない場合のフルパス
USE_BISON	その port のビルドに bison が使われます。
USE_SDL	その port のビルドや実行に SDL が使われます。 USE_SDL の使い方について、詳しくは SDL の利用 をご覧ください。
NO_INSTALL_MANPAGES	install.man ターゲットを使いません。

その ports が X Window System を必要とするのであれば、 USE_XLIB=yes を定義してください (これは USE_IMAKE が定義されていれば自動的に定義されます)。 BSD make ではなく GNU make を必要とする場合には USE_GMAKE=yes を、 GNU autoconf を実行する必要がある場合には USE_AUTOCONF=yes を、 最新の qt toolkit を使用する場合には USE_QT=yes を、 perl 言語のバージョン 5 を必要とする場合には USE_PERL5=yes を定義してください (特に最後のものは重要です。 FreeBSD のバージョンにより、基本システムに perl5 が含まれていたり、いなかったりします)。

5.7.9. 依存関係に関する注意

上で述べたように、依存する ports が必要になったときに呼ばれるデフォルトのターゲットは DEPENDS_TARGET で、そのデフォルトは install です。 これはユーザが使用する変数であり、 port の Makefile で定義するものではありません。 もし、その port が特別な方法で依存関係を扱う必要がある場合には、 DEPENDS_TARGET を再定義するのではなく *_DEPENDS 変数の :target 部分を使用してください。

make clean と入力したときには、 その port が依存する port も自動的に clean されます。そうならないようにしたい場合には、 環境変数 NOCLEANDEPENDS を設定してください。KDE, GNOME や Mozilla のように、再ビルドするのに時間がかかる port に依存している場合は特に望ましいかもしれません。

無条件に他の port に依存させるには、 BUILD_DEPENDS や RUN_DEPENDS の最初のフィールドに ${NONEXISTENT} という変数を指定してください。 これは、他の port のソースが必要なときのみ使用してください。 ターゲットも指定することで、コンパイルの時間を節約できる場合もあります。 たとえば

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

とすると、常に jpeg port のディレクトリに行ってソースの展開を行ないます。

あなたがやりたいことが他の方法ではできない場合以外には DEPENDS を使わないでください。これは常に他の port の作成を行ない (さらにデフォルトでは インストールも行ない)、package まで作成します。 この動作が本当に所望のものでしたら、 それを BUILD_DEPENDS と RUN_DEPENDS に書くべきでしょう - 少なくとも意図を明確にすることができます。

5.7.10. オプション選択可能な依存ライブラリ

巨大なアプリケーションの中には、
複数のコンフィギュレーションでビルドすることができるものがあります。
つまり、いくつもの外部ライブラリやアプリケーションの中の、 あるものが利用可能な場合に、それを拡張機能として使用するように設定することができるということです。
それらのライブラリやアプリケーションを、 必ずしもすべてのユーザが必要としているわけではありませんので、ports システムではどのコンフィギュレーションがビルドされるべきかを port 作者が決めるために使えるフックを用意しています。
これらを適切にサポートすることにより、ユーザをハッピーにしたり、 port 1 つ分のコストで 2 つまたはそれ以上の port を提供するのと同様の効率化を行なうことが可能です。

これらのフックのうちで最も簡単に使えるものは WITHOUT_X11 でしょう。 その port が X Window System のサポートありと、 サポートなしの設定でビルドできるのであれば、 通常は X Window System サポートありでビルドするべきでしょう。 ビルド時に WITHOUT_X11 が定義されていれば、 その時は X Window System サポートなしのバージョンが ビルドされるべきです。

GNOME 環境の様々なパーツも、そのようなノブ (フック) を持っていますが、それらは幾分使いにくいものです。 Makefile 中で その目的に使用される変数は WANT_* と HAVE_* になります。 そのアプリケーションが、以下に示されている依存ライブラリの一つについて、 サポートあり、なしの両方でビルドできる場合、Makefile には WANT_PKG をセットする必要があります。 そして、ビルド時に HAVE_PKG が定義されていれば PKG を使うバージョンがビルドされることになります。

現在、このような形でサポートされている WANT_* 変数は、 WANT_GLIB, WANT_GTK, WANT_ESOUND, WANT_IMLIB, そして WANT_GNOME です。

5.7.11. 致命的な依存の循環

Ports ツリーに循環する依存性を持ち込まないでください!

Ports の構築技術は循環依存性を許容していません。
循環依存させてしまうと、たちまちどこかの誰かがインストールしている FreeBSD を駄目にしてしまい、その数はまたたく間に増えて行きます。
この問題は見付けるのが非常に難しいです。 問題がありそうな場合は、その変更を行う前に `cd /usr/ports; make index` を実行するようにしてください。
この処理は古いマシンではかなり遅いかもしれませんが、 (あなたも含めて)
多くの人がその処理を行って嘆くことにならずに済ませられるでしょう。

5.8. 作業ディレクトリの指定

それぞれの port は作業ディレクトリに展開されるので、
作業ディレクトリは書き込み可能でなければなりません。 Ports システムは、デフォルトでは `DISTFILES` が `${DISTNAME}` というディレクトリに展開されると想定します。 つまり、次のように設定していたら、

```
PORTNAME=       foo
PORTVERSION=    1.0
```

その port の配布ファイルの内容は、最上位のディレクトリが foo-1.0 で、
残りのファイルはそのディレクトリの下に置かれているということです。

そうでない場合に上書きできる変数がたくさんあります。

5.8.1. WRKSRC

この変数は、
アプリケーションの配布ファイルが展開された時に作成されるディレクトリの名称を示します。
前の例で、(foo-1.0 ではなく) foo というディレクトリに展開されるなら、

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

または、

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

と書いてください。

5.8.2. NO_WRKSUBDIR

その port がサブディレクトリに展開しないのであれば、 それを示すために `NO_WRKSUBDIR` を設定してください。

```
NO_WRKSUBDIR= yes
```

5.9. CONFLICTS

あなたが作成している package が他の packageと (ファイルの衝突や実行時の非互換性により) 共存できないのであれば、CONFLICTS 変数にその package の名称を挙げてください。 * や ? のようなシェルの補完が利用できます。package 名は /var/db/pkg にあるのと同じ形式で並べてください。

5.10. ビルドのメカニズム

そのソフトウェアがビルドの際に GNU make を使う場合には、USE_GMAKE=yes をセットしてください。 configure を使う場合には、 HAS_CONFIGURE=yes をセットしてください。 GNU configure を使う場合には、 GNU_CONFIGURE=yes をセットしてください (これにより HAS_CONFIGURE もセットされます)。 configure に追加の引数を渡したい場合には、 追加部分を CONFIGURE_ARGS に指定してください。 (デフォルトの引数リストは、GNU configure では --prefix=${PREFIX} に、 GNU でない configure では空リストになります。) GNU autoconf を使う場合には、 USE_AUTOCONF=yes をセットしてください。 これにより GNU_CONFIGURE もセットされ、 configure を実行する前に autoconf が実行されます。

もしそのパッケージが GNU configure を使っていて、作成された実行形式のファイルが i386-portbld-freebsd4.7-*appname* のような"奇妙な"名称だった場合は、さらに CONFIGURE_TARGET を上書きして、新しいバージョンの autoconf で生成されたスクリプトが要求する方法でターゲットを指定する必要があります。 Makefile の GNU_CONFIGURE=yes 行のすぐ後に次の行を追加してください。

```
CONFIGURE_TARGET=--build=${MACHINE_ARCH}-portbld-freebsd${OSREL}
```

そのソフトウェアが X Window System のアプリケーションなどで、 imake を使って Imakefile から Makefile を作成する場合には、 USE_IMAKE=yes を指定してください。 そうするとコンフィグレーションステージで自動的に xmkmf -a が実行されます。 もし -a フラグが問題を引き起こすなら、 さらに XMKMF=xmkmf をセットしてください。 もし、その port が imake を使用するけれども install.man ターゲットを持たない場合には、 NO_INSTALL_MANPAGES=yes をセットしてください。
ついでに、そのソフトウェアの作者を探し出して八つ裂きにするといいでしょう。 (--#)_

そのソフトウェアの元々の Makefile が all 以外のものをメインのターゲットとしている場合には、それを ALL_TARGET に指定してください。 install と INSTALL_TARGET も同様です。

Chapter 6. 特別な配慮

port を作成する場合、考慮しなくてはいけないことが他にもいくつかあります。
このセクションでは、それらのうちでも特によくあることについて説明します。

6.1. 共有ライブラリ

その port が共有ライブラリのインストールを行なう場合、make 変数 INSTALLS_SHLIB を定義してください。これにより、bsd.port.mk が post-install ターゲットの実行時に新しいライブラリがインストールされたディレクトリ (通常は PREFIX/lib) に ${LDCONFIG} -m を実行し、共有ライブラリキャッシュへの登録が行なわれるようになります。また、この変数が定義されている場合、共有ライブラリを インストールしたユーザがそれをすぐに使い始められるように、また、削除の際には そのライブラリが まだ存在しているとシステムに誤認されないように、適切な @exec /sbin/ldconfig -m と @unexec /sbin/ldconfig -R のペアが pkg-plist ファイルに 指定されているように扱われます。

必要であれば、共有ライブラリがインストールされるディレクトリのリストを格納する make 変数 LDCONFIG_DIRS を定義することにより、新しいライブラリがインストールされるデフォルトの位置を上書きすることも可能です。例えば、その port が共有ライブラリを PREFIX/lib/foo と PREFIX/lib/bar に インストールする場合、Makefile で以下の記述を使用することができます:

```
INSTALLS_SHLIB= yes
LDCONFIG_DIRS=  %%PREFIX%%/lib/foo %%PREFIX%%/lib/bar
```

pkg-plist の他の部分と同様に、LDCONFIG_DIRS の内容も sed(1) による処理が行なわれるため、ここでも PLIST_SUB に指定した置換が行なわれることに注意してください。PREFIX には %%PREFIX%% を、LOCALBASE には %%LOCALBASE%%, X11BASE には %%X11BASE%% を使用することを推奨します。

6.2. 配布制限がある ports

ライセンスにはさまざまなものがあり、なかには、
アプリケーションをパッケージ化するやり方、営利目的で販売できるか、
といったことに制限をかけているものがあります。

port 作成者として、あなたには、使用許諾条件をよく読み、FTP/HTTP または CD-ROM
で再配布してはいけないソースコードやコンパイルされたバイナリを配布してしまい、
その責任が FreeBSD
プロジェクトにかかってくることのないよう注意する義務があります。
疑わしい場合には FreeBSD ports メーリングリスト で聞いてみてください。

そのような場合、次の節で説明されている変数が設定できます。

RESTRICTED は、それだけを設定すれば済むように、暗黙のうちに NO_CDROM と NO_PACKAGE を設定します。それに対して、NO_PACKAGE と NO_CDROM は独立で、同時に設定可能です。

6.2.1. NO_PACKAGE

この変数が設定されていたら、
このアプリケーションのバイナリパッケージを作成してはいけないということです。 ただし、この port の DISTFILES は自由に配布できます。

また、NO_PACKAGE は、バイナリパッケージが汎用的ではなく、いつもアプリケーションをソースコードからコンパイルすべき場合にも利用すべきです。
たとえば、アプリケーションにサイト特有の設定情報がコンパイル時にハードコードされるような場合には、 NO_PACKAGE を設定してください。

NO_PACKAGE には、 パッケージを作成すべきではない理由を述べた文字列を設定すべきです。

6.2.2. NO_CDROM

この変数は、バイナリパッケージの作成は許可されていますが、 その package や port の DISTFILES を販売用の CD-ROM (や DVD-ROM) に載せるのは許されていないことを表します。なんにせよ、バイナリパッケージと port の DISTFILES は、FTP/HTTP で入手できます。

NO_CDROM には、その port が何故 CD-ROM で再配布できないか説明する文字列を設定するべきです。
これはたとえば、その port のライセンスが"非商用" 利用に限っている場合に使うべきです。

6.2.3. RESTRICTED

アプリケーションのライセンスが、FTP/HTTP で、そのアプリケーションの DISTFILES をミラーすることや、 バイナリパッケージを配布することを禁じていたら、この変数を設定してください。

RESTRICTED には、その port が何故再配布できないか説明する文字列を設定するべきです。
典型的な場合としては、その port がプロプライエタリなソフトウェアを含んでいて、 DISTFILES を、おそらくソフトウェアに関する登録を行ったり、 EULA を承諾した後で、手動でダウンロードしなければならないことを表します。

6.2.4. RESTRICTED_FILES

RESTRICTED や NO_CDROM が設定されている時は、この変数はデフォルトで ${DISTFILES} ${PATCHFILES} になります。それ以外は空です。 一部の配布ファイルだけに制限がかかっていたら、この変数にそのファイルのリストを設定してください。

port committer は、設定した配布ファイル毎に /usr/ports/LEGAL に制限の内容を説明する項目を追加すべきことに注意してください。

6.3. perl の利用

表 2. perl を使用する *ports* 用の変数

変数	意味
USE_PERL5	その port のビルドと実行に perl 5 を使用することを示します。
USE_PERL5_BUILD	その port をビルドするのに perl 5 を使用することを示します。

変数	意味
USE_PERL5_RUN	その port を実行するのに perl 5 を使用することを示します。
PERL	システムまたは port からインストールされた perl 5 の完全なパスからバージョン番号を省いたもの。スクリプトの “#!” 行を置き換える必要があれば使ってください。
PERL_CONFIGURE	Perl の MakeMaker を使ってコンフィグレーションを行います。暗黙のうちに USE_PERL5 を設定します。

6.4. X11 の利用

表 3. *X* を利用する *ports* 用の変数

USE_X_PREFIX	その port は PREFIX ではなく X11BASE にインストールされます。
USE_XLIB	その port は X ライブラリを使用します。
USE_MOTIF	その port は Motif ツールキットを使用します。USE_XPM が自動的に設定されます。
USE_IMAKE	その port は imake を使用します。USE_X_PREFIX が自動的に設定されます。
XMKMF	xmkmf が PATH にない場合にパスを設定してください。デフォルトは xmkmf -a になります。

6.5. automake, autoconf および libtool の利用

表 4. *automake, autoconf* または *libtool* を使用する *ports* 用の変数

変数	意味
USE_AUTOMAKE	その port は automake を使用します。USE_AUTOCONF と USE_AUTOMAKE_VER?=14 が自動的に設定されます。
AUTOMAKE	automake が PATH に含まれない場合のフルパス。
USE_AUTOMAKE_VER	その port は automake を使用します。この変数の有効な値は 14 と 15 で、AUTOMAKE_DIR および ACLOCAL_DIR 変数が適切な値に設定されます。
AUTOMAKE_ARGS	USE_AUTOMAKE_VER が設定されていた場合に AUTOMAKE に渡す 1 つまたはそれ以上のコマンドライン引数
AUTOMAKE_ENV	AUTOMAKE を実行する前に設定する 1 つまたはそれ以上の環境変数 (とその値)

変数	意味
ACLOCAL	GNU aclocal が PATH にない場合にパスを設定してください。デフォルトは USE_AUTOMAKE_VER 変数に応じて設定されます。
ACLOCAL_DIR	GNU aclocal の共有ディレクトリのパスを設定してください。デフォルトは USE_AUTOMAKE_VER 変数に応じて設定されます。
AUTOMAKE_DIR	GNU automake の共有ディレクトリのパスを設定してください。デフォルトは USE_AUTOMAKE_VER 変数に応じて設定されます。
USE_AUTOCONF_VER	その port が autoconf を使用することを指定します。デフォルト値は 213 です。
USE_AUTOCONF	その port が autoconf を使用することを指定します。GNU_CONFIGURE および USE_AUTOCONF_VER?=213 を自動的に設定します。
AUTOCONF	GNU autoconf が PATH にない場合にパスを設定してください。デフォルトは USE_AUTOCONF_VER 変数の値に応じて設定されます。
AUTOCONF_ARGS	autoconf に渡すコマンドライン引数
AUTOCONF_ENV	この変数で指定された 変数=値 の組を autoconf を実行する前に環境変数として設定してください。
AUTOHEADER	GNU autoheader が PATH にない場合にパスを設定してください。デフォルトは USE_AUTOCONF_VER の値に応じて設定されます。
AUTORECONF	GNU autoreconf が PATH にない場合にパスを設定してください。デフォルトは USE_AUTOCONF_VER に応じて設定されます。
AUTOSCAN	GNU autoscan が PATH にない場合にパスを設定してください。デフォルトは USE_AUTOCONF_VER に応じて設定されます。
AUTOIFNAMES	GNU autoifnames が PATH にない場合にパスを設定してください。デフォルトは USE_AUTOCONF_VER に応じて設定されます。
USE_LIBTOOL	その port は libtool を使用します。GNU_CONFIGURE を自動的に設定します。
LIBTOOL	libtool が PATH にない場合にパスを設定してください。

変数	意味
LIBTOOLFILES	libtool 用のパッチファイル。 デフォルトは USE_AUTOCONF が設定されていれば aclocal.m4、それ以外は configure です。
LIBTOOLFLAGS	ltconfig に追加で渡すフラグ。 デフォルトは --disable-ltlibs。

6.6. GNOME の利用

FreeBSD/GNOME プロジェクトは、ある特定の port が使っている GNOME コンポーネントを特定するために 独自の変数群を使っています。 FreeBSD/GNOME プロジェクトのページに その変数のわかりやすい一覧 があります。

6.7. KDE の利用

表 5. *KDE* を利用する *ports* 用の変数

USE_QT_VER	その port は Qt ツールキットを使用します。 設定できる値は、 1, 2 および 3 で、それぞれ使用する Qt のメジャーバージョンを示します。 これは、MOC と QTCPPFLAGS をデフォルトの適切な値に設定します。
USE_KDELIBS_VER	その port は KDE ライブラリを使用します。 設定できる値は、 1, 2 および 3 で、それぞれ使用する KDE のメジャーバージョンを示します。 暗黙で USE_QT_VER に適切なバージョンを設定します。
USE_KDEBASE_VER	その port は KDE base を使用します。 設定できる値は、 1, 2 および 3 で、それぞれ使用する KDE のメジャーバージョンを示します。 暗黙で USE_KDELIBS_VER に適切なバージョンを設定します。
MOC	moc へのパスを設定してください。 デフォルトでは、USE_QT_VER の値に応じて設定されます。
QTCPPFLAGS	Qt のコードを処理する際の CPPFLAGS を設定してください。デフォルトでは USE_QT_VER の値に応じて設定されます。

6.8. Bison の利用

この節はまだ書かれていません。

6.9. Java の利用

あなたが作成している port の構築、実行、 または配布ファイルの展開に Java™ 開発キット (JDK) が必要なら、USE_JAVA を定義してください。

Ports Collection には、さまざまなベンダの JDK のいろいろなバージョンがあります。あなたが作成している port がその中のいずれかのバージョンを使わなければならないなら、どれを使うか指定できます。最新のバージョンは、 java/jdk14 です。

表 6. *Java* を利用する *ports* で定義すべき変数

変数	意味
USE_JAVA	この後の変数を有効にするには、 この変数を定義しなければなりません。
JAVA_VERSION	スペースで区切られた、適合する Java のバージョン一覧。""` を使ってバージョンの範囲を指定することもできます（使える値は、 `1.1[] 1.2[] 1.3[] 1.4[+]) です)。
JAVA_OS	スペースで区切られた、その port に適合する JDK port の OS 一覧。(使える値は、 native linux) です。
JAVA_VENDOR	スペースで区切られた、その port に適合する JDK port のベンダの一覧。(使える値は、 freebsd bsdjava sun ibm blackdown) です。
JAVA_BUILD	この変数が設定されていると、選択した JDK を、その port の構築依存性に追加します。
JAVA_RUN	この変数が設定されていると、選択した JDK を、その port の実行依存性に追加します。
JAVA_EXTRACT	この変数が設定されていると、選択した JDK を、その port の展開依存性に追加します。
USE_JIKES	その port の構築に jikes バイトコードコンパイラを使うべきかどうかを示します。 この変数に何の値も設定されていない時は、port は jikes が使えたら構築に利用します。 jikes の利用を明示的に禁止したり、 強制したりすることも可能です ('no' か 'yes' を設定してください)。 後者の場合は、devel/jikes が port の構築依存性に追加されます。

以下は、USE_JAVA を設定した port で行われる設定の一覧です。

表 7. *Java* を利用する *ports* で設定される変数

変数	値
JAVA_PORT	JDK port の名称 (例: 'java/jdk14')。

変数	値
JAVA_PORT_VERSION	JDK の完全なバージョン (例: '1.4.2')。最初の数字 2 つだけしか必要でなければ、${JAVA_PORT_VERSION:C/^([0-9])\.([0-9])(.*)$/\1.\2/} を使ってください。
JAVA_PORT_OS	JDK port が利用する OS (例: 'linux')。
JAVA_PORT_VENDOR	JDK port のベンダ (例: 'sun')。
JAVA_PORT_OS_DESCRIPTION	JDK port が利用する OS の説明 (例: 'Linux')。
JAVA_PORT_VENDOR_DESCRIPTION	JDK port のベンダの説明 (例: 'FreeBSD Foundation')。
JAVA_HOME	JDK がインストールされているディレクトリのパス (例: '/usr/local/jdk1.3.1')。
JAVAC	使用する Java コンパイラのパス (例: '/usr/local/jdk1.1.8/bin/javac' または '/usr/local/bin/jikes')。
JAR	使用する jar ツールのパス (例: '/usr/local/jdk1.2.2/bin/jar' または '/usr/local/bin/fastjar')。
APPLETVIEWER	appletviewer ユーティリティへのパス (例: '/usr/local/linux-jdk1.2.2/bin/appletviewer')。
JAVA	java 実行ファイルへのパス。 Java プログラムの実行にはこれを使ってください (例: '/usr/local/jdk1.3.1/bin/java')。
JAVADOC	javadoc ユーティリティプログラムへのパス。
JAVAH	javah プログラムへのパス。
JAVAP	javap プログラムへのパス。
JAVA_KEYTOOL	keytool ユーティリティプログラムへのパス。この変数は、JDK が Java 1.2 以上の場合のみ利用可能です。
JAVA_N2A	native2ascii ツールへのパス。
JAVA_POLICYTOOL	policytool プログラムへのパス。この変数は、JDK が Java 1.2 以上の場合のみ利用可能です。
JAVA_SERIALVER	serialver ユーティリティプログラムへのパス。
RMIC	RMI スタブ/スケルトンジェネレータ rmic へのパス。
RMIREGISTRY	RMI レジストリプログラム rmiregistry へのパス。
RMID	RMI デーモンプログラム rmid へのパス。この変数は、JDK が Java 1.2 以上の場合のみ利用可能です。

変数	値
JAVA_CLASSES	JDK クラスファイルが入っているアーカイブへのパス。 JDK 1.2 以降では、これは ${JAVA_HOME}/jre/lib/rt.jar です。 それより前の JDK は、 ${JAVA_HOME}/lib/classes.zip を使います。

Port のデバッグ情報を得るのに、java-debug make ターゲットが使えます。これは、前述の変数の多くについて値を表示します。

6.10. Python の利用

この節はまだ書かれていません。

6.11. Emacs の利用

この節はまだ書かれていません。

6.12. Ruby の利用

この節はまだ書かれていません。

6.13. SDL の利用

USE_SDL 変数は、devel/sdl12 や x11-toolkits/sdl_gui など、SDL ベースの ports への依存を自動設定するのに使われます。

今のところ、次の SDL ライブラリが認識されます。

- sdl: devel/sdl12
- gfx: graphics/sdl_gfx
- gui: x11-toolkits/sdl_gui
- image: graphics/sdl_image
- ldbad: devel/sdl_ldbad
- mixer: audio/sdl_mixer
- mm: devel/sdlmm
- net: net/sdl_net
- sound: audio/sdl_sound
- ttf: graphics/sdl_ttf

したがって、ある port が net/sdl_net と audio/sdl_mixer に依存していたら、構文は次のようになります。

```
USE_SDL=      net mixer
```

net/sdl_net と audio/sdl_mixer が必要とする devel/sdl12 の依存も自動的に追加されます。

USE_SDL を使うと、以下のことが自動的に行われます。

- BUILD_DEPENDS に sdl11-config への依存を追加します。
- CONFIGURE_ENV に SDL_CONFIG 変数を追加します。
- LIB_DEPENDS に選択したライブラリへの依存を追加します。

SDL ライブラリが利用できるかどうか調べるためには、 WANT_SDL 変数を使ってください。

```
WANT_SDL=yes

.include <bsd.port.pre.mk>

.if ${HAVE_SDL:Mmixer}!=""
USE_SDL+=   mixer
.endif

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

Chapter 7. MASTERDIR

その port の変数 (たとえば解像度とか紙のサイズなど) を変えたりした、少しだけ違うバージョンを作成する必要があるときには、 ユーザが分りやすいように package ごとに別々のサブディレクトリを作成し、 できるだけ port 間でファイルを共有するようにしてください。
ほとんどの場合、うまく変数を使えば、一つを除くすべてのディレクトリには とても短い Makefile を置くだけで済みます。 その短い Makefile では、 MASTERDIR を使って、残りのファイルがあるディレクトリを指定できます。 また、PKGNAMESUFFIX の一部に変数に使って、package が別々の名前を持つようにしてください。

具体的な例を示すのが一番わかりやすいでしょう。 これは japanese/xdvi300/Makefile の一部です。

```
PORTNAME=       xdvi
PORTVERSION=    17
PKGNAMEPREFIX=  ja-
PKGNAMESUFFIX=  ${RESOLUTION}
 :
# default
RESOLUTION?=   300
.if ${RESOLUTION} != 118 && ${RESOLUTION} != 240 && \
       ${RESOLUTION} != 300 && ${RESOLUTION} != 400
       @${ECHO} "Error: invalid value for RESOLUTION: \"${RESOLUTION}\""
       @${ECHO} "Possible values are: 118, 240, 300 (default) and 400."
       @${FALSE}
.endif
```

japanese/xdvi300 には Makefile の他に通常のパッチや、 package ファイル等が置かれています。 このディレクトリで make を実行すると、 デフォルトの解像度 (300) を使って、 普通に port のビルドを行ないます。

他の解像度に関していうと、 xdvi118/Makefile に 必要なのは_これだけ_です:

```
RESOLUTION=     118
MASTERDIR=      ${.CURDIR}/../xdvi300

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

(xdvi240/Makefile や xdvi400/Makefile も同様のものになります)。 bsd.port.mk は、 MASTERDIR の定義から FILESDIR や SCRIPTDIR 等の 通常のサブディレクトリが xdvi300 以下に存在することを理解します。 RESOLUTION=118 の行が、 xdvi300/Makefile の RESOLUTION=300 の行を上書きし、 port は解像度を 118 として作成されます。

Chapter 8. 共有ライブラリのバージョン

まず共有ライブラリのバージョンについての指針を読んで、
一般的に共有ライブラリのバージョンをどうすれば良いかを理解してください。
ソフトウェアの作者は自分がしていることを理解していると、
盲目的に信じていてはいけません。多くの場合は理解していないのです。
細部にわたって注意深く考慮することは大変重要です。なぜなら我々は、
互換性がないかもしれない大量のソフトウェアを共存させようとする特殊な状況にあるからです。
むかし、不注意な port の導入が共有ライブラリに関する重大な問題を引き起してしまったことがあります (なぜ jpeg-6b の共有ライブラリのバージョン番号が 9 なのか、今まで不思議に思ったことはありませんか?)。
もし疑問があれば、FreeBSD ports メーリングリスト にメールを送ってください。
ほとんどの時間は正しい共有ライブラリのバージョンを決めることと、
それを実現するためのパッチを作成することに終始します。

Chapter 9. マニュアルページ

MAN[1-9LN] 変数に指定したマニュアルは 自動的に pkg-plist に追加されます (つまり、マニュアルを pkg-plist に加えてはいけません- pkg-plist の生成を参照してください)。 また、/etc/make.conf 中の NOMANCOMPRESS の設定に従って、インストール時に マニュアルを自動的に圧縮したり復元したりします。

その port が、シンボリックリンクやハードリンクを用いて、複数のファイル名を持つマニュアルをインストールする場合には、 それらを識別するために MLINKS 変数を使用しなければなりません。 port によってインストールされたリンクは、意図したファイルをきちんと指しているかどうか確認するため、 bsd.port.mk によって削除されたり、再作成されたりします。 MLINKS に指定されたマニュアルも、 pkg-plist に含めてはいけません。

マニュアルをインストール時に圧縮するかどうかを指定するには、 MANCOMPRESSED 変数を使用します。この変数は yes, no そして maybe の三つの値をとることができます、 yes はマニュアルが既に圧縮されてインストールされていること、 no は圧縮されていないこと、 maybe は既にそのソフトウェアが NOMANCOMPRESS の値に従っていて、 bsd.port.mk は特別なにもする必要がないことを意味します。

USE_IMAKE がセットされていて、 NO_INSTALL_MANPAGES がセットされていなければ、 MANCOMPRESSED は自動的に yes に設定されます。それ以外の場合には、MANCOMPRESSED は no に設定されます。 その port にとって、デフォルトの設定が適切でない場合以外には、 明示的に設定する必要はありません。

PREFIX 以外のディレクトリの下にマニュアルを置くような port では、そのディレクトリを MANPREFIX で指定することができます。 さらに、いくつかの perl モジュールの ports のように、特定のセクションのマニュアルだけを非標準の場所にインストールする場合、 個々のマニュアルのパスを MAN_sect_PREFIX (ここで *sect* は 1-9, L, または N のいずれか) により指定することができます。

マニュアルが言語特有のサブディレクトリに置かれる場合には、 その言語名を MANLANG に設定してください。 この変数のデフォルト値は "" になっています (つまり、英語のみ)。

これは、全部をまとめた例です。

```
MAN1=           foo.1
MAN3=           bar.3
MAN4=           baz.4
MLINKS=         foo.1 alt-name.8
MANLANG=        "" ja
MAN3PREFIX=     ${PREFIX}/shared/foobar
MANCOMPRESSED=  yes
```

これは、この port により以下の 6 個のファイルがインストールされることを表しています。

```
${PREFIX}/man/man1/foo.1.gz
${PREFIX}/man/ja/man1/foo.1.gz
${PREFIX}/shared/foobar/man/man3/bar.3.gz
${PREFIX}/shared/foobar/man/ja/man3/bar.3.gz
${PREFIX}/man/man4/baz.4.gz
${PREFIX}/man/ja/man4/baz.4.gz
```

さらに ${PREFIX}/man/man8/alt-name.8.gz がこの port によってインストールされるかどうかわかりませんが、 それとは無関係に foo(1) と alt-name(8) のマニュアルページを指すシンボリックリンクが作成されます。

Chapter 10. Motif を必要とする port

コンパイルに Motif ライブラリを必要とするアプリケーションがいくつかあります (Motif 自体は有料のものがいくつかの会社から手に入りますし、x11-toolkits/lesstif には多くのアプリケーションを動作させることが可能な無料の互換ライブラリもあります)。 Motif は広く使われているツールキットですし、
有料のもののライセンスでもライブラリを静的にリンクした実行形式の再配布が認められている場合が多いので、 Motif を必要とするソフトウェアを簡単に (port からコンパイルする人々のために) 動的にでも、 (package を配布する人々のために)
静的にでもリンクできるような仕組みが用意されています。

10.1. USE_MOTIF

Motif が無いとコンパイルできない port の Makefile では、この変数を指定してください。これにより、Motif を持っていない人がこの port をコンパイルしようとするのを未然に防ぎます。

10.2. MOTIFLIB

この変数は bsd.port.mk によって適当な Motif ライブラリの指定に置き換えられます。Port のソースの Makefile や Imakefile で Motif ライブラリを参照しているところを、この変数を参照するようにパッチを適用してください。

代表的な例としては以下の二つがあげられます:

1. Makefile か Imakefile の中で Motif ライブラリが `-lXm` として使われている場合には、 かわりに `MOTIFLIB` と書いてください。
2. Imakefile の中で `XmClientLibs` が使われている場合には、それを `${MOTIFLIB} ${XTOOLLIB} ${XLIB}` と書きかえてください。

なお `MOTIFLIB` は通常、 `-L/usr/X11R6/lib -lXm` か `/usr/X11R6/lib/libXm.a` に置き換えられます。したがって前に `-L` や `-l` をつける必要はありません。

Chapter 11. X11 のフォント

もし、あなたの port が X Window System のフォントをインストールするのであれば、 それらを X11BASE/lib/X11/fonts/local に置くようにしてください。このディレクトリは XFree86 3.3.3 で新設されたものです。 このディレクトリが存在しなければ作成して、ユーザに XFree86 を 3.3.3 かそれより新しいものに更新するか、 少なくともこのディレクトリを /etc/XF86Config のフォントパスに加えるように促すメッセージを出力するようにしてください。

Chapter 12. Info ファイル

あなたが作成している package で GNU info ファイルをインストールする必要があれば、それを INFO 変数に (後ろの .info なしで) 書いてください。package 登録の前に一時的に生成された pkg-plist に、適切なインストールおよびアンインストールコードが自動的に追加されます。

Chapter 13. pkg-* ファイル

pkg-* ファイルには、 まだ取り上げていない何かと重宝なトリックがいくつかあります。

13.1. pkg-message

もしインストールする人にメッセージを表示する必要がある場合には、 そのメッセージを pkg-message に置くことができます。 この機能は pkg_add(1) の後の追加のインストール手続きを表示するときなどに重宝します。

pkg-message ファイルは pkg-plist に加える必要はありません。 また、もしユーザが package ではなく port を使用している場合には自動的には表示されませんので、明示的に `post-install` で表示するようにするべきでしょう。

13.2. pkg-install

バイナリパッケージが pkg_add(1) でインストールされるときに実行する必要のあるコマンドがあれば、pkg-install スクリプトを使って実行することができます。 このスクリプトは自動的に package に加えられ、 pkg_add(1) によって 2 回実行されます。 1 回目は `${SH} pkg-install ${PKGNAME} PRE-INSTALL` として、2 回目には `${SH} pkg-install ${PKGNAME} POST-INSTALL` として実行されます。どちらのモードで実行されているかは `$2` を調べることによってわかります。 環境変数 `PKG_PREFIX` には package がインストールされるディレクトリが設定されます。 詳細は pkg_add(1) を見てください。

port を `make install` でインストールするときにはこのスクリプトは自動的に実行されません。
もし実行される必要があるならば port の Makefile から明示的に呼ぶ必要があります。

13.3. pkg-deinstall

このスクリプトは package が削除される際に実行されます。

このスクリプトは、pkg_delete(1) から 2 回実行されます。1 回目は `${SH} pkg-install ${PKGNAME} DEINSTALL` と、2 回目は `${SH} pkg-install ${PKGNAME} POST-DEINSTALL` という形で実行されます。

13.4. pkg-req

(訳注: 実行されるマシンの状態に応じて) その port をインストールするべきか、そうでないかを判断する必要があるときには、 "要件 (requirements)" スクリプト pkg-req を作ることができます。 インストールや削除を実行すべきかどうか判断するために、このスクリプトがインストールや削除を実行する際に自動的に実行されます。

このスクリプトはインストール時には pkg_add(1) により `pkg-req ${PKGNAME} INSTALL` として実行され、削除時には pkg_delete(1) により `pkg-req ${PKGNAME} DEINSTALL` として 実行されます。

13.5. make の変数にあわせた pkg-plist の変更

いくつかの port、特に p5- ports などは、configure のオプション (あるいは、p5- の場合は perl のバージョン) によって pkg-plist を変える必要があります。 これを容易に実現するために pkg-plist 中の %%OSREL%%, %%PERL_VER%%, %%PERL_VERSION%% は適切に置き換えられるようになっています。 %%OSREL%% の値はオペレーティングシステムの数字で表されたリビジョンです (たとえば 4.9)。 %%PERL_VERSION%% は perl のバージョン番号全体 (たとえば 5.00502) で、%%PERL_VER%% は perl のバージョン番号からパッチレベルを引いたものです (たとえば 5.005)。

他の置き換えが必要であれば、PLIST_SUB 変数に VAR=VALUE という形式のペアのリストを設定することによって、 pkg-plist 中の %%VAR%% は *VALUE* に置き換えられます。 たとえばバージョンに固有のたくさんのファイルをインストールする場合には、Makefile に

```
OCTAVE_VERSION= 2.0.13
PLIST_SUB=      OCTAVE_VERSION=${OCTAVE_VERSION}
```

と書いて、PLIST 中のバージョン番号が表われるすべてのところに、 %%OCTAVE_VERSION%% と書きます。 このようにしておけば、port をアップグレードするときに、 何十行 (時として、何百行) も pkg-plist を書き替えないですみます。

この書き換えは (マニュアルの追加も) pre-install と do-install ターゲットの間に pkg-plist を読み TMPPLIST (デフォルトは WRKDIR/.PLIST.mktmp) に書き込むことによって行なわれます。 もし、あなたの port が PLIST を実行時に生成するのであれば、 pre-install の間かその前に行なうようにしてください。
また、書きかえられたあとのファイルを編集する必要がある場合には、 post-install で TMPPLIST を書きかえてください。

13.6. pkg-* ファイルの名前変更

pkg-* ファイルの名前はすべて変数を使用して定義されていますので、 必要であれば Makefile 中で変更可能です。 いくつかの ports で一つの pkg-* ファイルを共有する場合や、上記のファイルに書き込みをしなければならないときなど特に便利です (pkg-* サブディレクトリに直接書き込むのが良くない理由については WRKDIR 以外への書きこみ を参照してください)。

以下に変数名と そのデフォルト値のリストを示します。 (PKGDIR のデフォルト値は ${MASTERDIR} になっています。)

変数名	デフォルト値
DESCR	${PKGDIR}/pkg-descr
PLIST	${PKGDIR}/pkg-plist
PKGINSTALL	${PKGDIR}/pkg-install
PKGDEINSTALL	${PKGDIR}/pkg-deinstall
PKGREQ	${PKGDIR}/pkg-req
PKGMESSAGE	${PKGDIR}/pkg-message

PKG_ARGS を上書きせずにこれらの変数を変更するようにしてください。 PKG_ARGS を変更すると、これらのファイルは port から正しく /var/db/pkg にインストールされなくなります。

Chapter 14. port のテスト

14.1. portlint

送付や commit をする前に `portlint` を使ってチェックしましょう。

14.2. PREFIX

なるべく port は PREFIX に対する相対パスにインストールすることができるように心がけてください (この変数の値は USE_X_PREFIX か USE_IMAKE が指定してある時には X11BASE (デフォルトは /usr/X11R6)、そうでない場合にはLOCALBASE (デフォルトは /usr/local) にセットされます)。

サイトによってフリーソフトウェアがインストールされる場所が違いますので、 ソース内で /usr/local や /usr/X11R6 を明示的に書かないようにしてください。 X のプログラムで imake を使うものについては、これは問題にはなりません。 それ以外の場合には port の scripts/Makefile で /usr/local (imake を使わない X のプログラムは /usr/X11R6) と書いてあるところを、 ${PREFIX} に書き換えてください。この値は port のコンパイルおよび、インストールの全段階において、自動的に下位のプロセスに渡されます。

そのアプリケーションが PREFIX を使用しないで、何かを直接 /usr/local にインストールしないことを確認してください。
以下のようにすると、簡単なテストを行なうことができます:

```
# make clean; make package PREFIX=/var/tmp/port-name
```

この時、もし PREFIX の外に 何かがインストールされていた場合、package 生成プロセスはファイルが見つからないと文句を言うはずです。

ただし、これは そのソフトウェアが内部で決め打ちの参照を していないかどうか だとか、他の port によってインストールされる ファイルを参照する際に LOCALBASE を正しく使用しているかどうかをテストしているわけではありません。 その port を他の場所にインストールした状態で、 /var/tmp/port-name に対するインストールを試みることにより、 そのテストをすることができるでしょう。

USE_X_PREFIX は本当に必要な時 (つまり X のライブラリをリンクしたり、X11BASE 以下にあるファイルを参照したりする必要がある時) 以外には 設定しないでください。

変数 PREFIX の値は port の Makefile やユーザの環境で変更することもできます。 しかし、個々の port が Makefile でこの変数の値を明示的に設定することはなるべくしないでください。

また、他の port によりインストールされるプログラムやファイルを指定する場合には、直接的なパス名を使用するのではなく
上で述べた変数を使用してください。 たとえば less のフルパスを PAGER というマクロに入れたい場合は、 -DPAGER=\"/usr/local/bin/less\"
というフラグをコンパイラに渡すかわりに

```
-DPAGER=\"${PREFIX}/bin/less\"
```

(X Window System を使う port の場合には

```
-DPAGER=\"${LOCALBASE}/bin/less\"
```

) を渡してください。 こうしておけば、システム管理者が /usr/local を まるごと どこか他の場所に移していたとしても、その port が そのまま使える可能性が高くなります。

14.3. FreshPorts 正当性テスト

http://www.FreshPorts.org/ には、FreeBSD ports へ commit されたものについて、自動的に正当性テストを行う仕組みがあります。 このサービスに登録すると、あなたが commit したものについて、 正当性テストでエラーが起きると連絡が行きます。

このサービスを利用したい場合、必要なのは FreshPorts のアカウントだけです。登録したメールアドレスが `@FreeBSD.org` のものであれば、ウェブページの右側にサービスを選択するリンクがあるはずです。 FreshPorts にアカウントを持っていても `@FreeBSD.org` のメールアドレスを利用していない場合、メールアドレスを `@FreeBSD.org` に変え、登録したあとで、 メールアドレスをまた変更してください。

Chapter 15. アップグレード

port のバージョンが原作者からのものに比べて古いことに気がついたら、 まずはあなたの持っている port が私たちの最新のもの (最新の port は FreeBSD FTP ミラーサイトの ports/ports-current というディレクトリにあります) であることを確認してください。 また、Ports Collection 全体を最新の状態に保つために CVSup を利用することもできます。 詳しくは FreeBSD ハンドブックをご覧ください。

次に port の Makefile に MAINTAINER (保守担当者) のアドレスが書いてある場合には、その人にメールを出してみましょう。
保守担当者の人がすでにアップグレードの準備をしているかも知れませんし、
(新しいバージョンの安定度に問題があるなど)
あえてアップグレードをしない理由があるのかも知れません。
その人たちと作業を重複させたくはないでしょう。 なお、保守されていない ports は、保守担当者が ports@FreeBSD.org になっています。その場合は、
そのアドレスにメールを送っても役に立たないでしょう。

保守担当者にアップグレードをしてくれと頼まれた場合、 あるいは、保守担当者が ports@FreeBSD.org の場合は、あなたがアップグレードをしてくださると助かります。
その場合にはアップグレードを作成した後、 変更前と変更後のディレクトリの再帰的 diff の出力結果を保存してください (たとえば変更前のディレクトリが superedit.bak という名前でとってあり、変更後のものが superedit に入っているなら、 diff -ruN superedit.bak superedit の結果を保存してください)。 unified または context diff のどちらでも構いませんが port のコミッターは一般に unified diff の方を好むようです。 -N オプションの使い方を覚えておいてください。 これは、新たにファイルが追加されたり、古いファイルが削除された場合を diff が正しく扱えるようにする方法です。わたしたちに差分を送る前に、
すべての変更が正しくなされているか出力結果を確認してください。

差分を送る一番良い方法は send-pr(1) (カテゴリは ports) に diff の出力結果を添えて、わたしたちに送ってもらうのが一番良いです。 commit する際に CVS に明確に記述しなければならないので、 あなたがその port のメンテナになるなら、概要 (synopsis) 行の先頭に [maintainer update] と記入し、PR の "Class" を maintainer-update にしてください。 付け加えたり削除したりしたファイルがあればそれについて書いておいてください。 もし diff の大きさが 20 KB 程度を超えるようであれば、 圧縮したものを uuencode してください。 そうでなければそのまま PR に入れるだけで構いません。

最後に、役に立つ障害報告の書き方について詳しいことは、 障害報告についての記事の障害報告の書き方の節を読んでください。

更新の動機が、セキュリティ上の問題や、 現在 commit されている port に重大な欠陥である場合は、 ports 管理チーム <portmgr@FreeBSD.org> に連絡して、あなたの port のパッケージをただちに作りなおして再配布するように要求してください。
そうしないと、無防備な pkg_add(1) のユーザたちが、何週間にもわたって pkg_add -r で古いバージョンをインストールし続けてしまいます。

繰り返しになりますが、既存の ports の変更を送るときには shar(1) ではなく diff(1) を使用してください!

Chapter 16. やっていいことといけないこと

16.1. はじめに

このセクションではソフトウェアを port する上で、良くある落し穴などについて説明します。このリストを使ってあなた自身が作成した port のチェックはもとより、さらに障害報告データベースにある他の人が提出した port のチェックもできます。PR データベースにある、他の人が作成した port のチェックもできます。あなたがチェックした port についてのコメントをバグ報告と一般的な論評にしたがって送ってください。PR データベースにある port をチェックすることは、わたしたちがそれらを commit するのを早めるとともに、何をしているかをあなたが理解していることも証明します。

16.2. バイナリの strip

バイナリは特に必要がなければ、手動で strip しないでください。すべてのバイナリは strip すべきですが、INSTALL_PROGRAM マクロがバイナリのインストールと strip を同時に行います(次節をご覧ください)。

ファイルを strip する必要はあるものの INSTALL_PROGRAM マクロを使いたくない場合は、${STRIP_CMD} でプログラムを strip できます。これは、多くの場合 post-install ターゲット内で行われます。たとえば

```
post-install:
    ${STRIP_CMD} ${PREFIX}/bin/xdl
```

インストールされた実行形式がすでに strip されているかどうかは file コマンドで確認できます。"not stripped" と表示されなければ strip されていることを示しています。さらに、strip(1) はすでに strip されたプログラムは strip せず、問題なく終了します。

16.3. INSTALL_* マクロ

あなた自身の *-install ターゲットでファイルの正しいモードとオーナを保証するために、必ず bsd.port.mk で提供されているマクロを使用してください。

- ${INSTALL_PROGRAM} は実行可能なバイナリをインストール(し、その過程で strip 処理)するコマンドです。
- ${INSTALL_SCRIPT} は実行可能なスクリプトをインストールするコマンドです。
- ${INSTALL_DATA} は共有可能なデータをインストールするコマンドです。
- ${INSTALL_MAN} はマニュアルとその他の文書をインストールするコマンドです (圧縮はしません)。

これらは基本的に install コマンドに適切なフラグを与えたものです。それらは distfile の Makefile で、頭に "BSD_" が付けられた (つまり BSD_INSTALL_PROGRM というような) 形で使うことができます。どのようにこれらを使用するかは以下の例を見てください。

16.4. WRKDIR

WRKDIR の外に存在するファイルには何も書き込んではいけません。port のビルド中に書き込み可能なことが保証されているのは WRKDIR の中だけです (書き込み不可のツリー上での port ビルドの例については、CDROM からの ports のインストール を参照のこと)。pkg-* ファイルを変更する必要があるときには、ファイルを上書きするのではなく 変数の再定義により 行なうようにしてください。

16.5. WRKDIRPREFIX

WRKDIRPREFIX を尊重していることを確認してください。特に、別の port の WRKDIR を参照しているときには気を付けてください。正しい場所は、WRKDIRPREFIXPORTSDIR/subdir/name/work です。PORTSDIR/subdir/name/work や .CURDIR/../../subdir/name/work ではありません。

また、自分で WRKDIR 定義するときには先頭に ${WRKDIRPREFIX}${.CURDIR} が付いていることを確認してください。

16.6. OS の種類やバージョンの識別

どのバージョンの Unix で動かすかによって、変更や条件つきコンパイルが必要なコードに出くわすこともあるでしょう。
そのような変更を行なう場合には、古い FreeBSD システムへのバックポートや、CSRG の 4.4BSD, BSD/386, 386BSD, NetBSD, OpenBSD 等、他の BSD システムへの移植が可能なように、できるだけ汎用的な変更を行なうことを心がけてください。

4.3BSD/Reno (1990) と、それより新しいバージョンの BSD コードを区別するには、<sys/param.h> で定義されている BSD マクロを利用するのがよいでしょう。
このファイルがすでにインクルードされていれば良いのですが、そうでない場合には、その .c ファイルの 適当な場所に以下のコードを追加してください。

```
#if (defined(__unix__) || defined(unix)) && !defined(USG)
#include <sys/param.h>
#endif
```

これらの二つのシンボルが定義されているシステムには必ず sys/param.h があるはずです。もしそうでないシステムを発見したら、FreeBSD ports メーリングリスト までメールを送ってわたしたちに伝えてください。

あるいは、GNU Autoconf のスタイルを使用することもできます。

```
#ifdef HAVE_SYS_PARAM_H
#include <sys/param.h>
#endif
```

この方法を使用するときには、Makefile 中の CFLAGSに -DHAVE_SYS_PARAM_H を加えることを忘れないようにしてください。

いったん sys/param.h がインクルードされると、

```
#if (defined(BSD) && (BSD >= 199103))
```

このようにしてそのコードが 4.3 Net2 コードベース、 またはそれより新しいもの (例: FreeBSD 1.x, 4.3/Reno, NetBSD 0.9, 386BSD, BSD/386 1.1 とそれ以前) の上でコンパイルされているかを検出できます。

```
#if (defined(BSD) && (BSD >= 199306))
```

これは、4.4コードベース、またはそれより新しいもの (例: FreeBSD 2.x, 4.4, NetBSD 1.0, BSD/386 2.0 とそれ以後) の上でコンパイルされているかどうかを検出するために使用します。

4.4BSD-Lite2 コードベースでは BSD マクロの値は 199506 になっています。 これは参考程度の意味合いしかありません。 4.4-Lite ベースの FreeBSD と 4.4-Lite2 での変更がマージされたバージョンとを区別するのに使用するべきものではありません。
この目的のためにはかわりに {freebsd} マクロを使用してください。

以下は控え目に使ってください。

- {freebsd} はFreeBSDのすべての版で定義されています。 変更が FreeBSD だけに適用されるとき以外は使用しないでください。 port でよくある strerror() ではなく sys_errlist[] を使うなどは FreeBSDでの変更ではなく BSD の流儀です。
- FreeBSD 2.xでは {freebsd} が 2 と定義されています。 それ以前の版では 1 になっています。 その後の版でもメジャー番号に合うように上げています。
- もし FreeBSD 1.x システムと FreeBSD 2.x 以降のシステムを区別する必要があれば、 上で述べた BSD マクロを使用するのが大抵の場合において正しい答です。 もし FreeBSD 特有の変更であれば (ld を使うときの共有ライブラリ用のオプションなど)、 {freebsd}を使い #if {freebsd} > 1 のようにFreeBSD 2.x および、 それ以降のシステムを検出するのはかまいません。 もし 2.0-RELEASE 以降の FreeBSD システムを細かく検出したければ、 以下を使用することができます。

  ```
  #if __FreeBSD__ >= 2
  #include <osreldate.h>
  #    if __FreeBSD_version >= 199504
           /* 2.0.5+ release specific code here */
  #    endif
  #endif
  ```

これまで、何百もの port が作られてきましたが、 {freebsd} が正しく使われたのは一つか二つの場合だけでしょう。 以前の port が間違ってふさわしくない場所でそのマクロを使っているからといって、 それをまねする理由はありません。

16.7. __FreeBSD_version の値

以下は、sys/param.h で定義されている __FreeBSD_version の便利な一覧です。

表 8. *__FreeBSD_version values*

Release	__FreeBSD_version
2.0-RELEASE	119411
2.1-CURRENT	199501, 199503
2.0.5-RELEASE	199504
2.1 以前の 2.2-CURRENT	199508
2.1.0-RELEASE	199511
2.1.5 以前の 2.2-CURRENT	199512
2.1.5-RELEASE	199607
2.1.6 以前の 2.2-CURRENT	199608
2.1.6-RELEASE	199612
2.1.7-RELEASE	199612
2.2-RELEASE	220000
2.2.1-RELEASE	220000 (変更なし)
2.2.1-RELEASE 以降の 2.2-STABLE	220000 (変更なし)
texinfo-3.9 以降の 2.2-STABLE	221001
top 導入以降の 2.2-STABLE	221002
2.2.2-RELEASE	222000
2.2.2-RELEASE 以降の 2.2-STABLE	222001
2.2.5-RELEASE	225000
2.2.5-RELEASE 以降の 2.2-STABLE	225001
ldconfig -R マージ以降の 2.2-STABLE	225002
2.2.6-RELEASE	226000
2.2.7-RELEASE	227000
2.2.7-RELEASE 以降の 2.2-STABLE	227001
semctl(2) 変更以降の 2.2-STABLE	227002
2.2.8-RELEASE	228000
2.2.8-RELEASE 以降の 2.2-STABLE	228001
mount(2) 変更以前の 3.0-CURRENT	300000
mount(2) 変更以降の 3.0-CURRENT	300001
semctl(2) 変更以降の 3.0-CURRENT	300002
ioctl 引数変更以降の 3.0-CURRENT	300003
ELF 化以降の 3.0-CURRENT	300004
3.0-RELEASE	300005

Release	__FreeBSD_version
3.0-RELEASE 以降の 3.0-CURRENT	300006
3/4 の分岐以降の 3.0-STABLE	300007
3.1-RELEASE	310000
3.1-RELEASE 以降の 3.1-STABLE	310001
C++ コンストラクタ/デストラクタ順序変更の後の 3.1-STABLE	310002
3.2-RELEASE	320000
3.2-STABLE	320001
バイナリ互換性のない IPFW とソケットの変更後の 3.2-STABLE	320002
3.3-RELEASE	330000
3.3-STABLE	330001
libc に mkstemp(3) が追加された後の 3.3-STABLE	330002
3.4-RELEASE	340000
3.4-STABLE	340001
3.5-RELEASE	350000
3.5-STABLE	350001
3.4 が分岐した後の 4.0-CURRENT	400000
dynamic linker の変更後の 4.0-CURRENT	400001
C++ コンストラクタ/デストラクタ順序変更の後の 4.0-CURRENT	400002
dladdr(3) 機能追加後の 4.0-CURRENT	400003
__deregister_frame_info dynamic linker のバグ修正、 EGCS 1.1.2 導入後の 4.0-CURRENT	400004
suser(9) の API 変更、newbus 化 以降の 4.0-CURRENT	400005
cdevsw 登録方法の変更後の 4.0-CURRENT	400006
ソケットレベルの証明書 (credential) のために so_cred が追加された後の 4.0-CURRENT	400007
libc_r への poll syscall ラッパー追加後の 4.0-CURRENT	400008
kernel の `dev_t` 型から `struct spacinfo` ポインタへの 変更後の 4.0-CURRENT	400009
jail(2) のセキュリティホール 修正後の 4.0-CURRENT	400010
`sigset_t` の データ型変更後の 4.0-CURRENT	400011

Release	__FreeBSD_version
システムコンパイラを gcc 2.95.2 にアップグレードした 後の 4.0-CURRENT	400012
動的組み込み可能な Linux モードの ioctl ハンドラが 追加された後の 4.0-CURRENT	400013
OpenSSL 導入後の 4.0-CURRENT	400014
GCC 2.95.2 の C++ ABI 変更で、 デフォルトを -fvtable-thunks から -fno-vtable-thunks に 変更した後の 4.0-CURRENT	400015
OpenSSH 導入後の 4.0-CURRENT	400016
4.0-RELEASE	400017
4.0-RELEASE 以降の 4.0-STABLE	400018
チェックサム計算タイミングの変更後の 4.0-STABLE	400019
libxpg4 が libc にマージされた後の 4.0-STABLE	400020
Binutils を 2.10.0 にアップグレードし、 ELF バイナリのマーク付け (branding) 方法を変更し、 tcsh をベースシステムに導入した後の 4.0-STABLE	400021
4.1-RELEASE	410000
4.1-RELEASE 以降の 4.1-STABLE	410001
setproctitle(3) が libutil から libc に 移動した後の 4.1-STABLE	410002
4.1.1-RELEASE	411000
4.1.1-RELEASE 以降の 4.1.1-STABLE	411001
4.2-RELEASE	420000
libgcc.a と libgcc_r.a の結合および、関連する GCC linkage 変更が行なわれた後の 4.2-STABLE	420001
4.3-RELEASE	430000
wint_t 導入後の 4.3-STABLE	430001
PCI パワーステート API マージ後の 4.3-STABLE	430002
4.4-RELEASE	440000
d_thread_t 導入後の 4.4-STABLE	440001
マウント構造変更 (ファイルシステム kld に影響あり) 後の 4.4-STABLE	440002
smbfs のユーザランド部が取り込まれた後の 4.4-STABLE	440003
4.5-RELEASE	450000

Release	__FreeBSD_version
usb の構成要素の名称が変更された後の 4.5-STABLE	450001
rc.conf(5) の sendmail_enable 変数が NONE という値をとれるようになった後の 4.5-STABLE	450004
package 作成のデフォルトを XFree86 4 に移行した後の 4.5-STABLE	450005
accept filter が修正され、簡単なサービス妨害攻撃には影響を受けなくなった後の 4.5-STABLE	450006
4.6-RELEASE	460000
sendfile(2) をドキュメントに適合するよう修正して、送信されたいかなるヘッダも、ファイルから送信されたデータの総量に合計しないようにした 4.6-STABLE	460001
4.6.2-RELEASE	460002
4.6-STABLE	460100
sed -i を MFC した後の 4.6-STABLE	460101
多くの新たな pkg_install の機能を HEAD から MFC した後の 4.6-STABLE	460102
4.7-RELEASE	470000
4.7-STABLE	470100
sF の代わりに std{in,out,err}p 参照生成を開始。これは、std{in,out,err} をコンパイル時の定数から、ランタイムに変更します。	470101
m_aux mbuf を m_tag で置き換える mbuf の変更を MFC した後の 4.7-STABLE	470102
OpenSSL 0.9.7 導入後の 4.7-STABLE	470103
4.8-RELEASE	480000
4.8-STABLE	480100
realpath(3) がスレッドセーフになった後の 4.8-STABLE	480101
4.8-STABLE における twe の 3ware API の変更	480102
4.9-RELEASE	490000
4.9-STABLE	490100
構造体 kinfo_eproc に e_sid が追加された後の 4.9-STABLE	490101

Release	__FreeBSD_version
rtld に libmap 機能を MFC した後の 4.9-STABLE	490102
4.10-RELEASE	491000
5.0-CURRENT	500000
ELF ヘッダフィールドの追加と ELF バイナリのマーク付け (branding) 方法の変更後の 5.0-CURRENT	500001
kld メタデータ変更後の 5.0-CURRENT	500002
buf/bio 変更後の 5.0-CURRENT	500003
binutils アップグレード後の 5.0-CURRENT	500004
libxpg4 コードの libc へのマージと、 TASKQ インターフェイスの導入後の 5.0-CURRENT	500005
AGP インターフェイス追加後の 5.0-CURRENT	500006
Perl を 5.6.0 にアップグレードした後の 5.0-CURRENT	500007
KAME コードを 2000/07 版のソースに更新した後の 5.0-CURRENT	500008
ether_ifattach() および ether_ifdetach() 変更後の 5.0-CURRENT	500009
mtree のデフォルトをオリジナルの変種に戻し、シンボリックリンクをたどる -L オプションを追加した後の 5.0-CURRENT	500010
kqueue API 変更後の 5.0-CURRENT	500011
setproctitle(3) が libutil から libc へ移動した後の 5.0-CURRENT	500012
最初の SMPng がコミットされた後の 5.0-CURRENT	500013
<sys/select.h> が <sys/selinfo.h> に 移動した後の 5.0-CURRENT	500014
libgcc.a と libgcc_r.a の結合および関連する GCC linkage 変更が行なわれた後の 5.0-CURRENT	500015
libc と libc_r の混合リンクを許し、 -pthread オプションを deprecate する 変更後の 5.0-CURRENT	500016
mountd 等が使用する kernel-exported API の安定化のため、ucred 構造体から xucred 構造体へ移行した後の 5.0-CURRENT	500017
CPU 依存の最適化を制御するための make 変数 CPUTYPE が追加された後の 5.0-CURRENT	500018

Release	__FreeBSD_version
<machine/ioctl_fd.h> が <sys/fdcio.h> に移動した後の 5.0-CURRENT	500019
ロケール名変更の後の 5.0-CURRENT	500020
Bzip2 導入後の 5.0-CURRENT。 また、S/Key が削除されていることも示す。	500021
SSE サポート後の 5.0-CURRENT	500022
KSE マイルストーン 2 以降の 5.0-CURRENT	500023
d_thread_t 導入、および UUCP を ports に移動した後の 5.0-CURRENT	500024
64 ビットプラットホーム上のデスクリプタおよび cred 受け渡し ABI 変更後の 5.0-CURRENT	500025
package 作成のデフォルトを XFree86 4 に移行し、libc に新たに strnstr() 関数を追加した後の 5.0-CURRENT	500026
libc に新たに strcasestr() 関数を追加した後の 5.0-CURRENT	500027
smbfs のユーザランド部が取り込まれた後の 5.0-CURRENT	500028
C99 の新しい特定サイズの整数型追加後の 5.0-CURRENT	500028 (変更なし)
sendfile(2) の戻り値が変更された後の 5.0-CURRENT	500029
ファイルフラグにふさわしいサイズの `fflags_t` が導入された後の 5.0-CURRENT	500030
usb の構成要素の名称が変更された後の 5.0-CURRENT	500031
Perl 5.6.1 導入後の 5.0-CURRENT	500032
rc.conf(5) の `sendmail_enable` 変数が `NONE` という値をとれるようになった後の 5.0-CURRENT	500033
mtx_init() に 3 番目の引数が加わった後の 5.0-CURRENT	500034
GCC 3.1 が取り込まれた 5.0-CURRENT	500035
/usr/src に Perl がなくなった 5.0-CURRENT	500036
dlfunc(3) 追加後の 5.0-CURRENT	500037
構造体 sockbuf のメンバの型が一部変更され、順序が変更された後の 5.0-CURRENT	500038

Release	__FreeBSD_version
ヘッダで *BSD_FOO_T* の使用をやめ、_FOO_T_DECLARED を使うようになった後の 5.0-CURRENT。 また、この変数は bzip2(1) パッケージに対応したことが確実な目安としても使えます。	500039
ディスクラベルの内部構造の依存性を除く名目で行われた、ディスク関連の機能へのさまざまな変更を加えた後の 5.0-CURRENT	500040
libc に getopt_long(3) を加えた後の 5.0-CURRENT	500041
Binutils 2.13 にアップグレードした後の 5.0-CURRENT。このアップグレードには、新たな FreeBSD の emulation, vec および出力形式が含まれている。	500042
libc に pthread_XXX への弱いスタブを追加し、libXThrStub.so が obsolete になった後の 5.0-CURRENT。 5.0-RELEASE	500043
RELENG_5_0 が分岐した後の 5.0-CURRENT	500100
<sys/dkstat.h> は空なので include すべきではない	500101
d_mmap_t インターフェイス変更後の 5.0-CURRENT	500102
taskqueue_swi が Giant ロック無しで実行され、Giant ロックされて実行される taskqueue_swi_giant が追加された後の 5.0-CURRENT	500103
cdevsw_add() と cdevsw_remove() はもう存在しません。 MAJOR_AUTO 割り当て機能が登場しました	500104
cdevsw の新たな初期化方法が導入された後の 5.0-CURRENT	500105
devstat_add_entry() が devstat_new_entry() に置き換えられました	500106
Devstat のインターフェイス変更。sys/sys/param.h 1.149 を参照のこと	500107
トークンリングインターフェイスの変更	500108
vm_paddr_t の追加	500109
realpath(3) がスレッドセーフになった後の 5.0-CURRENT	500110
usbhid(3) が NetBSD と同期した後の 5.0-CURRENT	500111

Release	__FreeBSD_version
新たな NSS 実装と POSIX.1 準拠の getpw*_r, getgr*_r 関数が導入後の 5.0-CURRENT	500112
古い rc システムを削除した後の 5.0-CURRENT	500113
5.1-RELEASE	501000
RELENG_5_1 が分岐した後の 5.1-CURRENT	501100
sigtimedwait(2) と sigwaitinfo(2) の動作を修正した後の 5.1-CURRENT	501101
bus_dma_tag_create(9) に lockfunc と lockfuncarg フィールドを追加した後の 5.1-CURRENT	501102
GCC 3.3.1-pre 20030711 snapshot 導入後の 5.1-CURRENT	501103
5.1-CURRENT における twe の 3ware API の変更	501104
/bin と /sbin がダイナミックリンクされ、ライブラリを /lib に移動した 5.1-CURRENT	501105
Coda 6.x のカーネルサポートを追加した後の 5.1-CURRENT	501106
16550 UART 定数を <dev/sio/sioreg.h> から <dev/ic/ns16550.h> に移動した後の 5.1-CURRENT。 また、rtld が 無条件で libmap 機能をサポートした時点。	501107
PFIL_HOOKS API を更新した後の 5.1-CURRENT	501108
kiconv(3) を追加した後の 5.1-CURRENT	501109
cdevsw の open および close のデフォルトの操作を変更した後の 5.1-CURRENT	501110
cdevsw のレイアウトを変更した後の 5.1-CURRENT	501111
kobj の多重継承を追加した後の 5.1-CURRENT	501112
構造体 ifnet の if_xname が変更された後の 5.1-CURRENT	501113
/bin と /sbin をダイナミックリンクに変更した後の 5.1-CURRENT	501114
5.2-RELEASE	502000
5.2.1-RELEASE	502010
RELENG_5_2 が分岐した後の 5.2-CURRENT	502100
cxa_atexit/cxa_finalize 関数が libc に追加された後の 5.2-CURRENT	502101

Release	__FreeBSD_version
デフォルトの pthread ライブラリを libc_r から libpthread に変更した後の 5.2-CURRENT	502102
デバイスドライバ API の大規模パッチをあてた後の 5.2-CURRENT	502103
getopt_long_only() が追加された後の 5.2-CURRENT	502104
C に対して NULL が ((void *)0) になり、warning をより多く出すようになった 5.2-CURRENT	502105
pf がビルドおよびインストールされるようになった後の 5.2-CURRENT	502106
sparc64 で time_t を 64 ビットの値に変更した後の 5.2-CURRENT	502107
一部のヘッダで Intel C/C++ に対応し、 execve(2) をより厳密に POSIX に適合させた後の 5.2-CURRENT	502108
bus_alloc_resource_any API 導入後の 5.2-CURRENT	502109
UTF-8 ロケール追加後の 5.2-CURRENT	502110
getvfsent(3) API を削除した後の 5.2-CURRENT	502111
make に .warning 命令を追加した後の 5.2-CURRENT	502112

(2.2-STABLE は 2.2.5-RELESE 以後、 "2.2.5-STABLE" と呼ばれることがあります。) 見てのとおりこれは年・月というフォーマットになっていましたが、 バージョン 2.2 からより直接的にメジャー/マイナー番号を使うように変更になりました。 並行していくつかのブランチ (枝分かれしたバージョン) を開発する場合には、 リリースされた日付でそれらのリリースを分類することが不可能だからです (あなたが今 port を作成するときに、古い -CURRENT 達について心配する必要はありません。 これは参考のために挙げられているに過ぎないからです)。

16.8. bsd.port.mk の後に書くこと

`.include <bsd.port.mk>` の行の後には何も書かないようにしてください。 大抵の場合は Makefile の中程のどこかで bsd.port.pre.mk をインクルードして、 最後に bsd.port.pre.mk をインクルードすることによって避けることができます。

bsd.port.pre.mk/bsd.port.post.mk のペアか bsd.port.mk だけのどちらかだけをインクルードし、二つを混ぜないでください。

前者はいくつかの変数の定義だけをして Makefile でのテストに使用し、後者は残りを定義します。

以下は bsd.port.pre.mk で定義される重要な変数です (これは、すべてではありません。 完全なリストは bsd.port.mk を参照してください)。

変数名	解説
ARCH	uname -m で返されるアーキテクチャ。(例、i386)。
OPSYS	uname -s で返されるオペレーティングシステム (例、FreeBSD)。
OSREL	オペレーティングシステムのリリースバージョン (例、2.1.5, 2.2.7)。
OSVERSION	数字形式のオペレーティングシステムのバージョン、 上記の __FreeBSD_version と同じです。
PORTOBJFORMAT	システムのオブジェクトフォーマット (elf あるいは aout ただし、"最近の" FreeBSD のバージョンでは aout は廃止予定になっています)。
LOCALBASE	"local" ツリーのベース。 (例、/usr/local/)。
X11BASE	"X11" ツリーのベース。 (例、/usr/X11R6/)。
PREFIX	ports のインストール先 (PREFIXについてを参照)。

USE_IMAKE, USE_X_PREFIX あるいは MASTERDIR などの変数を定義する必要がある場合には、 bsd.port.pre.mk をインクルード前に定義してください。 他のものは bsd.port.pre.mk の前でも後でもかまいません。

以下は bsd.port.pre.mk の後に書けるものの例です。

```
# no need to compile lang/perl5 if perl5 is already in system
.if ${OSVERSION} > 300003
BROKEN= perl is in system
.endif

# only one shlib version number for ELF
.if ${PORTOBJFORMAT} == "elf"
TCL_LIB_FILE=  ${TCL_LIB}.${SHLIB_MAJOR}
.else
TCL_LIB_FILE=  ${TCL_LIB}.${SHLIB_MAJOR}.${SHLIB_MINOR}
.endif

# software already makes link for ELF, but not for a.out
post-install:
.if ${PORTOBJFORMAT} == "aout"
       ${LN} -sf liblinpack.so.1.0 ${PREFIX}/lib/liblinpack.so
.endif
```

BROKEN= と TCL_LIB_FILE= の後にスペースではなくタブを使うことを覚えていましたか? :-)

16.9. 付加的な文書のインストール

普通のマニュアルや info ファイルの他にユーザにとって有用だと思えるような文書がある場合には、PREFIX/shared/doc の下にインストールしてください。これは前記と同様 post-install ターゲットの中から行なうと良いでしょう。

まず、あなたの port のために新しいディレクトリを作ります。どの port の文書か簡単にわかるような名前にする必要がありますので、普通は PORTNAME を使うと良いでしょう。もちろん、ユーザが異なるバージョンのものを同時に使うことが予想される port の場合には PKGNAME をそのまま使っても構いません。

ユーザが /etc/make.conf でこの部分を禁止するために NOPORTDOCS という変数をセットしている場合には、これらの文書がインストールされないようにしてください。こんな具合です。

```
post-install:
.if !defined(NOPORTDOCS)
        ${MKDIR} ${DOCSDIR}
        ${INSTALL_MAN} ${WRKSRC}/docs/xvdocs.ps ${DOCSDIR}
.endif
```

ここでは、変数をいくつかと、それを Makefile で利用した時にどう展開されるかを説明します。

- ${DATADIR} は ${PREFIX}/shared/${PORTNAME} に展開されます。
- ${DOCSDIR} は ${PREFIX}/shared/doc/${PORTNAME} に展開されます。
- ${EXAMPLESDIR} は ${PREFIX}/shared/examples/${PORTNAME} に展開されます。

文書ファイルおよびディレクトリはすべて pkg-plist の中に %%PORTDOCS%% を頭につけて書く必要があります。たとえば、次のようにしてください。

```
%%PORTDOCS%%%%DOCSDIR%%/AUTHORS
%%PORTDOCS%%%%DOCSDIR%%/CONTACT
%%PORTDOCS%%@dirrm %%DOCSDIR%%
```

インストール時に pkg-message ファイルを利用してメッセージを表示することができます。詳細は pkg-message を使う のセクションを参照してください。

pkg-message ファイルを pkg-plist に加える必要はありません。

16.10. ディレクトリ構成

インストール時には PREFIX の正しいサブディレクトリにファイルを置くように心がけてください。ソフトウェアによっては新しいディレクトリを一つ作って、ファイルを全部それに入れてしまうものがありますが、それは良くありません。また、バイナリ、ヘッダファイルとマニュアル以外のすべてを lib というディレクトリに入れてしまう port もありますが、これも BSD 的なファイルシステム構成とはうまく合いません。

これは以下のように分散すべきです。 etc にセットアップ/コンフィグレーションファイル、 libexec に内部で使用されるプログラム (コマンドラインから呼ばれることのないコマンド)、 sbin に管理者用のコマンド、 info に GNU Info 用の文書、 そして share にアーキテクチャに依存しないファイルが入ります。 詳細については hier(7) を参照してください。 /usr の構成方針はほとんどそのまま /usr/local にもあてはまります。 USENET "ニュース"を扱う ports は例外です。 これらはファイルのインストール先として PREFIX/news を使用します。

16.11. 空のディレクトリの削除

ports は削除の際に、 自分自身を消去したあとに (ディレクトリの) 削除をするようにしてください。 これは大抵の場合 `@dirrm` の行を ports が作成するすべてのディレクトリについて加えることによって実現できます。
親ディレクトリは子ディレクトリを先に消さないと消せないことに注意してください。

```
 :
lib/X11/oneko/pixmaps/cat.xpm
lib/X11/oneko/sounds/cat.au
 :
@dirrm lib/X11/oneko/pixmaps
@dirrm lib/X11/oneko/sounds
@dirrm lib/X11/oneko
```

といった感じです。

しかし時として、 他の port とディレクトリを共有しているために `@dirrm` がエラーを返すことがあります。 `rmdir` を `@unexec` から呼びだすことによって、 警告(warning)なしで空のディレクトリのみを削除することができます。

```
@unexec rmdir %D/shared/doc/gimp 2>/dev/null || true
```

これを使えば、たとえ他の port がファイルをインストールしていて PREFIX/shared/doc/gimp が空でない場合でもエラーメッセージは表示されませんし、 pkg_delete(1) が異常終了することもありません。

16.12. UID

あなたの port が、 インストールされるシステム上に特定のユーザを必要とする場合は pkg-install スクリプトから `pw` コマンドを実行して自動的にそのユーザを追加するようにしてください。 net/cvsup-mirror の port が参考になるでしょう。

あなたの port がバイナリの package としてインストールされる場合とコンパイルされる場合の両方で、同じユーザー/グループ ID を使わなければならないのなら、50 から 999 の間で空いている UID を選んで登録してください。 japanese/Wnn6 の port が参考になるでしょう。

既にシステムや他の port で利用されている UID を使わないように十分注意してください。

現在の 50 から 999 までの間の UID は以下のとおりです。

```
bind:*:53:53:Bind Sandbox:/:/sbin/nologin
majordom:*:54:54:Majordomo Pseudo User:/usr/local/majordomo:/nonexistent
cyrus:*:60:60:the cyrus mail server:/nonexistent:/nonexistent
gnats:*:61:1:GNATS database owner:/usr/local/shared/gnats/gnats-db:/bin/sh
proxy:*:62:62:Packet Filter pseudo-user:/nonexistent:/nonexistent
uucp:*:66:66:UUCP pseudo-user:/var/spool/uucppublic:/usr/libexec/uucp/uucico
xten:*:67:67:X-10 daemon:/usr/local/xten:/nonexistent
pop:*:68:6:Post Office Owner (popper):/nonexistent:/sbin/nologin
wnn:*:69:7:Wnn:/nonexistent:/nonexistent
pgsql:*:70:70:PostgreSQL pseudo-user:/usr/local/pgsql:/bin/sh
oracle:*:71:71::0:0:Oracle:/usr/local/oracle7:/sbin/nologin
ircd:*:72:72:IRC daemon:/nonexistent:/nonexistent
ircservices:*:73:73:IRC services:/nonexistent:/nonexistent
ifmail:*:75:66:Ifmail user:/nonexistent:/nonexistent
www:*:80:80:World Wide Web Owner:/nonexistent:/sbin/nologin
alias:*:81:81:QMail user:/var/qmail/alias:/nonexistent
qmaild:*:82:81:QMail user:/var/qmail:/nonexistent
qmaill:*:83:81:QMail user:/var/qmail:/nonexistent
qmailp:*:84:81:QMail user:/var/qmail:/nonexistent
qmailq:*:85:82:QMail user:/var/qmail:/nonexistent
qmailr:*:86:82:QMail user:/var/qmail:/nonexistent
qmails:*:87:82:QMail user:/var/qmail:/nonexistent
mysql:*:88:88:MySQL Daemon:/var/db/mysql:/sbin/nologin
vpopmail:*:89:89:VPop Mail User:/usr/local/vpopmail:/nonexistent
firebird:*:90:90:Firebird Database Administrator:/usr/local/firebird:/bin/sh
mailman:*:91:91:Mailman User:/usr/local/mailman:/sbin/nologin
gdm:*:92:92:GDM Sandbox:/:/sbin/nologin
jabber:*:93:93:Jabber Daemon:/nonexistent:/nonexistent
p4admin:*:94:94:Perforce admin:/usr/local/perforce:/sbin/nologin
interch:*:95:95:Interchange user:/usr/local/interchange:/sbin/nologin
squeuer:*:96:96:SQueuer Owner:/nonexistent:/bin/sh
mud:*:97:97:MUD Owner:/usr/local/shared/dgd:/bin/sh
msql:*:98:98:mSQL-2 pseudo-user:/var/db/msqldb:/bin/sh
rscsi:*:99:99:Remote SCSI:/usr/local/rscsi:/usr/local/sbin/rscsi
squid:*:100:100:squid caching-proxy pseudo user:/usr/local/squid:/sbin/nologin
quagga:*:101:101:Quagga route daemon pseudo user:/usr/local/etc/quagga:/sbin/nologin
ganglia:*:102:102:Ganglia User:/nonexistent:/sbin/nologin
sgeadmin:*:103:103:Sun Grid Engine Admin:/nonexistent:/sbin/nologin
slimserv:*:104:104:Slim Devices SlimServer pseudo-user:/nonexistent:/sbin/nologin
fido:*:111:111:Fido System:/usr/local/fido:/bin/sh
postfix:*:125:125:Postfix Mail System:/var/spool/postfix:/sbin/nologin
rbldns:*:153:153:rbldnsd pseudo-user:/nonexistent:/sbin/nologin
sfs:*:171:171:Self-Certifying File System:/nonexistent:/sbin/nologin
agk:*:172:172:AquaGateKeeper:/nonexistent:/nonexistent
ldap:*:389:389:OpenLDAP Server:/nonexistent:/sbin/nologin
drweb:*:426:426:Dr.Web Mail Scanner:/nonexistent:/sbin/nologin
qtss:*:554:554:Darwin Streaming Server:/nonexistent:/sbin/nologin
ircdru:*:555:555:Russian hybrid IRC server:/nonexistent:/bin/sh
bacula:*:910:910:Bacula Daemon:/var/db/bacula:/sbin/nologin
```

以下は、現在予約されている GID の一覧です。

```
bind:*:53:
cyrus:*:60:
proxy:*:62:
authpf:*:63:
uucp:*:66:
dialer:*:68:
network:*:69:
pgsql:*:70:
www:*:80:
qnofiles:*:81:
qmail:*:82:
mailman:*:91:
postfix:*:125:
maildrop:*:126:
rbldns:*:153:
qtss:*:554:
ircdru:*:555:
```

このリストを最新の状態に保つためにも、この範囲の UID や GID を予約するような port を作ったり、既存の port にそのような改変を行なってわたしたちに送るときには UID の予約に関する注意書きをつけてください。

16.13. 合理的な port

Makefile は単純かつ適切であるべきです。もし、Makefile を数行短かくできたり、もっと読みやすくできるのであればそうしてください。たとえば、シェルの `if` 構文を使うかわりに make の `.if` 構文を使う、`EXTRACT*` の再定義で代用できるのであれば `do-extract` を再定義しない、`CONFIGURE_ARGS += --prefix=${PREFIX}` とするかわりに `GNU_CONFIGURE` とする、などです。

何かをするのに自分で新しくコードをたくさん書かなければならなくなった場合は、戻って bsd.port.mk であなたがやろうとしていることが既に実装されていないか見直してください。
読むのは大変ですが、難しく見える問題で bsd.port.mk が簡単な解決法を提供しているものが数多くあります。

16.14. `CC` および `CXX` の尊重

Port は `CC` および `CXX` 変数を尊重すべきです。ここで言いたいのは、port は、既存の値を上書きしてこれらの変数をまるごと設定しなおすべきではなく、
その代わり必要な値を既存の値に追加してゆくべきだということです。そうすれば、すべての ports に影響するビルドオプションをグローバルに設定できます。

Port がこれらの変数を尊重しない場合は、Makefile に `NO_PACKAGE=ignores either cc or cxx` を追加してください。

`CC` と `CXX` 変数を尊重している Makefile の例を次に示します。`?=` に注意してください。

```
CC ?= gcc
```

```
CXX ?= g++
```

こちらは、CC 変数も CXX 変数も尊重していない例です。

```
CC = gcc
```

```
CXX = g++
```

FreeBSD システム上では、CC および CFLAGS 変数は、どちらも /etc/make.conf で定義できます。最初の例では、システム全体の定義を保存している /etc/make.conf で値がすでに設定されてない場合に限って、値を設定します。 2 番目の例では、すでに設定されていた内容を上書きしてしまいます。

16.15. CFLAGS の尊重

CFLAGS 変数は尊重すべきです。 ここでいいたいのは、port は、既存の値を上書きしてこの変数をまるごと設定しなおすべきではなく、その代わり必要な値を既存の値に追加してゆくべきだということです。 そうすれば、すべての ports に影響するビルドオプションをグローバルに設定できます。

port がこれを尊重しない場合は、 NO_PACKAGE=ignores cflags を Makefile に加えてください。

CFLAGS 変数をきちんと考慮した Makefile の例を以下に示します。 += の部分に注目してください。

```
CFLAGS += -Wall -Werror
```

次は CFLAGS 変数を考慮しない Makefile の例です。

```
CFLAGS = -Wall -Werror
```

CFLAGS 変数は、FreeBSD システムの /etc/make.conf で定義されています。最初の例では既存の定義を保存しつつ CFLAGS 変数にオプションフラグを追加しているのに対し、二番目の例では既存の定義をすべて無効にしてしまっています。

16.16. コンフィグレーション (設定) ファイル

もしあなたの port が設定ファイルを PREFIX/etc に置く必要がある場合には、それを単純にインストールしたり、 pkg-plist に加えてはいけません。こうしてしまうと pkg_delete(1) によってユーザが苦労して作ったファイルが消えてしまったり、新しくインストールする時に上書きされてしまったりします。

かわりに見本となるファイルをサフィックス (filename.sample が良いでしょう) を付けてインストールしてメッセージを表示し、
ソフトウェアを動かす前にユーザがそのファイルをコピーして編集をしなければならないことを知らせましょう。

16.17. フィードバック

port を作るためにソフトウェアに変更を加えたら、
なるべく原作者にその旨を伝えてパッチ等を送ってください。
これらが次のリリースに取り入れられればアップグレードが楽になります。

16.18. README.html

README.html というファイルを含めてはいけません。 このファイルは、cvs コレクションの一部ではなく、 `make readme` コマンドで生成されるファイルです。

16.19. Port に BROKEN, FORBIDDEN などの印をつける

ある port にセキュリティ脆弱性があることが判明したり、
根本的に壊れてしまい修正に何時間もの注意深い作業が必要になったり、
基本的には廃れてしまったものの、 何らかの理由で ports ツリーには残される (もちろんあとで修正しますよね?) という日が来るのは避けられません。 ある port が壊れていることを示すために、port の Makefile では 3 つの `make` 変数が使えます。以下の `make` 変数の値は、 その port が壊れている理由を説明するためにユーザに示されます。 それぞれの make 変数は、ユーザと Makefile を処理する自動化システムに対して根本的に異なる意味を伝えますので、正しい `make` 変数をお使いください。

- `BROKEN` は、動作しないためインストールすべきでない port 用のものです。これは、ユーザがその port をインストールしないようにしますが、 `BROKEN` とされた port は Bento クラスタで引き続きビルドされます。 ユーザには port をインストールしてほしくないけれども Bento ではビルドしてほしい場合は、port を `BROKEN` にしてください。
- `FORBIDDEN` は、セキュリティ脆弱性があったり、その port をインストールすると FreeBSD システムの安全性に重大な懸念を生じる (たとえば、セキュアでないという評判があるプログラムや、容易に悪用できるサービスを提供するプログラムなど) port 用のものです。 あるソフトウェアの一部に脆弱性があることが判明し、 修正がリリースされていない場合は `FORBIDDEN` にすべきです。 理想的には、セキュリティ脆弱性が発見された時は、 脆弱性を抱えた FreeBSD ホストの数を減らすために、 ただちに ports を更新すべきです (我々は、セキュアであるという評判を得たいのです)。 しかし、脆弱性が公表されてから、脆弱性を抱えたソフトウェアの新しい版がリリースされるまでに無視できない時間があくことがままあります。 セキュリティ以外の理由で port を `FORBIDDEN` にしないでください。
- `IGNORE` は、どんな理由であれビルドすべきではない port 用です。 ユーザも Bento クラスタも、どんな状況であれ `IGNORE` とされた port はビルドしません。 嘘だと思うなら、port のビルドを妨げるのに `IGNORE` を使ってみてください。

この変数を使うのは、port が更新できない場合の最後の手段にしてください。 ずっと壊れたままの port は、ports ツリーから完全に削除すべきです。

16.20. 必要な回避策

古いバージョンの FreeBSD のソフトウェアにあるバグを回避する必要があることがあります。

- make(1) は、少なくとも 4.8 と 5.0 を含むいくつかのバージョンで、OSVERSION に基づく比較に関してバグがあります。これは、make describe の最中にエラーを起こすことになりやすく (したがって make index 全体も失敗することに) なります。回避策は、条件比較を括弧にいれることで、たとえば

```
if ( ${OSVERSION} > 500023 )
```

となります。4.9 と 5.2 で port のインストールテストを行っても、 この問題は見つかりません。

16.21. その他諸々

ファイル pkg-descr と pkg-plist はそれぞれ二重にチェックしてください。再検討してもっと良い記述があればそれに置きかえてください。

GNU General Public License (GNU一般公有使用許諾) のコピーは (すでにあるので) コピーしないでください。 お願いします。

法律に関することには十分注意をはらってください。
わたしたちに法律に反するような形でソフトウェアの配布をさせないでください!

16.22. 困ったら...

わたしたちに質問を送る前に、 既存の port の例と bsd.port.mk をちゃんと読んでください! *;)*

それでもわからないことがあったら一人で悩まないでどんどん質問してください! :-)

Chapter 17. Makefile のサンプル

これは port の Makefile を作る際のお手本です。かぎかっこ ([]) 内のコメントは忘れずに取ってください。

変数の順番、段落の間の空行など、Makefile を作るときはなるべくこの形式に従ってください。この形式は重要な情報が簡単に見つけられるように設計されています。portlint を使って Makefile をチェックすることが推奨されています。

```
[ヘッダ ... どのような port の Makefile かすぐにわかるようになっています]
# New ports collection makefile for:    xdvi
["version required" 行は、PORTVERSION 変数では port のバージョンを
十分に表現できない場合にのみ必要です。]
# Date created: 26 May 1995
[このソフトウェアを最初に FreeBSD に port した人の名前、つまり、
 この Makefile の最初の版を書いた人です。この port をアップグレー
 ドするとき、この行も変えないでください。]
# Whom:         Satoshi Asami <asami@FreeBSD.org>
#
# $FreeBSD$
[ ^^^^^^^^^ この部分は、CVS ツリーに入れる時に自動的に RCS の ID 文字列に
 置き換えられます。]
#

[port 自体、およびオリジナルのソースを取ってくるところを記述する部分。
 最初は必ず PORTNAME と PORTVERSION, そして必要なら PKGNAME,
 CATEGORIES, 続いて MASTER_SITES が置かれ、さらに MASTER_SITE_SUBDIR が
 置かれることもあります。必要なら PKGNAMEPREFIX と PKGNAMESUFFIX が
 それに続き、そして DISTNAME, EXTRACT_SUFX, DISTFILES が、
 また、その後に必要に応じて EXTRACT_ONLY が置かれます。]
PORTNAME=       xdvi
PORTVERSION=    18.2
CATEGORIES=     print
[MASTER_SITE_* マクロを使用しない場合は、
最後のスラッシュを忘れないように ("/")!]
MASTER_SITES= ${MASTER_SITE_XCONTRIB}
MASTER_SITE_SUBDIR= applications
DISTNAME=       xdvi-pl18
[ソースファイルが標準の ".tar.gz" 形式でない時にこれを使いましょう]
EXTRACT_SUFX=   .tar.Z

[配布パッチセクション -- ない場合もあります]
PATCH_SITES=    ftp://ftp.sra.co.jp/pub/X11/japanese/
PATCHFILES=     xdvi-18.patch1.gz xdvi-18.patch2.gz

[保守責任者 -- これは *必ず* 必要です。担当者（あなた）自身、あるいは
 担当者に素早く連絡をとれる人のアドレスを書いてください。どうしてもこ
 こに自分のアドレスを書くのがいやな人は "ports@FreeBSD.org" と書いて
 もいいです]
MAINTAINER=     asami@FreeBSD.org
```

```
COMMENT=       A DVI Previewer for the X Window System

[依存するport -- ない場合もあります]
RUN_DEPENDS=    gs:${PORTSDIR}/print/ghostscript
LIB_DEPENDS=    Xpm.5:${PORTSDIR}/graphics/xpm

[ここには標準の bsd.port.mk の変数で、上のどれにもあてはまらないものを
 書きます]
[コンフィグレーション、コンパイル、インストールなどの時に質問をする
 なら…]
IS_INTERACTIVE=yes
[${DISTNAME} 以外のディレクトリにソースが展開されるなら…]
WRKSRC= ${WRKDIR}/xdvi-new
[配布されているパッチが ${WRKSRC} に対する相対パスで作られてい
 い場合にこの変数の指定が必要かも…]
PATCH_DIST_STRIP=       -p1
[GNU autoconf によって生成された "configure" スクリプトを走らせたいなら…]
GNU_CONFIGURE=  yes
[/usr/bin/makeでなく、GNU make を使わないといけないなら…]
USE_GMAKE=      yes
[これが X のアプリケーションで、"xmkmf -a" を走らせたいなら…]
USE_IMAKE=      yes
[などなど]

[下の方のルールで使う非標準の変数]
MY_FAVORITE_RESPONSE=   "yeah, right"

[そして、特別なターゲット、使用順に]
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

[最後には必ず]
.include <bsd.port.mk>
```

Chapter 18. パッキングリストの自動生成

まず、あなたの port に pkg-plist がないことを除けば完成していることを確認してください。

次に、あなたの port をインストールする一時ディレクトリを作成して、依存するものをすべてインストールしてください。 *port-type* は X アプリケーションではない port については `local`、 XFree86 4 またはそれより前の XFree86 のディレクトリ階層にインストールする ports については、それぞれ `x11-4` または `x11` にすべきです。

```
# mkdir /var/tmp/port-name
# mtree -U -f /etc/mtree/BSD.port-type.dist -d -e -p /var/tmp/port-name
# make depends PREFIX=/var/tmp/port-name
```

このディレクトリ構造を新しいファイルに保存してください。

```
# (cd /var/tmp/port-name && find -d * -type d) | sort > OLD-DIRS
```

空の pkg-plist ファイルを作成してください。

```
# touch pkg-plist
```

もしあなたの port が `PREFIX` にちゃんと従うなら、ここで port をインストールしてパッキングリストを作ることができます。

```
# make install PREFIX=/var/tmp
# (cd /var/tmp/port-name && find -d * \! -type d) | sort > pkg-plist
```

新しく生成されたディレクトリはすべてパッキングリストに追加する必要があります。

```
# (cd /var/tmp/port-name && find -d * -type d) | sort | comm -13 OLD-DIRS - | sort -r
| sed -e 's#^#@dirrm #' >> pkg-plist
```

最後にパッキングリストを手で整える必要があります; すべてが自動化されているわけではありません。マニュアルはパッキングリストに記述するのではなく、port の Makefile 中の `MAN_n_` に記述しなければなりません。ユーザ設定ファイルは削除するか filename.sample としてインストールされなければなりません。また info/dir ファイルはリストに含めず、info ファイルに記述されているように、適切な install-info 行に追加しなければなりません。port によってインストールされるライブラリは、共有ライブラリのセクションで示したように記載されるべきです。

または、/usr/ports/Tools/scripts/ にある `plist` スクリプトを使ってパッキングリストを自動的に生成してください。

Chapter 19. この文書と ports システムの変更

もしあなたが、たくさんの ports の保守をしているのであれば、 FreeBSD ports メーリングリストの内容を読むことを考えてください。 ports のしくみについての重要な変更点はここにアナウンスされます。 最新の変更点については、いつでも、 bsd.port.mk の CVS ログで詳細な情報を得ることができます。

port メンテナを補助するほかのリソースとして、 `bento` クラスタに置かれているパッケージビルド記録とエラー一覧、また FreeBSD Ports distfiles 調査があります。