FreeBSD Porter’s Handbook

Inhaltsverzeichnis

1. Einführung . . . 5
2. Einen neuen Port erstellen . . . 6
3. Einen neuen Port erstellen . . . 7
   3.1. Das Makefile schreiben . . . 7
   3.2. Die Beschreibungsdateien erstellen . . . 7
   3.3. Die Checksummendatei erzeugen . . . 9
   3.4. Den Port testen . . . 9
   3.5. Ihren Port mit `portlint` überprüfen . . . 10
   3.6. Den neuen Port einreichen . . . 10
4. Einen Port in aller Ruhe erstellen . . . 12
   4.1. Die Funktionsweise . . . 12
   4.2. Den originalen Quelltext besorgen . . . 13
   4.3. Den Port bearbeiten . . . 14
   4.4. Fehlerbehebung (Patches) . . . 14
   4.5. Konfigurieren . . . 16
   4.6. Handhabung von Benutzereingaben . . . 16
5. Die Konfiguration des Makefile . . . 17
   5.1. Der originale Quelltext . . . 17
   5.2. Bezeichnungen . . . 17
   5.3. Kategorisierung . . . 23
   5.4. Die Distributionsdateien . . . 30
   5.5. `MAINTAINER` . . . 41
   5.6. `COMMENT` . . . 42
   5.7. Abhängigkeiten (dependencies) . . . 42
   5.8. `MASTERDIR` . . . 47
   5.9. Manualpages . . . 48
   5.10. Info-Dateien . . . 50
   5.11. Makefile-Optionen . . . 50
   5.12. Die Festlegung des Arbeitsverzeichnisses . . . 53
   5.13. Konfliktbehandlung . . . 54
   5.14. Installation von Dateien . . . 55
6. Besonderheiten . . . 59
   6.1. Shared-Libraries . . . 59
   6.2. Ports mit beschränkter Verbreitung . . . 59
   6.3. Build-Mechanismen . . . 61
   6.4. Benutzung von GNU autotools . . . 63
   6.5. Benutzung von GNU `gettext` . . . 65
   6.6. Die Benutzung von `perl` . . . 67

6.7. Benutzung von X11 . . . 68
6.8. Benutzung von GNOME . . . 71
6.9. Benutzung von Qt . . . 71
6.10. Benutzung von KDE . . . 74
6.11. Benutzung von Java . . . 76
6.12. Webanwendungen, Apache und PHP . . . 80
6.13. Python benutzen . . . 83
6.14. Benutzung von Tcl/Tk . . . 84
6.15. Emacs benutzen . . . 85
6.16. Ruby benutzen . . . 85
6.17. SDL verwenden . . . 86
6.18. wxWidgets verwenden . . . 87
6.19. Verwendung von Lua . . . 92
6.20. Xfce verwenden . . . 97
6.21. Mozilla verwenden . . . 98
6.22. Benutzung von Datenbanken . . . 99
6.23. Starten und Anhalten von Diensten (rc Skripten) . . . 99
6.24. Hinzufügen von Benutzern und Gruppen . . . 101
6.25. Von Kernelquellen abhängige Ports . . . 102
7. Fortgeschrittene pkg-plist-Methoden . . . 103
7.1. Änderungen an pkg-plist mit Hilfe von make-Variablen . . . 103
7.2. Leere Verzeichnisse . . . 104
7.3. Konfigurationsdateien . . . 105
7.4. Dynamische oder statische Paketliste . . . 105
7.5. Automatisiertes Erstellen von Paketlisten . . . 106
8. Die pkg-* Dateien . . . 108
8.1. pkg-message . . . 108
8.2. pkg-install . . . 108
8.3. pkg-deinstall . . . 109
8.4. pkg-req . . . 109
8.5. Ändern der Namen der pkg-* Dateien . . . 109
8.6. Nutzung von SUB_FILES und SUB_LIST . . . 109
9. Ihren Port testen . . . 111
9.1. make describe ausführen . . . 111
9.2. Portlint . . . 111
9.3. Port Tools . . . 111
9.4. PREFIX und DESTDIR . . . 112
9.5. Die Tinderbox . . . 113
10. Einen existierenden Port aktualisieren . . . 114
10.1. Patches mit CVS erstellen . . . 115
10.2. Die Dateien UPDATING und MOVED . . . 117

11. Sicherheit der Ports . . . 118
11.1. Warum Sicherheit so wichtig ist . . . 118
11.2. Sicherheitslücken schliessen . . . 118
11.3. Die Community informiert halten . . . 119
12. Was man machen respektive vermeiden sollte . . . 124
12.1. Einführung . . . 124
12.2. WRKDIR . . . 124
12.3. WRKDIRPREFIX . . . 124
12.4. Unterschiedliche Betriebssysteme und Betriebssystemversionen . . . 124
12.5. __FreeBSD_version Werte . . . 126
12.6. Etwas hinter die bsd.port.mk-Anweisung schreiben . . . 175
12.7. Benutzen Sie die exec-Anweisung in Wrapper-Skripten . . . 176
12.8. Aufgaben vernünftig lösen . . . 177
12.9. Berücksichtigen Sie sowohl CC als auch CXX . . . 177
12.10. Berücksichtigen Sie CFLAGS . . . 178
12.11. Threading-Bibliotheken . . . 178
12.12. Rückmeldungen . . . 179
12.13. README.html . . . 179
12.14. Einen Port durch BROKEN, FORBIDDEN oder IGNORE als nicht installierbar markieren . . . 179
12.15. Kennzeichnen eines Ports zur Entfernung durch DEPRECATED oder EXPIRATION_DATE . . . 181
12.16. Vermeiden Sie den Gebrauch des .error-Konstruktes . . . 181
12.17. Verwendung von sysctl . . . 182
12.18. Erneutes Ausliefern von Distfiles . . . 182
12.19. Verschiedenes . . . 183
13. Beispiel eines Makefile . . . 184
14. Auf dem Laufenden bleiben . . . 186
14.1. FreshPorts . . . 186
14.2. Die Webschnittstelle zum Quelltext-Repository . . . 186
14.3. Die FreeBSD Ports-Mailingliste . . . 186
14.4. Der Cluster zum Bauen von FreeBSD-Ports auf pointyhat.FreeBSD.org . . . 186
14.5. Der FreeBSD Ports-Distfile-Scanner . . . 187
14.6. Das FreeBSD Ports-Monitoring-System . . . 187

Kapitel 1. Einführung

Die Ports-Sammlung von FreeBSD ist der gebräuchlichste Weg, um Anwendungen ("Ports") unter FreeBSD zu installieren. Wie alles andere in FreeBSD auch, ist sie hauptsächlich das Ergebnis der Arbeit von Freiwilligen. Es ist wichtig, diesen Aspekt beim Lesen im Hinterkopf zu behalten.

In FreeBSD kann jeder einen neuen Port einsenden oder sich dazu bereit erklären, einen bereits vorhandenen Port zu pflegen, sofern der Port derzeit keinen Maintainer hat – dazu sind keine besonderen Rechte nötig.

Kapitel 2. Einen neuen Port erstellen

Sie sind also daran interessiert, einen neuen Port zu erstellen oder einen vorhandenen zu aktualisieren? Großartig!

Die folgenden Kapitel beinhalten einige Richtlinien, um einen neuen Port für FreeBSD zu erstellen. Wenn Sie einen vorhandenen Port auf den neuesten Stand bringen wollen, sollten Sie mit Einen existierenden Port aktualisieren fortfahren.

Wenn Ihnen dieses Dokument nicht detailliert genug ist, sollten Sie einen Blick in /usr/ports/Mk/bsd.port.mk werfen. Das Makefile jedes Ports bindet diese Datei ein. Auch wenn Sie nicht täglich mit Makefiles arbeiten, sollten Sie gut damit zurecht kommen, da die Datei gut dokumentiert ist und Sie eine Menge Wissen daraus erlangen können. Zusätzlich können Sie speziellere Fragen an die FreeBSD ports-Mailingliste stellen.



Nur ein Bruchteil der Variablen (VAR), die von Ihnen gesetzt werden können, finden hier Erwähnung. Die meisten von ihnen (wenn nicht sogar alle) sind am Anfang von /usr/ports/Mk/bsd.port.mk erläutert. Beachten Sie bitte, dass diese Datei eine nicht standardkonforme Tabulator-Einstellung verwendet. Emacs und Vim sollten diese Einstellung jedoch automatisch beim Öffnen der Datei setzen. Sowohl vi(1) als auch ex(1) können mit dem Befehl `:set tabstop=4` dazu gebracht werden, die Datei richtig anzuzeigen, wenn sie geöffnet wird.

Sind Sie auf der Suche nach einer neuen Aufgabe? Dann sehen Sie sich bitte die Ports-Wunschliste an und prüfen Sie, ob Sie an einem dieser Ports arbeiten können.

Kapitel 3. Einen neuen Port erstellen

Dieser Abschnitt beschreibt, wie Sie schnell einen neuen Port erstellen können. In vielen Fällen ist dies allerdings nicht ausreichend, dann werden Sie in diesem Buch weiterlesen müssen.

Als Erstes besorgen Sie sich das Original-Tarball (komprimiertes Archiv) und legen es im `DISTDIR` ab, welches standardmäßig /usr/ports/distfiles ist.



Im Folgenden wird angenommen, dass die Software unverändert kompiliert werden konnte, dass also keinerlei Änderungen nötig waren, um den Port auf Ihrem FreeBSD-Rechner zum Laufen zu bringen. Falls Sie Änderungen vornehmen mussten, werden Sie auch den nächsten Abschnitt beachten müssen.

3.1. Das Makefile schreiben

Ein minimales Makefile sieht in etwa so aus:

```
# 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>
```

Versuchen Sie es zu verstehen. Machen Sie sich keine Gedanken um die `$FreeBSD$`-Zeile, diese wird automatisch vom CVS eingefügt, wenn der Port in den Haupt-Ports-Tree importiert wird. Ein detailliertes Beispiel finden Sie im Abschnitt sample Makefile.

3.2. Die Beschreibungsdateien erstellen

Es gibt zwei Beschreibungsdateien, die für jeden Port benötigt werden, ob sie tatsächlich im Paket enthalten sind oder nicht. Dies sind pkg-descr und pkg-plist. Der pkg- Präfix unterscheidet sie von anderen Dateien.

3.2.1. pkg-descr

Diese enthält eine längere Beschreibung des Ports. Einer oder mehrere Absätze, die kurz und prägnant erklären, was der Port macht, sind ausreichend.



pkg-descr enthält *keine* Anleitung oder detaillierte Beschreibung wie der Port benutzt oder kompiliert wird! *Bitte seien Sie vorsichtig, wenn Sie aus dem README oder der Manualpage kopieren*; Diese sind oft keine prägnanten Beschreibungen des Ports oder sie sind in einem ungünstigen Format (Manualpages haben z.B. bündige Zwischenräume). Wenn es für die portierte Software eine offizielle Webseite gibt, sollten Sie diese hier angeben. Fügen Sie hierzu *eine* der Webseiten mit dem Präfix `WWW:` ein, damit automatische Werkzeuge korrekt arbeiten.

Das folgende Beispiel zeigt wie Ihre pkg-descr aussehen sollte:

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

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

3.2.2. pkg-plist

Diese Datei enthält eine Liste aller Dateien, die von diesem Port installiert werden. Sie wird auch die "Packliste" genannt, da das Paket durch die hier aufgeführten Dateien erstellt wird. Die Pfadangaben sind relativ zum Installationspräfix (für gewöhnlich /usr/local oder /usr/X11R6). Wenn Sie die `MANn`-Variablen verwenden (was Sie auch machen sollten), führen Sie hier keine Manualpages auf. Wenn der Port während der Installation Verzeichnisse erstellt, stellen Sie sicher entsprechende `@dirrm`-Zeilen einzufügen, um die Verzeichnisse zu entfernen, wenn das Paket gelöscht wird.

Hier ist ein kleines Beispiel:

```
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
```

Für weitere Details zur Packliste lesen Sie in der pkg_create(1) Manualpage nach.



Es wird empfohlen alle Dateinamen in dieser Datei alphabetisch sortiert zu halten. Das erlaubt Ihnen die Änderungen bei einem Upgrade Ihres Ports deutlich einfacher zu Überprüfen.



Eine Packlist von Hand zu erzeugen kann eine sehr mühsame Aufgabe sein. Wenn der Port eine große Anzahl Dateien installiert, kann es Zeit sparen, eine Packliste automatisch zu erstellen.

Es gibt nur einen Fall, in dem pkg-plist weggelassen werden kann. Wenn der Port nur eine handvoll Dateien und Verzeichnisse installiert, können diese in den Variablen `PLIST_FILES` und `PLIST_DIRS` im Makefile aufgelistet werden. Zum Beispiel könnten wir im obigen Beispiel ohne pkg-plist für den oneko-Port auskommen, indem wir die folgenden Zeilen ins Makefile einfügen:

```
PLIST_FILES=    bin/oneko \
                lib/X11/app-defaults/Oneko \
                lib/X11/oneko/cat1.xpm \
                lib/X11/oneko/cat2.xpm \
                lib/X11/oneko/mouse.xpm
PLIST_DIRS=     lib/X11/oneko
```

Natürlich sollte `PLIST_DIRS` ungesetzt bleiben, wenn der Port keine eigenen Verzeichnisse installiert.

Der Preis für diese Art die Dateien eines Ports anzugeben ist, dass man keine Befehlsfolgen wie in pkg_create(1) nutzen kann. Deshalb ist es nur für einfache Ports geeignet und macht diese noch einfacher. Gleichzeitig bringt es den Vorteil die Anzahl der Dateien in der Ports-Sammlung zu reduzieren. Deshalb ziehen Sie bitte diese Vorgehensweise in Erwägung, bevor Sie pkg-plist benutzen.

Später werden wir uns ansehen, wie pkg-plist und `PLIST_FILES` benutzt werden können, um anspruchsvollere Aufgaben zu erfüllen.

3.3. Die Checksummendatei erzeugen

Geben Sie einfach `make makesum` ein. Die Regeln von Make sorgen dafür, dass die Datei distinfo automatisch erstellt wird.

Wenn sich die Checksumme einer heruntergeladenen Datei regelmäßig ändert und Sie sicher sind, dass Sie der Quelle trauen können (weil sie z.B. von einer Hersteller-CD oder täglich erstellter Dokumentation stammt), sollten Sie diese Dateien in der Variable `IGNOREFILES` angeben. Dann wird die Checksumme für diese Datei bei `make makesum` nicht berechnet, sondern auf `IGNORE` gesetzt.

3.4. Den Port testen

Sie sollten sicherstellen, dass die Port-Regeln genau das einhalten, was Sie von ihnen erwarten, auch beim Erzeugen eines Pakets aus dem Port. Dies sind die wichtigen Punkte, die Sie überprüfen sollten.

- pkg-plist enthält nichts, das nicht von Ihrem Port installiert wurde.
- pkg-plist enthält alles, was von Ihrem Port installiert wurde.
- Ihr Port kann mit Hilfe von `make reinstall` mehrmals installiert werden.

- Ihr Port räumt bei der Deinstallation hinter sich auf.

Procedure: Empfohlene Testreihenfolge

1. `make install`
2. `make package`
3. `make deinstall`
4. `pkg_add Paket-Name`
5. `make deinstall`
6. `make reinstall`
7. `make package`

Stellen Sie bitte sicher, dass während `make package` und `make deinstall` keine Warnungen ausgegeben werden. Nach Schritt 3 überprüfen Sie bitte, ob alle neuen Verzeichnisse korrekt entfernt wurden. Und versuchen Sie die Software nach Schritt 4 zu benutzen, um sicherzustellen, dass sie korrekt funktioniert, wenn diese aus einem Paket installiert wird.

Der gründlichste Weg diese Schritte zu automatisieren ist eine Tinderbox zu installieren. Diese verwaltet `Jails`, in denen Sie alle oben genannten Schritte durchführen können, ohne den Zustand Ihres laufenden Systems zu verändern. Mehr Informationen hierzu entält ports/ports-mgmt/tinderbox

3.5. Ihren Port mit `portlint` überprüfen

Bitte verwenden Sie `portlint`, um festzustellen, ob Ihr Port unseren Richtlinien entspricht. Das Programm ports-mgmt/portlint ist Teil der Ports-Sammlung. Stellen Sie vor allem sicher, dass das Makefile in der richtigen Form und das Paket passend benannt ist.

3.6. Den neuen Port einreichen

Bevor Sie den neuen Port einreichen, lesen Sie bitte unbedingt den Abschnitt DOs and DON'Ts.

Nun, da Sie mit Ihrem Port zufrieden sind, müssen Sie ihn nur noch in den Haupt-Ports-Tree von FreeBSD einbringen, damit alle daran teilhaben können. Wir benötigen nicht Ihr work-Verzeichnis oder Ihr pkgname.tgz-Paket - diese können Sie nun löschen. Wenn Ihr Port beispielsweise `oneko` heißt, wechseln Sie in das Verzeichnis, in dem sich das Verzeichnis `oneko` befindet und führen den Befehl `shar find oneko > oneko.shar` aus.

Fügen Sie Ihre Datei `oneko.shar` einem Fehlerbericht an und senden Sie diesen mit Hilfe des Programms send-pr(1) (unter Bug Reports and General Commentary finden Sie weitere Informationen über send-pr(1)). Ordnen Sie den Fehlerbericht bitte in die Kategorie `Ports` mit der Klasse `Change-Request` ein (Markieren Sie den Bericht nicht als `vertraulich` (`confidential`)!). Fügen Sie bitte eine kurze Beschreibung des Programms, das Sie portiert haben, in das "Beschreibungs"-Feld des Problemberichts und die shar-Datei in das "Fix"-Feld ein (bespielsweise eine kurze Version des `COMMENT`).



Sie können uns die Arbeit um einiges vereinfachen, wenn Sie eine gute Beschreibung in der Zusammenfassung des Problemberichtes verwenden. Wir bevorzugen etwas wie "Neuer Port: <Kategorie>/<Portname><Kurzbeschreibung des Ports>" für neue Ports. Wenn Sie sich an dieses Schema halten, ist die Chance, dass sich jemand bald Ihren Bericht ansieht, deutlich besser.

Noch einmal: *Bitte fügen Sie nicht das distfile der Originalquelle, das work-Verzeichnis oder das Paket, das Sie mit* `make package` *erstellt haben, ein.* Und verwenden Sie shar(1) für neue Ports (und NICHT diff(1)).

Haben Sie bitte etwas Geduld, nachdem Sie den Port eingereicht haben. Manchmal kann es einige Monate dauern, bevor ein Port in FreeBSD eingefügt wird, obwohl es wahrscheinlich nur ein paar Tage dauert. Sie können sich die Liste der PRs, die darauf warten, in FreeBSD committet zu werden, ansehen.

Nachdem wir einen Blick auf Ihren Port geworfen haben, werden wir, wenn nötig, bei Ihnen nachfragen und ihn in die Ports-Sammlung übernehmen. Ihr Name taucht dann auch in der Liste der Additional FreeBSD Contributors und in anderen Dateien auf. Ist das nicht toll?! :-)

Kapitel 4. Einen Port in aller Ruhe erstellen

Ok, das war nicht ganz einfach und der Port hat einige Veränderungen erfordert, um funktionieren zu können. In diesem Abschnitt werden wir Schritt für Schritt erklären, wie man den funktionierenden Port den Vorgaben der Ports entsprechend anpasst.

4.1. Die Funktionsweise

Beginnen wir mit der Abfolge der Ereignisse, die eintreten, wenn der Nutzer das erste `make` in Ihrem Portsverzeichnis ausführt. Sie empfinden es für das Verständnis vielleicht hilfreich bsd.port.mk in einem anderen Fenster offen zu haben, während Sie diesen Abschnitt lesen.

Aber machen Sie sich keine Sorgen, falls Sie nicht wirklich verstehen, was bsd.port.mk macht, die Wenigsten begreifen dies... :>

1. Das Target `fetch` wird aufgerufen. Es ist dafür verantwortlich sicherzustellen, dass der Tarball lokal im `DISTDIR` verfügbar ist. Falls `fetch` die benötigten Dateien in `DISTDIR` nicht finden kann, durchsucht es die URL `MASTER_SITES`, welche im Makefile gesetzt ist, ebenso wie unsere Haupt-FTP-Seite unter ftp://ftp.freebsd.org/pub/FreeBSD/ports/distfiles/ , wo wir genehmigte Distfiles als Backup aufbewahren. Danach wird versucht, so eine direkte Internetverbindung besteht, dass genannte Distfile mit `FETCH` herunterzuladen. Falls dies gelingt, wird die Datei in `DISTDIR` für weitere Nutzung abgelegt und fährt fort.
2. Das Target `extract` wird aufgerufen. Es sucht nach den Distfiles Ihres Ports (normalerweise ein gzip-komprimierter Tarball) in `DISTDIR` und entpackt diese in ein temporäres Unterverzeichnis, welches von `WRKDIR` festgelegt wird (standardmäßig work).
3. Das Target `patch` wird aufgerufen. Zuerst werden alle in `PATCHFILES` festgelegten Patches eingespielt. Anschließend werden, falls Patches der Form patch-* in `PATCHDIR` (standardmäßig das files-Unterverzeichnis) gefunden werden, diese in alphabetischer Reihenfolge eingespielt.
4. Das Target `configure` wird aufgerufen. Dieses kann viele verschiedene Dinge machen.
   a. Existiert scripts/configure, so wird es aufgerufen.
   b. Falls `HAS_CONFIGURE` oder `GNU_CONFIGURE` gesetzt sind, wird WRKSRC/configure ausgeführt.
   c. Falls `USE_IMAKE` gesetzt ist, wird `XMKMF` (standardmäßig `xmkmf -a`) ausgeführt.
5. Das Target `build` wird aufgerufen. Es ist für das Wechseln in das private Arbeitsverzeichnis (`WRKSRC`) und das Bauen des Ports zuständig. Ist `USE_GMAKE` gesetzt, so wird GNU `make` verwendet, sonst das System-`make`.

Die oben genannten Schritte sind die Standardaktionen. Zusätzlich können Sie `pre-_ irgendwas_` oder `post-irgendwas` als Targets definieren oder Skripten mit diesen Namen in das scripts -Unterverzeichnis legen. Sie werden dann vor bzw. nach den Standardaktionen aufgerufen.

Angenommen Sie haben das Target `post-extract` in Ihrem Makefile definiert und eine Datei pre-build im scripts Unterverzeichnis, so wird das Target `post-extract` nach dem normalen Entpacken

aufgerufen und das Skript pre-build ausgeführt, bevor die vordefinierten Bau-Regeln abgearbeitet sind. Es wird empfohlen, dass Sie Makefile-Targets verwenden, falls die Aktionen es erlauben, da es so für jemanden einfacher sein wird herauszufinden, was für eine nicht-standardmäßige Aktion der Port benötigt.

Die Standardaktionen werden aus den Targets bsd.port.mk `do-irgendwas` übernommen. Zum Beispiel sind die Befehle zum Entpacken eines Ports im Target `do-extract` zu finden. Falls Sie mit einem vorgegebenen Target nicht zufrieden sind, können Sie es verändern, indem Sie das Target `do-irgendwas` in Ihrem Makefile neu definieren.



Die "Haupt"-Targets (z.B. `extract`, `configure` usw.) machen nicht mehr als sicherzustellen, dass bis hierhin alle Abschnitte abgeschlossen sind, um danach die eigentlichen Targets oder Skripte aufzurufen. Und es ist nicht beabsichtigt, dass diese geändert werden. Falls Sie das Entpacken verändern wollen, verändern Sie `do-extract`, aber niemals die Art, wie `extract` arbeitet!

Jetzt, da Sie verstehen, was geschieht, wenn der Benutzer `make` eingibt, lassen Sie uns durch die empfohlenen Schritte gehen, um den perfekten Port zu erstellen.

4.2. Den originalen Quelltext besorgen

Normalerweise liegt der original Quelltext als gepackte Datei (foo.tar.gz oder foo.tar.Z) vor. Kopieren Sie diese nach `DISTDIR`. Nutzen Sie, soweit möglich, immer die Quellen aus dem *Hauptzweig*.

Es ist notwendig die Variable `MASTER_SITES` anzupassen, um anzugeben, wo sich der originale Quelltext befindet. In bsd.sites.mk finden sich hilfreiche Definitionen für die gebräuchlichsten Seiten. Bitte nutzen Sie diese Seiten und die zugehörigen Definitionen, soweit dies möglich ist. Damit wird vermieden, immer und immer wieder dieselben Informationen zu wiederholen. Da die Hauptseiten regelmäßig angepasst werden müssen, vereinfacht dieses Vorgehen die Pflege der Dateien für jeden Beteiligten.

Falls keine zuverlässige und gut erreichbare FTP/HTTP-Seite zu finden ist, oder nur Seiten auffindbar sind, die keinen Standards entsprechen, sollte eine Kopie des Quelltextes auf einer zuverlässigen Seite abgelegt werden. Dies könnte z.B. die eigene Internetseite sein.

Ist kein geeigneter Ort zum Ablegen des Quelltextes auffindbar, ist es möglich diesen "intern" auf `ftp.FreeBSD.org` abzulegen; dies sollte jedoch als letzte Möglichkeit angesehen werden. Das Distfile muss in diesem Fall in ~/public_distfiles/ eines `freefall`-Accounts abgelegt werden. Bitten Sie den Committer Ihres Ports dies zu erledigen. Er wird außerdem `MASTER_SITES` nach `MASTER_SITE_LOCAL` und `MASTER_SITE_SUBDIR` auf den `freefall`-Benutzernamen angepasst.

Sollte sich das Distfile des Ports regelmäßig ohne Versionsanpassungen des Autors ändern, sollte überlegt werden, das Disfile auf der eigenen Internetseite abzulegen und diese in der Liste der `MASTER_SITES` an die erste Stelle zu setzen. Falls möglich, sollte der Autor des Ports gebeten werden, dies zu erledigen; hierüber wird die Kontrolle des Quelltextes verbessert. Wird eine eigene Version des Quelltextes auf eigenen Internetseiten verfügbar gemacht, verhindert dies Warnungen von `checksum mismatch` und reduziert den Arbeitsaufwand der Maintainer der FTP-Seiten. Auch wenn

nur eine Quelle für den Quelltext des Ports zur Verfügung steht, ist es empfohlen, ein Backup auf einer weiteren Seite abzulegen und diese als zweiten Eintrag in MASTER_SITES aufzunehmen.

Sind für den Port zusätzlich aus dem Internet verfügbare Patches erforderlich, sollten diese ebenfalls in DISTDIR abgelegt werden. Sollten diese Patches von anderer Quelle als der Hauptseite des Ports stammen, ist das kein Grund zur Sorge. Es gibt Wege diesem Umstand gerecht zu werden (beachten Sie die unten stehende Beschreibung zu PATCHFILES).

4.3. Den Port bearbeiten

Entpacken Sie eine Kopie des Tarballs in ein privates Verzeichnis und nehmen Sie alle Änderungen vor, die nötig sind, um den Port unter einer aktuellen FreeBSD-Version kompilieren zu können. *Protokollieren Sie sorgfältig* alle Schritte, die Sie vornehmen, da Sie den Prozess in Kürze automatisieren werden. Alles, auch das Entfernen, Hinzufügen oder Bearbeiten von Dateien, sollte von einem automatisierten Skript oder einer Patch-Datei machbar sein, wenn Ihr Port fertig ist.

Falls Ihr Port bedeutende Interaktionen/Veränderungen durch den Benutzer benötigt, um ihn zu Kompilieren oder zu Installieren, sollten Sie einen Blick auf Larry Walls klassische Configure-Skripte werfen oder vielleicht etwas Ähnliches selbst erstellen. Das Ziel der Ports-Sammlung ist es, jeden Port so "plug-and-play-fähig" wie möglich für den Endbenutzer zu machen, während ein Minimum an Speicherplatz gebraucht wird.



Solange nicht anders angegeben wird von Patch-Dateien, Skripten und anderen Dateien, die Sie erstellt und der FreeBSD Ports-Sammlung hinzugefügt haben, angenommen, dass Sie unter den standardmäßigen BSD-Copyright-Bedingungen stehen.

4.4. Fehlerbehebung (Patches)

Bei der Vorbereitung eines Ports können die Dateien, die hinzugefügt oder verändert wurden, mittels diff(1) abgefangen werden, um Sie später an patch(1) zu übergeben. Jeder Patch, der dem Quelltext übergeben werden soll, sollte in einer Datei patch-* abgelegt werden, wobei * dem Pfadnamen der zu korrigierenden Datei entspricht, wie er auch in patch-Imakefile oder im patch-src-config.h erscheint. Diese Dateien sollten in PATCHDIR (normalerweise files) abgelegt sein, von wo sie automatisch übernommen werden. Alle Patches müssen sich relativ zur WRKSRC-Variable (normalerweise dem Verzeichnis, in dem sich der Quelltext des Ports entpackt und wo auch der Bau stattfindet) befinden.

Um Korrekturen und Updates zu vereinfachen, sollte es vermieden werden, mehr als einen Patch für eine Datei zu nutzen (z.B. patch-file und patch-file2, welche beide WRKSRC/foobar.c verändern). Beachten Sie, dass, falls der Pfad einer zu korrigierenden Datei einen Unterstrich (_) enthält, der Patch stattdessen zwei Unterstriche im Namen haben muss. Zum Beispiel muss der Patch, der eine Datei namens src/freeglut_joystick.c korrigieren soll, patch-src-freeglut__joystick.c genannt werden.

Für die Benennung der Patches sollten nur die Zeichen [-+._a-zA-Z0-9] genutzt werden. Bitte verwenden Sie keine weiteren Zeichen als die angegebenen. Die Namensvergabe sollte nicht patch-aa oder patch-ab etc. entsprechen, erwähnen Sie immer den Pfad und Dateinamen.

RCS-Zeichenketten sollten vermieden werden, da CVS diese verstümmeln würde, sobald wir diese Dateien in die Ports-Sammlung einpflegen. Wenn wir die Dateien wieder abrufen wären diese verändert und der Patch würde fehlschlagen. RCS-Zeichenketten sind in Dollar-Zeichen (`$`) eingefügte Zeichen und beginnen üblicherweise mit `$Id` oder `$RCS`.

Die Option rekursiv (`-r`) zu nutzen diff(1), um Patches zu erstellen, ist zulässig, jedoch sollte der Patch anschließend geprüft werden, um Unnötiges aus dem Patch zu entfernen. Im Einzelnen bedeutet dies, dass Diffs zwischen zwei Backup-Dateien, Makefiles oder wenn der Port `Imake` oder GNU `configure` usw. nutzt, überflüssig sind und entfernt werden sollten. Falls es es notwendig war, configure.in zu bearbeiten und es soll `autoconf` zum Neuerstellen von `configure` genutzt werden, sollten die Diffs aus `configure` nicht genutzt werden (diese werden oft einige tausend Zeilen groß!); - hier sollte `USE_AUTOTOOLS=autoconf:261` definiert und das Diff aus configure.in genutzt werden.

Zusätzlich sollte man unnötige Markup-Änderungen in Patches/Änderungen möglichst vermeiden. In der Open Source-Welt teilen sich Projekte häufig große Teile des Quellcodes. Allerdings verwenden die einzelnen Projekte oft unterschiedliche Programmierstile und Vorgaben für Einrückungen. Wenn man also einen funktionierenden Teil einer Funktion aus einem Projekt verwendet, um ein ähnliches Problem in einem anderen Projekt zu lösen, sollte man besonders vorsichtig sein, weil sich ansonsten die CVS-Änderungseinträge mit überflüssigen Einträgen füllen, die nur das Markup des Quellcodes betreffen, ohne dass sich an der Funktion des eigentlichen Quellcode etwas ändert ("withspace-only changes"). Solche Änderungen vergrößern nicht nur das CVS-Repository, sondern erschweren es auch die Ursache für eventuell auftretende Probleme zu finden.

War es notwendig eine Datei zu entfernen, wird dies besser mittels des `post-extract`-Targets als über den Patch selbst realisiert.

Ein einfacher Austausch kann direkt über das Makefile des Ports umgesetzt werden, indem der in-place-Modus von sed(1) genutzt wird. Dies ist sehr hilfreich, wenn variable Werte korrigiert werden sollen. Beispiel:

```
post-patch:
      @${REINPLACE_CMD} -e 's|for Linux|for FreeBSD|g' ${WRKSRC}/README
      @${REINPLACE_CMD} -e 's|-pthread|${PTHREAD_LIBS}|' ${WRKSRC}/configure
```

Relativ häufig ergibt sich die Situation, in der die portierte Software die CR/LF-Konventionen für Zeilenenden nutzt (dies ist bei unter Windows® entwickelter Software häufig der Fall). Dies kann bei weiteren Patches Probleme (Compiler-Warnungen, Fehlermeldungen bei der Ausführung von Skripten wie z.B. `/bin/sh^M` not found) und anderes ergeben. Um schnell alle Dateien von CR/LF nach LF zu konvertieren, kann `USE_DOS2UNIX=yes` in das Makefile des Ports geschrieben werden. Hierzu kann eine Liste der zu konvertierenden Dateien erstellt werden:

```
USE_DOS2UNIX=    util.c util.h
```

Sollen Gruppen von Dateien über verschiedene Unterverzeichnisse konvertiert werden, kann `DOS2UNIX_REGEX` genutzt werden, dessen Argumente `find`-kompatible, reguläre Ausdrücke sind. Mehr zur Formatierung findet sich in re_format(7). Diese Option ist beim Konvertieren aller Dateien mit

definierter Endung, z.B. aller Dateien im Quellcode, wobei binäre Dateien unberührt bleiben, sinnvoll:

```
USE_DOS2UNIX=    yes
      DOS2UNIX_REGEX=  .*\.(c|cpp|h)
```

Wenn Sie einen Patch zu einer bereits existierenden Datei erstellen wollen, können Sie von ihr eine Kopie mit der Endung .orig erstellen und anschließend die Originaldatei bearbeiten. Das make-Ziel `makepatch` führt dann zu einer entsprechenden Patch-Datei im Verzeichnis files des Ports.

4.5. Konfigurieren

Fügen Sie alle zusätzlichen Veränderungsbefehle Ihrem Skript configure hinzu und speichern Sie es im scripts-Unterverzeichnis. Wie vorstehend schon erwähnt, können Sie dies auch mit den Targets Makefile und/oder Skripte mit dem Namen pre-configure oder post-configure erledigen.

4.6. Handhabung von Benutzereingaben

Sollte der Port Eingaben vom Benutzer benötigen, muss `IS_INTERACTIVE` im Makefile des Ports gesetzt werden. Dies erlaubt "overnight builds" Ihren Port zu überspringen, falls der Nutzer die Variable `BATCH` setzt (setzt der Nutzer hingegen die Variable `INTERACTIVE`, werden *nur* Ports gebaut, die Interaktion vom Nutzer erwarten). Dies erspart den Rechnern, welche kontinuierlich Ports bauen, eine Menge Zeit (siehe unten).

Zudem ist es empfohlen, falls sinnvolle Vorgaben für interaktive Optionen gesetzt sind, die `PACKAGE_BUILDING`-Variable zu prüfen und das interaktive Skript abzuschalten. Dies macht es uns möglich, Pakete für CDROMs und FTP-Server zu bauen.

Kapitel 5. Die Konfiguration des Makefile

Das Konfigurieren des Makefile ist sehr einfach und wir schlagen vor, dass Sie zunächst einen Blick auf vorhandene Beispiele werfen. Zusätzlich gibt es ein Beispiel eines Makefile in diesem Handbuch. Schauen Sie es sich an und verfolgen Sie bitte die Abfolge der Variablen und Abschnitte in dieser Vorlage. Damit erleichtern Sie es anderen, Ihren Port zu lesen.

Bedenken Sie bitte die folgenden Probleme in der hier vorgegebenen Abfolge der Unterabschnitte dieses Kapitels, wenn Sie Ihr neues Makefile erstellen:

5.1. Der originale Quelltext

Liegt der Quelltext in DISTDIR als eine standardisierte und mit gzip gepackte Datei in der Art foozolix-1.2.tar.gz? Falls ja, können Sie zum nächsten Schritt übergehen. Falls nicht, sollten Sie versuchen, die Variablen DISTVERSION, DISTNAME, EXTRACT_CMD, EXTRACT_BEFORE_ARGS, EXTRACT_AFTER_ARGS, EXTRACT_SUFX, oder DISTFILES zu ändern. Das hängt davon ab, wie fremdartig das Distributionsfile Ihres Ports ist (der häufigste Fall ist EXTRACT_SUFX=.tar.Z, wenn der Tarball durch ein normales compress und nicht durch gzip gepackt wurde).

Im schlimmsten Fall können Sie einfach Ihre eigene Vorgabe mittels do-extract erzeugen und die Standardvorgabe überschreiben; aber dies sollte in den wenigsten Fällen, wenn überhaupt, notwendig sein.

5.2. Bezeichnungen

Der erste Teil des Makefile beschreibt die Versionsnummer des Ports und führt ihn in der richtigen Kategorie auf.

5.2.1. PORTNAME und PORTVERSION

Setzen Sie bitte die Variable PORTNAME auf den Basisnamen Ihres Ports und die Variable PORTVERSION auf dessen Versionsnummer.

5.2.2. PORTREVISION und PORTEPOCH

PORTREVISION

Die PORTREVISION-Variable ist ein streng monoton wachsender Wert, welcher auf 0 zurückgesetzt wird, nachdem PORTVERSION erhöht wurde (d.h. jedes Mal, wenn ein offizielles Release erfolgt). Sie wird an den Namen des Pakets angehängt, wenn sie ungleich 0 ist. Änderungen an PORTREVISION werden von automatisierten Werkzeugen (z.B. pkg_version(1)) genutzt, um anzuzeigen, dass ein neues Paket verfügbar ist.

PORTREVISION sollte jedes Mal erhöht werden, wenn eine Änderung am Port erfolgt, die beträchtliche Auswirkungen auf den Inhalt oder Struktur des aus dem Port erzeugten Pakets zur Folge hat.

Beispiele dafür, wann PORTREVISION erhöht werden sollte:

- Hinzufügen von Patches, welche Sicherheitslücken schließen, Fehler beseitigen oder neue Funktionalität zum Port hinzufügen.
- Änderungen am Makefile des Ports, welche compile-time-Optionen hinzufügen oder entfernen.
- Änderungen bezüglich Packliste oder am Verhalten während der Installation des Pakets (d.h. Änderungen an einem Skript, welches Ausgangsdaten für das Paket erzeugt, wie z.B. SSH-Hostschlüssel).
- Versionssprung einer Shared-Library, welche eine Abhängigkeit dieses Ports ist (In diesem Fall würde ein Anwender bei der Installation des alten Pakets scheitern, falls er eine neue Version der Abhängigkeit bereits installiert hat, weil nach der alten Bibliothek libfoo.x anstatt nach libfoo.(x+1)) gesucht wird).
- Schleichende Änderungen am Distfile, welche bedeutende funktionale Änderungen verursachen, d.h. Änderungen des Distfile erfordern eine Korrektur an distinfo, ohne dass damit zusammenhängend die `PORTVERSION` verändert wird, obwohl ein `diff -ru` zwischen der alten und der neuen Version bedeutende Veränderungen am Code nachweist.

Beispiele für Änderungen, welche keine Erhöhung von `PORTREVISION` erfordern:

- Stilistische Änderungen am Grundgerüst des Ports ohne funktionale Änderungen am daraus resultierenden Paket.
- Änderungen an der Variable `MASTER_SITES` oder andere funktionale Änderungen, welche das resultierende Paket nicht verändern.
- Marginale Patches am Distfile wie die Korrektur von Tippfehlern, welche nicht wichtig genug sind, um dem Benutzer die Bürde eines Upgrades aufzuerlegen.
- Build fixes, die ein Paket erst kompilierbar machen, welches ohne diese Änderungen vorher nicht erzeugt werden konnte (solange die Änderungen keine funktionale Differenz bringen auf Plattformen, auf denen dieses Paket schon vorher gebaut werden konnte). Da `PORTREVISION` den Inhalt des Pakets wiederspiegelt, ist es nicht notwendig `PORTREVISION` zu erhöhen, wenn das Paket vorher nicht erstellt werden konnte.

Als Faustregel gilt: Stellen Sie sich die Frage, ob die durchgeführte Änderung am Port jedem hilft (entweder aufgrund einer Verbesserung, Beseitigung eines Fehlers, oder der Annahme, dass das neue Paket überhaupt erst funktioniert) und wägen Sie es gegen den Umstand ab, dass jedermann, der seine Ports-Sammlung regelmässig auf dem neuesten Stand hält, zu einer Aktualisierung gezwungen wird. Falls Sie die Frage positiv beantworten sollten, erhöhen Sie die Variable `PORTREVISION`.

PORTEPOCH

Von Zeit zu Zeit geschieht es, dass irgendjemand (Drittanbieter von Software oder FreeBSD Ports Committer) etwas Dummes tut und eine Version einer Software veröffentlicht, deren Versionsnummer niedriger ist als die der vorherigen. Ein Beispiel hierfür ist ein Port, der von foo-20000801 auf foo-1.0 geändert wird (der Erstere wird fälschlicherweise als neue Version behandelt, weil 2000801 ein numerisch größerer Wert ist als 1).

In Situationen wie diesen sollte die Variable `PORTEPOCH` erhöht werden. Wenn `PORTEPOCH` größer als 0 ist, wird sie an den Namen des Pakets angehängt, wie in Abschnitt 0 oberhalb bereits beschrieben.

PORTEPOCH darf niemals verringert oder auf 0 gesetzt werden, weil der Vergleich des Pakets mit einem früheren Zeitpunkt scheitern würde (d.h. das Paket würde niemals als veraltet erkannt werden): Die neue Versionsnummer (1.0,1 im obigen Beispiel) ist immer noch numerisch kleiner als die vorherige Version (2000801), aber das Suffix ,1 wird von automatisierten Werkzeugen gesondert behandelt und wird als größer erkannt, als das implizit angenommene Suffix ,0 im früheren Paket.

Das Entfernen oder Zurücksetzen von PORTEPOCH führt zu unendlichem Ärger. Wenn Sie die obigen Ausführungen nicht vollständig verstanden haben, lesen Sie es bitte unbedingt nochmals bis Sie es vollständig verinnerlicht haben, oder fragen Sie vor jeder Änderung auf den Mailinglisten nach!

Es wird erwartet, dass PORTEPOCH für die weitaus überwiegende Zahl der Ports nicht verwendet wird und der verantwortungsvolle und vorausschauende Umgang mit PORTVERSION macht es meist überflüssig, falls ein späteres Release die Versionsstruktur ändern sollte. Vorsicht ist geboten, wenn ein Release einer Drittanbieter-Software ohne eine offizielle Versionsnummer veröffentlicht wird, wie z.B. bei "Snapshot-Versionen". Man ist versucht, das Release mit dem jeweiligen Datum zu bezeichnen, was unweigerlich zu den oben beschriebenen Problemen führt, wenn das nächste "offizielle" Release erscheint.

Wenn z.B. ein Snapshot zum Datum 20000917 veröffentlicht wird und die vorherige Version der Software war 1.2, dann sollte der Snapshot die PORTVERSION 1.2.20000917 oder ähnlich erhalten und nicht 20000917, damit das nachfolgende Release, angenommen 1.3, immer noch einen größeren numerischen Wert aufweist.

Beispiel für den Gebrauch von PORTREVISION und PORTEPOCH

Der gtkmumble-Port, Version 0.10, befindet sich in der Ports-Sammlung:

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

PKGNAME wird zu gtkmumble-0.10.

Ein Sicherheitsloch wurde entdeckt, das einen lokalen Patch von FreeBSD erforderlich macht. PORTREVISION wird entsprechend erhöht.

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

PKGNAME wird zu gtkmumble-0.10_1

Eine neue Version wird vom Software-Drittanbieter veröffentlicht, bezeichnet mit der Version 0.2 (es stellt sich heraus, dass der Autor beabsichtigte, dass 0.10 eigentlich 0.1.0 bedeuten sollte, nicht "was kommt nach 0.9" - Hoppla, aber nun ist es zu spät). Da die neue Unterversion 2 numerisch kleiner ist als die vorherige Version 10, muss PORTEPOCH erhöht werden, um sicherzustellen, dass das neue Paket auch als "neuer" erkannt wird. Da es ein neues Release des Drittanbieters ist, wird PORTREVISION auf 0 zurückgesetzt (oder aus dem Makefile entfernt).

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

PKGNAME wird zu gtkmumble-0.2,1

Das nächste Release ist 0.3. Da PORTEPOCH niemals verringert wird, sind die Versionsvariablen nun wie folgt:

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

PKGNAME wird zu gtkmumble-0.3,1



Falls PORTEPOCH mit diesem Upgrade auf 0 zurückgesetzt worden wäre, dann würde jemand, der das Paket gtkmumble-0.10_1 installiert hätte, das Paket gtkmumble-0.3 nicht als neuer erkennen, da 3 immer noch numerisch kleiner ist als 10. Bedenken Sie, dass genau dies der springende Punkt an PORTEPOCH ist.

5.2.3. PKGNAMEPREFIX und PKGNAMESUFFIX

Zwei optionale Variablen, PKGNAMEPREFIX und PKGNAMESUFFIX, werden verknüpft mit PORTNAME und PORTVERSION, um PKGNAME zu bilden als ${PKGNAMEPREFIX}${PORTNAME}${PKGNAMESUFFIX}-${PORTVERSION}. Stellen Sie bitte unbedingt sicher, dass diese Variablen den Richtlinien für einen guten Paketnamen entsprechen. Insbesondere dürfen Sie *keinesfalls* einen Bindestrich (-) in PORTVERSION verwenden. Falls das Paket den *language-* oder *-compiled.specifics*-Teil aufweist (siehe unten) benutzen Sie PKGNAMEPREFIX oder PKGNAMESUFFIX respektive. Machen Sie diese Variablen nicht zum Bestandteil von PORTNAME!

5.2.4. LATEST_LINK

Die Umgebungsvariable LATEST_LINK wird während der Paketerstellung verwendet, um einen Kurznamen festzulegen, der danach von pkg_add -r genutzt werden kann. Dadurch wird es beispielsweise möglich, die aktuelle Perl-Version durch einen einfachen Aufruf von pkg_add -r perl zu installieren (ohne die Angabe der korrekten Versionsnummer). Dieser Name muss eindeutig sowie "offensichtlich" sein.

In einigen Fällen können mehrere Versionen einer Applikation gleichzeitig in der Ports-Sammlung sein. Das index build- und das package build-System müssen nun in der Lage sein, diese als unterschiedliche Ports zu erkennen, obwohl diese Versionen alle die gleichen Variablen PORTNAME, PKGNAMEPREFIX und sogar PKGNAMESUFFIX aufweisen. In solchen Fällen sollte die optionale Variable LATEST_LINK auf einen unterschiedlichen Wert für alle Ports gesetzt werden mit Ausnahme des "Haupt-Ports". Beispiele hierfür sind die lang/gcc46 und lang/gcc-Ports und die www/apache* -Familie. Wenn Sie die Umgebungsvariable NO_LATEST_LINK setzen, wird kein Link erzeugt, was für alle Versionen (aber nicht für die "Hauptversion") nützlich sein kann. Beachten Sie bitte, dass die

Frage der Auswahl der "wichtigsten" Version ("am populärsten", "am besten Unterstützt", "zuletzt gepatcht" usw.) ausserhalb der Möglichkeiten dieses Handbuches liegt. Wir sagen Ihnen nur, wie Sie die anderen Ports spezifizieren, nachdem Sie den "Haupt-Port" erkoren haben.

5.2.5. Namensregeln für Pakete

Im Folgenden finden Sie die Regeln für die Benennung Ihrer Pakete. Diese sollen gewährleisten, dass das Paketverzeichnis leicht zu durchsuchen ist, da es bereits abertausende Pakete gibt und die Nutzer sich mit Schauder abwenden, wenn Ihre Augen überstrapaziert werden!

Der Paketname soll aussehen wie language_region-name-compiled.specifics-version.numbers.

Der Paketname ist definiert als `${PKGNAMEPREFIX}${PORTNAME}${PKGNAMESUFFIX}-${PORTVERSION}`. Stellen Sie bitte sicher, dass die Variablen Ihres Ports diesem Format entsprechen.

1. FreeBSD bemüht sich ausserordentlich, die Landessprachen seiner Nutzer zu unterstützen. Die _language-_Variable soll eine Abkürzung mit 2 Buchstaben sein der Sprachen gemäß ISO-639, falls der Port für eine bestimmte Sprache spezifisch ist. Beispiele hierfür sind `ja` für Japanisch, `ru` für Russisch, `vi` für Vietnamesisch, `zh` für Chinesisch, `ko` für Koreanisch und `de` für Deutsch.

   Sollte der Port spezifisch sein für eine gewisse Region innerhalb eines Sprachraumes, dann fügen Sie bitte auch den Ländercode mit 2 Buchstaben hinzu. Beispiele sind `en_US` für nordamerikanisches Englisch und `fr_CH` für schweizerisches Französisch.

   Der *language*-Teil muss in der `PKGNAMEPREFIX`-Variable gesetzt werden.

2. Der erste Buchstabe des name-Teils muss kleingeschrieben werden (der Rest des Namens kann Großbuchstaben enthalten. Daher seien Sie bitte umsichtig, wenn Sie den Namen einer Software konvertieren, welche Grossbuchstaben enthält). Es ist Tradition, `Perl 5`-Module durch ein vorstehendes `p5-` und durch Umwandlung des doppelten Doppelpunktes in Bindestriche zu bezeichnen. So wird z.B. aus dem `Data::Dumper`-Modul der `p5-Data-Dumper`-Port.
3. Vergewissern Sie sich, dass der Name des Ports und seine Versionsnummer klar getrennt sind und in den Variablen `PORTNAME` und `PORTVERSION` stehen. Der einzige Grund, um in `PORTNAME` einen Versionsteil aufzunehmen ist der, dass die Software wirklich so bezeichnet wird, wie z.B. die Ports textproc/libxml2 oder japanese/kinput2-freewnn. Ansonsten sollte `PORTNAME` keine versionsspezifischen Bestandteile aufweisen. Es ist vollkommen normal, dass viele Ports den gleichen `PORTNAME` aufweisen wie z.B. die www/apache*-Ports. In diesem Falle werden unterschiedliche Versionen (und unterschiedliche Indexeinträge) unterschieden durch die Werte von `PKGNAMEPREFIX`, `PKGNAMESUFFIX` und `LATEST_LINK`.
4. Falls der Port mit verschiedenen, fest kodierten Vorgaben (üblicherweise Teil des Verzeichnisnamens in einer Familie von Ports) gebaut werden kann, dann soll der *-compiled.specifics*-Teil die einkompilierten Vorgaben anzeigen (der Bindestrich ist optional). Beispiele hierfür sind Papiergrößen und Font-Einheiten.

   Der *-compiled.specifics*-Teil muss in der Variablen `PKGNAMESUFFIX` gesetzt werden.

5. Die Versionszeichenfolge sollte einen Bindestrich (`-`) am Schluss haben und eine von Punkten getrennte Liste von Integer-Zahlen und kleingeschriebenen Buchstaben sein. Es ist nicht zulässig, einen weiteren Bindestrich innerhalb des Versionsstrings zu verwenden! Die einzige

Ausnahme hiervon ist die Zeichenfolge `pl` (bedeutet "patchlevel"), welche *nur* dann gebraucht werden darf, wenn die Applikation über keine Haupt- oder Unterversionsnummern verfügt. Wenn die Versionsbezeichnung der Software Zeichenketten wie "alpha", "beta", "rc" oder "pre" enthält, dann nehmen Sie bitte den ersten Buchstaben daraus und setzen ihn unmittelbar hinter einen Punkt. Falls die Versionszeichenfolge nach diesem Punkt fortgesetzt wird, sollen die Zahlen ohne einen Punkt zwischen den einzelnen Buchstaben folgen.

Das Ziel ist es, die Ports anhand der Versionszeichenfolge zu sortieren. Stellen Sie bitte unbedingt sicher, dass die Bestandteile der Versionsnummer immer durch einen Punkt getrennt sind und falls Datumsangaben verwendet werden, dass diese im Format `0.0.yyyy.mm.dd` und nicht `dd.mm.yyyy` oder gar dem nicht Y2K-kompatiblen Format `yy.mm.dd` vorliegen. Es ist wichtig, dass die Versionsnummer mit `0.0.` beginnt, da die Versionsnummer im Falle einer Veröffentlichung auf jeden Fall kleiner als `yyyy` sein wird.

Hier sind einige reale Beispiele, die aufzeigen, wie man den Namen einer Applikation zu einem vernünftigen Paketnamen umwandelt:

Softwarename	**PKGNAMEPREFIX**	**PORTNAME**	**PKGNAMESUFFIX**	**PORTVERSION**	**Grund**
mule-2.2.2	(leer)	mule	(leer)	2.2.2	Keine Änderung erforderlich
EmiClock-1.0.2	(leer)	emiclock	(leer)	1.0.2	keine Großbuchstaben für einzelne Applikationen
rdist-1.3alpha	(leer)	rdist	(leer)	1.3.a	Keine Zeichenketten wie `alpha` erlaubt
es-0.9-beta1	(leer)	es	(leer)	0.9.b1	keine Zeichenketten wie `beta` erlaubt
mailman-2.0rc3	(leer)	mailman	(leer)	2.0.r3	keine Zeichenketten wie `rc` erlaubt
v3.3beta021.src	(leer)	tiff	(leer)	3.3	Was sollte denn das eigentlich sein?
tvtwm	(leer)	tvtwm	(leer)	pl11	Versionsstring zwingend erforderlich

Softwarename	PKGNAMEPREFIX	PORTNAME	PKGNAMESUFFIX	PORTVERSION	Grund
piewm	(leer)	piewm	(leer)	1.0	Versionsstring zwingend erforderlich
xvgr-2.10pl1	(leer)	xvgr	(leer)	2.10.1	`pl` nur erlaubt, wenn keine Versionsnummer vorhanden
gawk-2.15.6	ja-	gawk	(leer)	2.15.6	Japanische Sprachversion
psutils-1.13	(leer)	psutils	-letter	1.13	Papergröße beim Paketbau fix kodiert
pkfonts	(leer)	pkfonts	300	1.0	Paket für 300 DPI Schriftarten

Falls es in der Originalquelle überhaupt keinen Anhaltspunkt für irgendeine Versionsbezeichnung gibt und es unwahrscheinlich ist, dass der Autor jemals eine neue Version veröffentlichen wird, dann setzen Sie bitte die Version einfach auf `1.0` (wie im obigen Beispiel `piewm`). Sie können auch den Autor fragen oder eine Datumszeichenfolge in der Art `0.0.yyyy.mm.dd` als Version verwenden.

5.3. Kategorisierung

5.3.1. `CATEGORIES`

Wenn ein Paket erzeugt wird, dann wird es unter /usr/ports/packages/All abgelegt und von einem oder mehreren Unterverzeichnissen werden auf /usr/ports/packages Links erstellt. Die Namen dieser Unterverzeichnisse werden durch die Variable `CATEGORIES` festgelegt. Dies geschieht, um dem Nutzer zu helfen, eine große Zahl von Paketen auf einer FTP-Webseite oder einer CD/DVD zu durchsuchen. Bitte werfen Sie einen Blick auf die Aktuelle Liste der Kategorien und suchen Sie die beste Kategorie für Ihren Port aus.

Diese Liste legt auch fest, an welcher Stelle in der Ports-Sammlung der Port eingefügt wird. Falls Sie mehrere Kategorien angeben wird angenommen, dass die Dateien des Ports im Unterverzeichnis mit dem Namen der ersten angegebenen Kategorie liegen. Schauen Sie bitte unten für weitere Informationen darüber, wie man die richtige Kategorie bestimmt.

5.3.2. Aktuelle Liste der Kategorien

Hier ist die aktuelle Liste der Kategorien. Die mit einem Asterisk (*) bezeichneten sind *virtuelle* Kategorien, also solche, welche über kein eigenes Unterverzeichnis in der Ports-Sammlung verfügen. Sie werden nur als Sekundärkategorien benutzt und sind nur für Suchzwecke eingerichtet worden.



Für nicht-virtuelle Kategorien finden Sie eine einzeilige Beschreibung in der Variable `COMMENT` im Makefile des jeweiligen Unterverzeichnisses.

Kategorie	Beschreibung	Anmerkung
accessibility	Ports für behinderte Menschen.	
afterstep*	Ports für den AfterStep Window Manager.	
arabic	Arabische Sprachunterstützung.	
archivers	Archivierungswerkzeuge.	
astro	Ports für Astronomie.	
audio	Sound-Unterstützung.	
benchmarks	Benchmarking-Werkzeuge.	
biology	Software für Biologie.	
cad	CAD-Werkzeuge.	
chinese	Chinesische Sprachunterstützung.	
comms	Kommunikationsprogramme.	Hauptsächlich Software für serielle Schnittstellen.
converters	Zeichensatz-Konverter.	
databases	Datenbanken.	
deskutils	Dinge, die vor der Erfindung des Computers auf dem Schreibtisch waren.	
devel	Entwicklungs-Werkzeuge.	Legen Sie keine Bibliotheken hier ab, nur weil es Bibliotheken sind, es sei denn, sie gehören wirklich nirgendwo anders hin.
dns	DNS-bezogene Software.	
docs*	Meta-Ports für die FreeBSD-Dokumentation.	
editors	allgemeine Editoren.	Spezielle Editoren gehören in Ihre jeweilige Kategorie, (z.B. gehört ein mathematischer Formeleditor in math).
elisp*	Emacs-lisp-Ports.	

Kategorie	Beschreibung	Anmerkung
emulators	Emulatoren für andere Betriebssysteme.	Terminal-Emulatoren gehören *nicht* hierher; X-basierende gehören zu x11 und text-basierende zu comms oder misc, abhängig von deren genauer Funktionalität.
finance	Finanz-Software und ähnliches.	
french	Französische Sprachunterstützung.	
ftp	FTP Client- und Server-Werkzeuge.	Falls Ihr Port sowohl FTP als auch HTTP unterstützt, stellen Sie ihn in ftp mit der Zweitkategorie www.
games	Spiele.	
geography*	geografische Software.	
german	Deutsche Sprachunterstützung.	
gnome*	Ports für GNOME	
gnustep*	Software für GNUstep.	
graphics	grafische Werkzeuge.	
hamradio*	Software für Amateurfunk.	
haskell*	Software für die Haskell-Programmiersprache.	
hebrew	Hebräische Sprachunterstützung.	
hungarian	Ungarische Sprachunterstützung.	
ipv6*	IPv6-bezogene Software.	
irc	Internet Relay Chat (IRC)-Werkzeuge.	
japanese	Japanische Sprachunterstützung.	
java	Software für die Java™-Programmiersprache.	Die java-Kategorie sollte nicht die Einzige für einen Port sein mit Ausnahme der direkt nur mit der Programmiersprache zusammenhängenden Applikationen. Porter sollten java nicht als Hauptkategorie eines Ports wählen.

Kategorie	Beschreibung	Anmerkung
kde*	Ports für das K Desktop Environment (KDE)-Projekt.	
kld*	Kernelmodule.	
korean	Koreanische Sprachunterstützung.	
lang	Programmiersprachen.	
linux*	Linux-Applikationen und -Werkzeuge.	
lisp*	Software für die Lisp-Programmiersprache.	
mail	Mail-Software.	
math	Numerische Berechnungen und andere mathematische Werkzeuge.	
mbone*	MBone-Applikationen.	
misc	Verschiedene Werkzeuge.	Hauptsächlich Werkzeuge, die nicht anderswo hingehören. Versuchen Sie, falls irgend möglich, eine bessere Kategorie für Ihren Port zu finden als `misc`, weil Ports hier leicht untergehen.
multimedia	Multimedia-Software.	
net	Verschiedene Netzwerk-Software.	
net-im	Instant Messaging-Software.	
net-mgmt	Netzwerk-Management-Software.	
net-p2p	Peer to peer-Netzwerkprogramme.	
news	USENET News-Software.	
palm	Software für Palm™.	
parallel*	Applikationen für paralleles Rechnen.	
pear*	Ports für das Pear PHP-Framework.	
perl5*	Ports, welche Perl Version 5 benötigen.	

Kategorie	Beschreibung	Anmerkung
plan9*	Verschiedene Programme von Plan9.	
polish	Polnische Sprachunterstützung.	
ports-mgmt	Hilfsprogramme für das Installieren und Entwickeln von FreeBSD Ports und Paketen.	
portuguese	Portugiesische Sprachunterstützung.	
print	Drucker-Software.	Desktop Veröffentlichungs-Werkzeuge (DTP, Betrachter etc.) gehören auch hierher.
python*	Software für Python.	
ruby*	Software für Ruby.	
rubygems*	Ports für RubyGems-Pakete.	
russian	Russische Sprachunterstützung.	
scheme*	Software für die Scheme-Programmiersprache.	
science	Wissenschaftliche Programme, die in keine andere Kategorie passen wie z.B. astro, biology und math.	
security	Security-Werkzeuge.	
shells	Shells.	
spanish*	Spanische Sprachunterstützung.	
sysutils	System-Werkzeuge.	
tcl*	Ports, welche Tcl benötigen.	
textproc	Textverarbeitungsprogramme.	Dies beinhaltet nicht DTP-Werkzeuge, diese gehören in print.
tk*	Ports, welche Tk benötigen.	
ukrainian	Ukrainische Sprachunterstützung.	
vietnamese	Vietnamesische Sprachunterstützung.	
windowmaker*	Ports für den WindowMaker Window-Manager.	
www	Software für das World Wide Web (WWW).	HTML-Werkzeuge gehören auch hierher.

Kategorie	Beschreibung	Anmerkung
x11	X-Window-System und dergleichen.	Diese Kategorie ist nur für Software, welche direkt X unterstützt. Fügen Sie keine normalen X-Applikationen hinzu. Die meisten davon gehören in eine andere x11-* -Kategorie (siehe unten). Falls Ihr Port eine X-Applikation *ist*, dann definieren Sie bitte `USE_XLIB` (impliziert durch `USE_IMAKE`) und fügen ihn der entsprechenden Kategorie hinzu.
x11-clocks	X11-Uhren.	
x11-drivers	X11-Treiber.	
x11-fm	X11-Dateimanager.	
x11-fonts	X11-Schriftarten und Werkzeuge.	
x11-servers	X11-Server.	
x11-themes	X11-Themes.	
x11-toolkits	X11-Toolkits.	
x11-wm	X11-Window-Manager.	
xfce*	Ports in Zusammenhang mit Xfce.	
zope*	Zope-Unterstützung.	

5.3.3. Wählen der richtigen Kategorie

Da viele der Kategorien sich überlappen, müssen Sie oft festlegen, welches die primäre Kategorie Ihres Ports ist. Hierzu gibt es einige Regeln, welche diese Auswahl bestimmen. Hier ist die Liste der Regeln mit abnehmender Wichtigkeit:

- Die erste (primäre) Kategorie muss eine physische (keine virtuelle, siehe oben) sein. Dies ist notwendig damit Pakete erstellt werden können. Die nachfolgenden Kategorien können wahllos virtuelle oder physische Kategorien sein.
- Sprachspezifische Kategorien kommen immer zuerst. Wenn Ihr Port z.B. Japanische X11-Schriftarten installiert, dann muss Ihre `CATEGORIES`-Zeile japanese x11-fonts enthalten.
- Spezifische Kategorien werden vor weniger spezifischen Kategorien aufgelistet. Ein HTML-Editor sollte z.B. als www editors aufgeführt werden und nicht umgekehrt. Genauso sollten Sie keinen Port unter net aufführen, wenn er zu irc, mail, news, security oder www passt, da net in diesen Kategorien bereits implizit eingeschlossen ist.

- x11 wird nur als sekundäre Kategorie benutzt, wenn die primäre Kategorie eine sprachspezifische ist. Keinesfalls sollten Sie x11 in die Kategorie-Zeile einer X-Applikation setzen.
- Emacs modes gehören in die gleiche Kategorie wie die vom jeweiligen mode unterstützte Applikation und nicht in editors. Ein Emacs mode z.B. für das Editieren von Quelltext einer bestimmten Programmiersprache gehört zur Kategorie lang.
- Für Ports, die vom Benutzer ladbare Kernelmodule installieren, sollte die virtuelle Kategorie kld in die `CATEGORIES`-Zeile aufgenommen werden.
- misc sollte nicht zusammen mit irgendeiner anderen nicht-virtuellen Kategorie auftreten. Falls Sie `misc` mit einer anderen Kategorie in `CATEGORIES` haben bedeutet dies, dass Sie gefahrlos `misc` streichen und die andere Kategorie alleine verwenden können!
- Falls Ihr Port wirklich in keine andere Kategorie passt, verwenden Sie bitte misc.

Falls Sie sich über die Kategorie im Unklaren sind, hinterlassen Sie bitte einen Kommentar in Ihrem per send-pr(1) eingereichten Bericht, damit wir diese Frage vor dem Import diskutieren können. Falls Sie ein Committer sind, schicken Sie bitte eine Nachricht an FreeBSD ports, damit die Frage im Vorhinein erörtert werden kann. Neue Ports werden zu häufig falsch kategorisiert und werden sofort wieder verschoben. Das bläht das Master Source Repository unnötig auf.

5.3.4. Eine neue Kategorie vorschlagen

Da die Ports-Sammlung über viele Jahre gewachsen ist, wurden viele neue Kategorien hinzugefügt. Neue Kategorien können *virtuell* (ohne eigenes Unterverzeichnis in der Ports-Sammlung) oder *physisch* sein. Der nachfolgende Text führt einige Punkte auf, welche bei der Neueinführung einer physischen Kategorie beachtet werden müssen, damit Sie dies bei einem eventuellen Vorschlag Ihrerseits berücksichtigen können.

Unsere bestehende Maxime ist die Vermeidung der Neuanlage von physischen Kategorien, solange nicht eine große Zahl von Ports zugeordnet werden können oder falls ihr nicht Ports zugehören würden, welche eine logisch abgegrenzte Gruppe von limitiertem öffentlichem Interesse zugehören würden (zum Beispiel neue Sprachkategorien) oder vorzugsweise beides.

Die Erklärung dafür ist, dass eine Neuanlage einer physischen Kategorie einen erheblichen Arbeitsaufwand sowohl für die Committer als auch diejenigen Nutzer bedeutet, welche die Änderungen der Ports-Sammlung nachvollziehen. Zusätzlich verursachen Vorschläge für neue Kategorien oftmals Kontroversen (natürlich deswegen, weil es keinen klaren Konsens darüber gibt, welche Kategorie als "zu groß" betrachtet werden muss noch ob sich bestimmte Kategorien zur einfachen Suche eignen (und wie viele Kategorien überhaupt ideal wären) und so weiter).

Hier ist das Prozedere:

1. Schlagen Sie die neue Kategorie auf FreeBSD ports vor. Sie sollten eine detaillierte Begründung für die neue Kategorie beifügen einschließlich einer Erklärung, warum Sie meinen, die existierenden Kategorien seien nicht ausreichend. Zeigen Sie außerdem eine Liste der zu verschiebenden Ports (falls neue Ports in GNATS auf ihren commit warten, die in diese Kategorie passen würden. Listen Sie diese bitte auch mit auf). Sind Sie der Maintainer oder Einreicher dieser Ports, erwähnen Sie es bitte. Es verleiht Ihrem Vorschlag

mehr Gewicht.

2. Nehmen Sie an der Diskussion teil.
3. Falls es Unterstützung für Ihren Vorschlag geben sollte, reichen Sie bitte einen PR ein, welcher die Begründung und die Liste der betroffenen Ports enthält, die verschoben werden müssen. Idealerweise sollte der PR Patches für Folgendes enthalten:
   - Makefiles für die neuen Ports nach dem Repocopy
   - Makefile für die neue Kategorie
   - Makefile für die alten Kategorien der betroffenen Ports
   - Makefiles für Ports, welche von den alten Ports abhängen
   - Für zusätzliches Ansehen sorgen Sie, wenn Sie die anderen Dateien, die geändert werden müssen, beifügen wie in der Direktive des Committer's Guide beschrieben.
4. Da es die Ports-Infrastruktur beeinflusst und nicht nur die Durchführung von Repocopies und möglicherweise sogar Regressionstests auf dem Build Cluster durchgeführt werden müssen, sollte der PR dem Ports Management Team Ports Management Team <portmgr@FreeBSD.org> zugeordnet werden.
5. Sobald der PR bestätigt wurde muss ein Committer den Rest der Prozedur durchführen, welche im Committers Guide beschrieben ist.

Das Vorschlagen einer neuen virtuellen Kategorie ist ähnlich, aber wesentlich weniger aufwendig, weil keine Ports verschoben werden müssen. In diesem Falle müssen nur die Patches an den PR beigefügt werden, welche die neue Kategorie zur Variable `CATEGORIES` der betroffenen Ports hinzufügen.

5.3.5. Vorschlagen einer Neuorganisation aller Kategorien

Von Zeit zu Zeit schlägt jemand eine komplette Neuorganisation aller Ports, entweder mit einer zweistufigen Struktur oder irgendeiner Art von Schlüsselwörtern, vor. Bis heute wurde keiner dieser Vorschläge umgesetzt, weil sie zwar einfach zu machen sind, aber der Aufwand zur Umsetzung und Reorganisation der kompletten Ports-Sammlung schlichtweg mörderisch wäre. Bitte lesen Sie die Geschichte dieser Vorschläge in den Archiven der Mailinglisten nach, bevor Sie diese Ideen nochmals unterbreiten. Zudem sollten Sie gewappnet sein, dass man Sie auffordert, einen arbeitsfähigen Prototyp vorzulegen.

5.4. Die Distributionsdateien

Der zweite Teil des Makefile beschreibt die Dateien, welche heruntergeladen werden müssen, um den Port zu bauen und wo diese Dateien zu finden sind.

5.4.1. `DISTVERSION/DISTNAME`

`DISTNAME` ist der Name der Applikation wie er von den Autoren vergeben wurde. `DISTNAME` hat als Vorgabe `${PORTNAME}-${PORTVERSION}` also überschreiben Sie diese Vorgabe nur, wenn es notwendig ist. `DISTNAME` wird nur an zwei Stellen genutzt. Erstens: (`DISTFILES`) hat als Vorgabe `${DISTNAME}${EXTRACT_SUFX}`. Zweitens: Die Distributionsdatei soll in einem Unterverzeichnis

namens WRKSRC extrahiert werden, dessen Vorgabe work/${DISTNAME} ist.

Manche Drittanbieter-Namen, welche nicht in das Schema ${PORTNAME}-${PORTVERSION} passen, können durch Setzen von DISTVERSION automatisch behandelt werden. PORTVERSION und DISTNAME werden automatisch abgeleitet, können aber natürlich manuell überschrieben werden. Die folgende Tabelle führt einige Beispiele auf:

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



PKGNAMEPREFIX und PKGNAMESUFFIX beeinflussen DISTNAME nicht. Beachten Sie bitte auch, dass Sie DISTNAME unverändert lassen sollten, falls WRKSRC denselben Wert hat wie work/${PORTNAME}-${PORTVERSION} und gleichzeitig dass Archiv des originalen Quelltextes anders benannt ist als ${PORTNAME}-${PORTVERSION}${EXTRACT_SUFX}. Es ist einfacher DISTFILES zu definieren, als DISTNAME und WRKSRC (und möglicherweise EXTRACT_SUFX) zu setzen.

5.4.2. MASTER_SITES

Dokumentieren Sie das Verzeichnis der FTP/HTTP-URL, welche auf den originalen Tarball zeigt, in der Variable MASTER_SITES. Bitte vergessen Sie niemals den Schrägstrich (/) am Ende!

Die make-Makros werden versuchen, diese Festlegung für die Aufbereitung der Distributionsdateien mittels FETCH zu benutzen, falls sie diese nicht schon auf dem System finden.

Es wird empfohlen, mehrere Webseiten in dieser Liste aufzuführen, vorzugsweise auf verschiedenen Kontinenten. Dies ist ein Schutz gegen Probleme bei größeren Ausfällen im Internet. Wir planen sogar Unterstützung einzubauen, die automatisch einen Server in der Nähe zum Herunterladen bestimmt. Die Verfügbarkeit von vielen Webseiten wird dieses Vorhaben beträchtlich erleichtern.

Falls der originale Tarball Teil eines populären Archivs ist, wie SourceForge, GNU oder Perl CPAN, können Sie möglicherweise auf diese Seiten in einer einfachen und kompakten Form mittels MASTER_SITE_* (d.h., MASTER_SITE_SOURCEFORGE,, MASTER_SITE_GNU und MASTER_SITE_PERL_CPAN) referenzieren. Setzen Sie einfach MASTER_SITES auf eine dieser Variablen und MASTER_SITE_SUBDIR auf den Pfad innerhalb des Archivs. Hier ist ein Beispiel:

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

Oder verwenden Sie ein kondensiertes Format:

```
MASTER_SITES=    GNU/make
```

Diese Variablen werden in /usr/ports/Mk/bsd.sites.mk definiert. Es werden ständig neue Einträge hinzugefügt, daher stellen Sie bitte unbedingt sicher, dass Sie die neueste Version verwenden, bevor Sie einen Port einschicken.

Für beliebte Seiten existieren sogenannte *magic*-Makros, die eine bestimmte Verzeichnisstruktur erstellen. Um eines dieser Makros zu verwenden, geben Sie dessen Abkürzung an und Ihr System wird versuchen, das korrekte Unterverzeichnis automatisch zu bestimmen.

```
MASTER_SITES=    SF
```

Ist das Ergebnis nicht korrekt, können Sie diesen Wert auch überschreiben.

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

Tabelle 1. Beliebte magic MASTER_SITES-Makros

Makro	Erwartetes Unterverzeichnis
APACHE_JAKARTA	/dist/jakarta/${PORTNAME:S,-,,/,}/source
BERLIOS	/${PORTNAME:L}
CHEESESHOP	/packages/source/source/${DISTNAME:C/(.).*/\1/}/${DISTNAME:C/(.*)-[0-9].*/\1/}
DEBIAN	/debian/pool/main/${PORTNAME:C/^((lib)?.).*$/\1/}/${PORTNAME}
GCC	/pub/gcc/releases/${DISTNAME}
GNOME	/pub/GNOME/sources/${PORTNAME}/${PORTVERSION:C/^(\.[0-9]).*/\1/}
GNU	/gnu/${PORTNAME}
MOZDEV	/pub/mozdev/${PORTNAME:L}
PERL_CPAN	/pub/CPAN/modules/by-module/${PORTNAME:C/-.*//}
PYTHON	/ftp/python/${PYTHON_PORTVERSION:C/rc[0-9]//}
RUBYFORGE	/${PORTNAME:L}
SAVANNAH	/${PORTNAME:L}
SF	/project/${PORTNAME:L}/${PORTNAME:L}/${PORTVERSION}

5.4.3. EXTRACT_SUFX

Falls Sie eine Distributionsdatei haben, die ein eigentümliches Suffix nutzt, um die Art der Kompression anzuzeigen, dann setzen Sie EXTRACT_SUFX.

Ist die Distributionsdatei zum Beispiel im Stil von foo.tgz anstatt des normalen foo.tar.gz benannt,

würden Sie schreiben:

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

Falls erforderlich, setzen die Variablen USE_BZIP2 und USE_ZIP automatisch EXTRACT_SUFX auf .tar.bz2 oder .zip. Falls keine der beiden gesetzt ist, dann verwendet EXTRACT_SUFX die Vorgabe .tar.gz.



Sie müssen niemals beide Variablen EXTRACT_SUFX und DISTFILES setzen.

5.4.4. DISTFILES

Manchmal haben die zu ladenden Dateien keinerlei Ähnlichkeit mit dem Namen des Ports. Es könnte z.B. source.tar.gz oder ähnlich heißen. In anderen Fällen könnte der Quelltext in mehreren Archiven sein und alle müssen heruntergeladen werden.

Falls dies der Fall ist, setzen Sie DISTFILES als eine durch Leerzeichen getrennte Liste aller Dateien, die geladen werden müssen.

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

Wenn nicht ausdrücklich gesetzt, verwendet DISTFILES als Vorgabe ${DISTNAME}${EXTRACT_SUFX}.

5.4.5. EXTRACT_ONLY

Falls nur einige der DISTFILES extrahiert werden müssen (z.B. eine Datei ist der Quelltext und eine andere ist ein unkomprimiertes Dokument), dann listen Sie die zu extrahierenden Dateien in EXTRACT_ONLY auf.

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

Falls *keine* der DISTFILES unkomprimiert sein sollte, dann setzen Sie EXTRACT_ONLY auf einen leeren String.

```
EXTRACT_ONLY=
```

5.4.6. PATCHFILES

Falls Ihr Port zusätzliche Patches benötigt, welche per FTP oder HTTP verfügbar sind, dann setzen Sie PATCHFILES auf den Namen der Dateien und PATCH_SITES auf die URL des Verzeichnisses, das diese Patches enthält (das Format ist das gleiche wie MASTER_SITES).

Falls ein Patch wegen einiger zusätzlicher Pfadnamen nicht relativ zum Anfang des

Quelltextbaumes (d.h., WRKSRC) liegt, dann setzen Sie bitte PATCH_DIST_STRIP entsprechend. Wenn z.B. alle Pfadnamen in diesem Patch ein zusätzliches foozolix-1.0/ vor ihren Dateinamen aufweisen, dann setzen Sie bitte PATCH_DIST_STRIP=-p1.

Kümmern Sie sich nicht darum, ob die Patches komprimiert sind. Sie werden automatisch dekomprimiert, wenn die Dateinamen auf .gz oder .Z enden.

Falls der Patch zusammen mit anderen Dateien in einem gezippten Tarball verteilt wird (z.B. mit Dokumentation), dann können Sie nicht PATCHFILES verwenden. In diesem Fall fügen Sie den Namen und den Ort dieses Tarballs zu DISTFILES und MASTER_SITES. Benutzen Sie dann die EXTRA_PATCHES -Variable, um auf diese Dateien zu zeigen und bsd.port.mk wird automatisch diese Dateien nutzen. Kopieren Sie *niemals* Patch-Dateien in das PATCHDIR-Verzeichnis, weil es möglicherweise nicht beschreibbar ist.



Der Tarball wird zusammen mit dem anderen Quelltext extrahiert werden. Eine ausdrückliche Dekomprimierung eines mit gzip oder compress erzeugten Tarball ist nicht notwendig. Sollten Sie dies dennoch vorgeben, so beachten Sie bitte peinlich genau, dass Sie nichts überschreiben, was bereits im Verzeichnis vorhanden ist. Vergessen Sie auch nicht den kopierten Patch im Target von pre-clean zu entfernen.

5.4.7. Verschiedene Distributionsdateien oder Patches von verschiedenen Seiten und Verzeichnissen (MASTER_SITES:n)

(Betrachten Sie es als in irgendeiner Form "fortgeschrittenes Thema". Neulinge sollten möglicherweise diesen Abschnitt beim ersten Lesen überspringen).

Dieser Abschnitt stellt Informationen über die Mechanismen zum Herunterladen von Dateien zur Verfügung und behandelt die Variablen MASTER_SITES:n und MASTER_SITES_NN. Wir beziehen uns im weiteren Text auf diese Variablen als MASTER_SITES:n.

Etwas Hintergrundinformation zu Beginn: OpenBSD verfügt über eine sehr elegante Option innerhalb der Variablen DISTFILES und PATCHFILES. Sowohl Dateien als auch Patches können mit angehängten :n-Bezeichnern versehen werden wobei n in beiden Fällen [0-9] sein kann und eine Gruppenzugehörigkeit anzeigt. Ein Beispiel hierfür ist:

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

In OpenBSD wird die Datei alpha mit der Variable MASTER_SITES0 verknüpft anstatt dem in FreeBSD gebräuchlichen MASTER_SITES und beta mit MASTER_SITES1.

Das ist eine sehr interessante Möglichkeit, die endlose Suche nach der richtigen Download-Seite zu verkürzen.

Stellen Sie sich zwei Dateien in DISTFILES und 20 Webseiten in der Variable MASTER_SITES vor. Alle Seiten sind erschreckend langsam, beta findet sich auf allen Seiten in MASTER_SITES und alpha kann nur auf der zwanzigsten Seite gefunden werden. Wäre es nicht reine Verschwendung, wenn der Maintainer alle Seiten zuvor überprüfen müsste? Kein guter Start für das wundervolle

Wochenende!

Übertragen Sie diesen Umstand auf noch mehr DISTFILES und mehr MASTER_SITES. Ganz sicher würde unser "distfiles survey master" die Erleichterung sehr zu schätzen wissen, die eine solche Verringerung der Netzwerkbelastung bringen würde.

In den nächsten Abschnitten sehen Sie die Implementierung dieser Idee durch FreeBSD. Dabei wurde das Konzept von OpenBSD ein wenig verbessert.

Prinzipielle Information

Dieser Abschnitt informiert Sie, wie Sie schnell ein fein granuliertes Herunterladen von vielen Dateien und Fehlerbereinigungen von verschiedenen Webseiten und Unterverzeichnissen bewerkstelligen. Wir beschreiben hier den Fall der vereinfachten Nutzung von MASTER_SITES:n. Das ist für die meisten Szenarien ausreichend. Falls Sie weitere Informationen benötigen, sollten Sie den nächsten Abschnitt lesen.

Einige Programme bestehen aus mehreren Dateien, welche von verschiedenen Webseiten heruntergeladen werden müssen. Zum Beispiel besteht Ghostscript aus dem Kern des Programms und einer großen Zahl von Treiberdateien, die vom Drucker des Benutzers abhängen. Einige dieser Treiberdateien werden mit der Kernapplikation mitgeliefert aber viele müssen von verschiedenen Webseiten heruntergeladen werden.

Um das zu unterstützen, muss jeder Eintrag in DISTFILES mit einem Komma und einem "tag name" abgeschlossen werden. Jeder in MASTER_SITES aufgeführte Webseite folgt ein Komma und eine Marke (tag), die anzeigt, welche Datei von dieser Webseite heruntergeladen werden kann.

Stellen Sie sich bitte eine Applikation vor, deren Quelltext in zwei Teile aufgeteilt ist, source1.tar.gz und source2.tar.gz, welche von zwei verschiedenen Webseiten heruntergeladen werden müssen. Das Makefile des Port würde Zeilen enthalten wie in Vereinfachtes Beispiel für den Gebrauch von MASTER_SITES:n mit einer Datei pro Webseite.

Beispiel 1. Vereinfachtes Beispiel für den Gebrauch von MASTER_SITES:n mit einer Datei pro Webseite

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

Verschiedene Dateien können die gleiche Marke aufweisen. Ausgehend vom vorherigen Beispiel nehmen wir an, dass es noch eine dritte Datei gibt (source3.tar.gz), welche von ftp.example2.com heruntergeladen werden soll. Das Makefile würde dann aussehen wie Vereinfachtes Beispiel für den Gebrauch von MASTER_SITES:n mit mehr als einer Datei pro Webseite.

Beispiel 2. Vereinfachtes Beispiel für den Gebrauch von MASTER_SITES:n mit mehr als einer Datei pro Webseite

```
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
```

Ausführliche Information

In Ordnung, das vorherige Beispiel reicht nicht für Ihre Bedürfnisse? In diesem Abschnitt werden wir im Detail erklären, wie der fein granulierte Mechanismus zum Herunterladen (MASTER_SITES:n) funktioniert und wie Sie Ihre Ports modifizieren, um ihn zu nutzen.

1. Elemente können nachstehend bezeichnet werden mit :n wobei *n* in diesem Falle ` ist. Das heißt _n_ könnte theoretisch jede alphanumerische Zeichenkette sein, aber wir beschränken sie auf `[a-zA-Z_][0-9a-zA-Z_] für diesen Moment.

   Zudem ist die Zeichenkette case sensitive; d.h. n unterscheidet sich von N.

   Allerdings dürfen die folgenden Wörter nicht gebraucht werden, da sie spezielle Bedeutungen haben: default, all und ALL (diese Wörter werden intern genutzt in Punkt ii). Ausserdem ist DEFAULT ein reserviertes Wort (beachten Sie 3).

2. Elemente mit angehängtem :n gehören zur Gruppe n, :m gehört zur Gruppe m und so weiter.
3. Elemente ohne Anhängsel sind gruppenlos, d.h. sie gehören alle zu der speziellen Gruppe DEFAULT. Falls sie an irgendeinem Element DEFAULT hängen, ist dies überflüssig, es sei denn Sie wollen, dass ein Element sowohl zu DEFAULT als auch anderen Gruppen gleichzeitig gehört (beachten Sie 5).

   Die folgenden Beispiele sind gleichwertig, aber das erste Beispiel ist vorzuziehen:

   ```
   MASTER_SITES=   alpha

   MASTER_SITES=   alpha:DEFAULT
   ```

4. Gruppen sind nicht ausschliessend, d.h. ein Element kann mehreren Gruppen gleichzeitig angehören und eine Gruppe wiederum kann entweder mehrere Elemente oder überhaupt keine aufweisen. Wiederholte Elemente sind schlicht nur wiederholte Elemente.
5. Wenn Sie wollen, dass ein Element gleichzeitig zu mehreren Gruppen gehört, dann können Sie diese durch ein Komma (,) trennen.

   Anstatt jedes Mal ein anderes Anhängsel zu verwenden und Wiederholungen aufzuführen, können Sie mehrere Gruppen auf einmal in einem einzigen Anhängsel bestimmen. Zum Beispiel markiert :m,n,o ein Element, welches zu den Gruppen m, n und o gehört.

Alle folgenden Beispiele sind gleichwertig, aber das erste Beispiel ist vorzuziehen:

```
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. Alle Webseiten in einer Gruppe werden gemäß MASTER_SORT_AWK sortiert. Alle Gruppen innerhalb von MASTER_SITES und PATCH_SITES werden genauso sortiert.
7. Gruppensemantik kann benutzt werden in den folgenden Variablen: MASTER_SITES, PATCH_SITES, MASTER_SITE_SUBDIR, PATCH_SITE_SUBDIR, DISTFILES und PATCHFILES entsprechend der folgenden Syntax:
   a. Elemente mit MASTER_SITES, PATCH_SITES, MASTER_SITE_SUBDIR und PATCH_SITE_SUBDIR müssen mit einem Schrägstrich beendet werden (/). Falls Elemente zu irgendwelchen Gruppen gehören, muss :n direkt nach dem Trenner / stehen. Der MASTER_SITES:n-Mechanismus verlässt sich auf das Vorhandensein des Trennzeichens /, um verwirrende Elemente zu vermeiden in denen :n ein zulässiger Bestandteil des Elementes ist und das Auftreten von :n die Gruppe n anzeigt. Aus Kompatibilitätsgründen (da der /-Trenner sowohl in MASTER_SITE_SUBDIR als auch PATCH_SITE_SUBDIR-Elementen nicht erforderlich ist) wird, falls das auf das Anhängsel folgende nächste Zeichen kein / ist, auch :n als gültiger Teil des Elementes behandelt anstatt als Gruppenzusatz, selbst wenn ein Element ein angehängtes :n aufweist. Beachten Sie sowohl Ausführliches Beispiel von MASTER_SITES:n in MASTER_SITE_SUBDIR als auch Ausführliches Beispiel von MASTER_SITES:n mit Komma-Operator, mehreren Dateien, mehreren Webseiten und mehreren Unterverzeichnissen.

Beispiel 3. Ausführliches Beispiel von MASTER_SITES:n in MASTER_SITE_SUBDIR

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

- Verzeichnisse innerhalb der Gruppe DEFAULT → old:n
- Verzeichnisse innerhalb der Gruppe NEW → new

Beispiel 4. Ausführliches Beispiel von MASTER_SITES:n mit Komma-Operator, mehreren Dateien, mehreren Webseiten und mehreren Unterverzeichnissen

```
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
```

Das vorstehende Beispiel führt zu einem fein granulierten Herunterladen. Die Webseiten werden in der exakten Reihenfolge ihrer Nutzung aufgelistet.

- file1 wird heruntergeladen von
  - MASTER_SITE_OVERRIDE
  - http://site1/directory-trial:1/
  - http://site1/directory-one/
  - http://site1/directory/
  - http://site2/
  - http://site7/
  - MASTER_SITE_BACKUP
- file2 wird genauso heruntergeladen wie file1, da sie zur gleichen Gruppe gehören
  - MASTER_SITE_OVERRIDE
  - http://site1/directory-trial:1/
  - http://site1/directory-one/
  - http://site1/directory/
  - http://site2/
  - http://site7/
  - MASTER_SITE_BACKUP
- file3 wird heruntergeladen von
  - MASTER_SITE_OVERRIDE
  - http://site3/
  - MASTER_SITE_BACKUP
- file4 wird heruntergeladen von
  - MASTER_SITE_OVERRIDE
  - http://site4/
  - http://site5/
  - http://site6/
  - http://site7/
  - http://site8/directory-one/
  - MASTER_SITE_BACKUP

- file5 wird heruntergeladen von
  - MASTER_SITE_OVERRIDE
  - MASTER_SITE_BACKUP
- file6 wird heruntergeladen von
  - MASTER_SITE_OVERRIDE
  - http://site8/
  - MASTER_SITE_BACKUP

8. Wie gruppiere ich eine der speziellen Variablen aus bsd.sites.mk, d.h. MASTER_SITE_SOURCEFORGE?

   Lesen Sie Ausführliches Beispiel von MASTER_SITES:n mit MASTER_SITE_SOURCEFORGE.

   Beispiel 5. Ausführliches Beispiel von MASTER_SITES:n mit MASTER_SITE_SOURCEFORGE

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

   something.tar.gz wird von allen Webseiten innerhalb von MASTER_SITE_SOURCEFORGE heruntergeladen.

9. Wie nutze ich dies mit PATCH*-Variablen.

   In allen Beispielen wurden MASTER*-Variablen genutzt, aber sie funktionieren exakt genauso mit PATCH*-Variablen, wie Sie an Vereinfachte Nutzung von MASTER_SITES:n mit PATCH_SITES.. sehen können.

   Beispiel 6. Vereinfachte Nutzung von MASTER_SITES:n mit PATCH_SITES.

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

Was ändert sich für die Ports? Was ändert sich nicht?

i. Alle bestehenden Ports bleiben gleich. Der Code für MASTER_SITES:n wird nur aktiviert, falls es Elemente mit angehängtem :n entsprechend den zuvor erwähnten Syntax-Regeln wie in 7 gezeigt gibt.

ii. Das Target des Port bleibt gleich: checksum, makesum, patch, configure, build etc. Mit der offensichtlichen Ausnahme von do-fetch, fetch-list, master-sites und patch-sites.

   - do-fetch: nutzt die neue Gruppierung DISTFILES und PATCHFILES mit ihren darauf zutreffenden Gruppenelementen in MASTER_SITES und PATCH_SITES welche zutreffende Gruppenelemente sowohl in MASTER_SITE_SUBDIR als auch PATCH_SITE_SUBDIR aufweisen. Sehen Sie hierzu Ausführliches Beispiel von MASTER_SITES:n mit Komma-Operator, mehreren

Dateien, mehreren Webseiten und mehreren Unterverzeichnissen.

- `fetch-list`: arbeitet wie das alte `fetch-list` mit der Ausnahme, dass es nur wie `do-fetch` gruppiert.
- `master-sites` und `patch-sites`: (inkompatibel zu älteren Versionen) geben nur die Elemente der Gruppe `DEFAULT` zurück. Beziehungsweise sie führen genau genommen die Targets von `master-sites-default` und `patch-sites-default` aus.

  Weiterhin ist der Gebrauch des Target entweder von `master-sites-all` oder `patch-sites-all` der direkten Überprüfung von `MASTER_SITES` oder `PATCH_SITES` vorzuziehen. Zudem ist nicht garantiert, dass das direkte Überprüfen in zukünftigen Versionen funktionieren wird. Sehen Sie B für weitere Informationen zu diesen neuen Port-Targets.

iii. Neue Port-Targets

a. Es gibt `master-sites-n` und `patch-sites-n`-Targets, welche die Elemente der jeweiligen Gruppe *n* innerhalb von `MASTER_SITES` und `PATCH_SITES` auflisten. Beispielweise werden sowohl `master-sites-DEFAULT` als auch `patch-sites-DEFAULT` die Elemente der Gruppe `DEFAULT`, `master-sites-test` und `patch-sites-test` der Gruppe `test` usw. zurückgeben.

b. Es gibt das neue Target `master-sites-all` und `patch-sites-all`, welche die Arbeit der alten Targets `master-sites` und `patch-sites` übernehmen. Sie geben die Elemente aller Gruppen zurück,als würden sie zur gleichen Gruppe gehören - mit dem Vorbehalt, dass sie so viele `MASTER_SITE_BACKUP` und `MASTER_SITE_OVERRIDE` auflisten wie Gruppen mittels `DISTFILES` oder `PATCHFILES` definiert sind. Das gleiche gilt entsprechend für `master-sites-all` und `patch-sites-all`.

5.4.8. DIST_SUBDIR

Verhindern Sie, dass Ihr Port das Verzeichnis /usr/ports/distfiles in Unordnung bringt. Falls Ihr Port eine ganze Reihe von Dateien herunterladen muss oder eine Datei enthält, die einen Namen hat, der möglicherweise mit anderen Ports in Konflikt stehen könnte (d.h.Makefile), dann setzen Sie die Variable `DIST_SUBDIR` auf den Namen des Ports (`${PORTNAME}` oder `${PKGNAMEPREFIX}${PORTNAME}` sollte hervorragend funktionieren). Dies wird `DISTDIR` von der Vorgabe /usr/ports/distfiles auf /usr/ports/distfiles/DIST_SUBDIR ändern und stellt tatsächlich alle für Ihren Port benötigten Dateien in dieses Unterverzeichnis.

Es wird zusätzlich nach dem Unterverzeichnis mit dem gleichen Namen auf der Sicherung der Hauptseite auf ftp.FreeBSD.org suchen (das ausdrückliche Setzen von `DISTDIR` in Ihrem `Makefile` wird dies nicht gewährleisten, also nutzen Sie bitte `DIST_SUBDIR`).



Dies hat keine Auswirkungen auf die Variable `MASTER_SITES`, die Sie in Ihrem Makefile definieren.

5.4.9. ALWAYS_KEEP_DISTFILES

Falls Ihr Port binäre Distfiles benutzt und eine Lizenz aufweist, die verlangt, dass das der Quelltext in Form binärer Pakete verteilt werden muss, z.B. GPL, dann wird `ALWAYS_KEEP_DISTFILES` den FreeBSD Build Cluster anweisen eine Kopie der Dateien in `DISTFILES` vorzuhalten. Nutzer dieser Ports benötigen generell diese Dateien nicht, daher ist es ein gutes Konzept, nur dann die Distfiles

zu DISTFILES hinzuzufügen, wenn PACKAGE_BUILDING definiert ist.

Beispiel 7. Nutzung von ALWAYS_KEEP_DISTFILES.

```
.if defined(PACKAGE_BUILDING)
DISTFILES+=             foo.tar.gz
ALWAYS_KEEP_DISTFILES=  yes
.endif
```

Wenn Sie zusätzliche Dateien zu DISTFILES hinzufügen, dann beachten Sie bitte, dass Sie diese auch in distinfo aufführen. Zudem werden die zusätzlichen Dateien normalerweise ebenso in WRKDIR extrahiert, was für einige Ports zu unbeabsichtigten Seiteneffekten führen mag und spezielle Behandlung erfordert.

5.5. MAINTAINER

Fügen Sie hier Ihre E-Mailadresse ein. Bitte. *:-)*

Beachten Sie bitte, dass nur eine einzelne E-Mailadresse ohne Kommentar in der Variable MAINTAINER zulässig ist. Das Format sollte user@hostname.domain sein. Bitte fügen Sie keinen beschreibenden Text wie z.B. Ihren wirklichen Namen ein, dies verwirrt lediglich bsd.port.mk.

Der Maintainer ist dafür verantwortlich, dass der Port aktuell gehalten wird und er sorgt dafür, dass der Port korrekt arbeitet. Für eine detaillierte Beschreibung der Verantwortlichkeiten eines Maintainers beachten Sie bitte den Abschnitt Die Herausforderung für einen Port-Maintainer.

Änderungen am Port werden dem Maintainer zur Begutachtung und Zustimmung vorgelegt, bevor sie committed werden. Falls der Maintainer einem Aktualisierungs-Wunsch nicht binnen 2 Wochen (ausgenommen wichtige öffentliche Feiertage) zustimmt, dann wird dies als Maintainer-Timeout betrachtet und eine Aktualisierung kann ohne ausdrückliche Zustimmung des Maintainers erfolgen. Falls der Maintainer nicht binnen 3 Monaten zustimmt, wird er als abwesend ohne Grund betrachtet und kann als Maintainer des fraglichen Ports durch eine andere Person ersetzt werden. Ausgenommen davon ist alles, was durch das Ports Management Team <portmgr@FreeBSD.org> oder das Security Officer Team <security-officer@FreeBSD.org> betreut wird. Es dürfen niemals committs ohne vorherige Zustimmung an solchen Ports vorgenommen werden!

Wir behalten uns das Recht vor, die Einreichungen eines Maintainers ohne ausdrückliche Zustimmung zu ändern, falls wir der Auffassung sind, dass dadurch die Einhaltung von Richtlinien und stilistischen Vorgaben für die Ports-Sammlung besser erfüllt wird. Zudem können größere Änderungen an der Infrastruktur der Ports zu Änderungen an einem bestimmten Port ohne Zustimmung des Maintainers führen. Diese Änderungen beeinflussen niemals die Funktionalität eines Ports.

Das Ports Management Team <portmgr@FreeBSD.org> behält sich das Recht vor, die Maintainerschaft jedem aus irgendeinem Grund zu entziehen oder ausser Kraft zu setzen, und das Security Officer Team Security Officer Team <security-officer@FreeBSD.org> behält sich das Recht vor, jede Maintainerschaft aus Sicherheitsgründen aufzuheben oder ausser Kraft zu setzen.

5.6. COMMENT

Dies ist eine einzeilige Beschreibung des Ports. *Bitte* fügen Sie nicht den Paketnamen (oder die Version der Software) in den Kommentar ein. Der Kommentar soll mit einem Großbuchstaben beginnen und ohne Punkt enden. Hier ist ein Beispiel:

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

Die COMMENT-Variable soll unmittelbar nach der MAINTAINER-Variable im Makefile stehen.

Bitte versuchen Sie die COMMENT-Zeile auf weniger als 70 Zeichen zu begrenzen, da pkg_info(1) diese zur Anzeige einer kurzen, einzeiligen Zusammenfassung des Ports verwendet.

5.7. Abhängigkeiten (dependencies)

Viele Ports hängen von anderen Ports ab. Dies ist ein sehr praktisches und nettes Feature der meisten Unix-ähnlichen Betriebssysteme, FreeBSD nicht ausgeschlossen. Es erlaubt, dass häufig vorkommende Abhängigkeiten nicht mit jedem Port oder Paket zusammen ausgeliefert werden müssen, da viele Ports diese gemeinsam benutzen. Es gibt sieben Variablen, die benutzt werden können, um sicherzustellen, dass alle benötigten Teile auf dem Rechner des Nutzers sind. Zusätzlich gibt es einige vordefinierte Variablen für Abhängigkeiten in häufigen Fällen und einige, welche das Verhalten der Abhängigkeiten bestimmen.

5.7.1. LIB_DEPENDS

Diese Variable spezifiziert die Shared-Libraries, von denen der Port abhängt. Es ist eine Liste von lib:dir:target-Tupeln wobei *lib* den Name der gemeinsam genutzten Bibliothek, *dir* das Verzeichnis, in welchem sie zu finden ist, falls nicht verfügbar, und *target* das Target in diesem Verzeichnis angeben. Zum Beispiel wird

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

auf eine jpeg-Bibliothek mit der Hauptversionsnummer 9 prüfen, in das graphics/jpeg -Unterverzeichnis Ihrer Ports-Sammlung wechseln, es bauen und installieren, falls es nicht gefunden wird. Der *target*-Teil kann weggelassen werden, falls er identisch mit `DEPENDS_TARGET` ist (Vorgabe hierfür ist `install`).



Der *lib*-Teil ist ein regulärer Ausdruck, welcher die Ausgabe von `ldconfig -r` ausgewertet. Werte wie `intl.[5-7]` und `intl` sind zulässig. Das erste Muster, `intl.[5-7]`, stimmt überein mit: `intl.5`, `intl.6` oder `intl.7`. Das zweite Muster, `intl`, stimmt überein mit jeder Version der `intl`-Bibliothek.

Die Abhängigkeit wird zwei Mal überprüft, einmal innerhalb des `extract`-Target und dann innerhalb des `install`-Target. Zudem wird der Name der Abhängigkeit in das Paket eingefügt, damit pkg_add(1) es automatisch installiert, falls es nicht auf dem Rechner des Nutzers ist.

5.7.2. RUN_DEPENDS

Diese Variable legt Binärdateien oder Dateien, von denen der Port abhängt, für die Laufzeit fest. Es ist eine Liste von path:dir:target-Tupeln, wobei *path* der Name der Binärdatei oder Datei, *dir* das Verzeichnis, in welchem sie gefunden werden kann, falls nicht vorhanden, und *target* das Target in diesem Verzeichnis angeben. Falls *path* mit einem Slash (/) beginnt, wird es als Datei behandelt und deren Vorhandensein wird mit `test -e`; überprüft. Andernfalls wird angenommen, dass es eine Binärdatei ist und `which -s` wird benutzt, um zu überprüfen, ob das Programm im Pfad vorhanden ist.

Zum Beispiel wird

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

überprüfen, ob die Datei oder das Verzeichnis /usr/local/etc/innd existiert und es erstellen und installieren aus dem news/inn-Unterverzeichnis der Ports-Sammlung, falls es nicht gefunden wird. Es wird zudem überprüft, ob die Binärdatei namens `xmlcatmgr` im Suchpfad vorhanden ist und danach zum Unterverzeichnis textproc/xmlcatmgr in Ihrer Ports-Sammlung wechseln, es bauen und installieren, falls es nicht gefunden wird.



In diesem Fall ist `innd` eine Binärdatei. Falls sich eine Binärdatei an einem ungewöhnlichen Platz befindet, der nicht im Suchpfad ist, dann sollten Sie die volle Pfadangabe verwenden.



Der offizielle Suchpfad `PATH`, welcher im Ports Cluster benutzt wird, ist

```
/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin:/usr/X11R6
/bin
```

Die Abhängigkeit wird innerhalb des `install`-Target überprüft. Zudem wird der Name der Abhängigkeit in das Paket übernommen, damit pkg_add(1) es automatisch installieren wird, falls es auf dem System des Nutzers nicht vorhanden ist. Der *target*-Teil kann weggelassen werden, wenn er der gleiche ist wie in der Variable `DEPENDS_TARGET`.

Es kommt recht häufig vor, dass `RUN_DEPENDS` genau dasselbe enthält wie `BUILD_DEPENDS`, gerade dann, wenn die portierte Software in einer Skriptsprache geschrieben ist oder dieselbe Umgebung, die zum Bau verwendet wurde, zur Laufzeit gebraucht wird. In diesem Fall ist es sowohl verlockend als auch intuitiv, den Wert der einen Variable der anderen direkt zuzuweisen:

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

Jedoch kann eine solche Zuweisung dazu führen, dass die Liste der Laufzeitabhängigkeiten mit überflüssigen Einträgen belastet wird, die sich nicht in der ursprünglichen Liste `BUILD_DEPENDS` des Ports befanden, da sich make(1) bei der Auswertung solcher Zuweisungen träge verhält. Stellen Sie

sich ein Makefile mit USE_*-Variablen vor, die von ports/Mk/bsd.*.mk verarbeitet werden, um initiale Bauabhängigkeiten zusammenzutragen. Zum Beispiel fügt USE_GMAKE=yes devel/gmake zu BUILD_DEPENDS hinzu. Um zu verhindern, dass solche zusätzlichen Abhängigkeiten RUN_DEPENDS belasten, achten Sie darauf, bei gleichzeitiger Auswertung zuzuweisen, d.h. der Ausdruck wird ausgewertet, bevor er als Wert der Variablen zugewiesen wird:

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

5.7.3. BUILD_DEPENDS

Diese Variable legt Binärdateien oder Dateien fest, die dieser Port zur Erstellung benötigt. Wie RUN_DEPENDS ist es eine Liste von path:dir:target-Tupeln. Zum Beispiel wird

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

überprüfen, ob eine Binärdatei unzip vorhanden ist und in das Unterverzeichnis archivers/unzip Ihrer Ports-Sammlung wechseln und sie erstellen und installieren, falls sie nicht gefunden wird.



"Erstellen" bedeutet hier alles von der Extraktion bis zur Kompilierung. Die Abhängigkeit wird im extract-Target überprüft. Der *target*-Teil kann weggelassen werden, falls er identisch mit der Variable DEPENDS_TARGET ist.

5.7.4. FETCH_DEPENDS

Diese Variable legt eine Binärdatei oder Datei fest, welche der Port benötigt, um heruntergeladen werden zu können. Wie die vorherigen beiden Variablen ist er eine Liste von path:dir:target-Tupeln. Zum Beispiel wird

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

überprüfen, ob eine Binärdatei namens ncftp2 vorhanden ist, in das Unterverzeichnis net/ncftp2 Ihrer Ports-Sammlung wechseln, sie erstellen und installieren, falls sie nicht gefunden wird.

Die Abhängigkeit wird innerhalb des fetch-Target überprüft. Der *target*-Teil kann weggelassen werden, falls er identisch mit der Variable DEPENDS_TARGET ist.

5.7.5. EXTRACT_DEPENDS

Diese Variable spezifiziert eine Binärdatei oder eine Datei, welche dieser Port für die Extraktion benötigt. Wie die vorherigen Variablen ist er eine Liste von path:dir:target-Tupeln. Zum Beispiel wird

```
EXTRACT_DEPENDS=
```

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

überprüfen, ob eine Binärdatei namens `unzip` vorhanden ist, in das Unterverzeichnis archivers/unzip Ihrer Ports-Sammlung wechseln, sie erstellen und installieren, falls sie nicht gefunden wird.

Die Abhängigkeit wird innerhalb des `extract`-Target überprüft. Der *target*-Teil kann weggelassen werden, falls er identisch mit der Variable `DEPENDS_TARGET` ist.



Nutzen Sie diese Variable nur, wenn die Extraktion nicht funktioniert (die Vorgabe nimmt `gzip` an) und nicht mit `USE_ZIP` oder `USE_BZIP2` wie in `USE_*` beschrieben zum Laufen gebracht werden kann.

5.7.6. PATCH_DEPENDS

Diese Variable legt eine Binärdatei oder eine Datei fest, welche dieser Port zum Patchen benötigt. Wie die vorhergehenden Variablen ist diese eine Liste von path:dir:target-Tupeln. Zum Beispiel wird

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

in das Unterverzeichnis java/jfc Ihrer Ports-Sammlung wechseln, um es zu entpacken.

Die Abhängigkeit wird innerhalb des `patch`-Target überprüft. Der *target*-Teil kann entfallen, falls er identisch mit der Variable `DEPENDS_TARGET` ist.

5.7.7. USE_*

Es gibt eine Reihe von Variablen, um gebräuchliche Abhängigkeiten einzukapseln, die viele Ports aufweisen. Obwohl Ihre Verwendung optional ist, können sie helfen die Übersichtlichkeit des Makefile eines Ports zu erhöhen. Jede von ihnen ist im Stil von `USE_*`. Der Gebrauch dieser Variablen ist beschränkt auf das Makefile eines Ports und ports/Mk/bsd.*.mk. Es ist nicht entworfen worden, um durch den Nutzer setzbare Optionen einzukapseln; benutzen Sie `WITH_*` und `WITHOUT_*` für diese Zwecke.



Es ist *immer* falsch, irgendeine `USE_*`-Variable in der /etc/make.conf zu setzen. Zum Beispiel würde das Setzen von

```
USE_GCC=3.4
```

eine Abhängigkeit für GCC34 für jeden Port einschliesslich GCC34 selbst hinzufügen!

Tabelle 2. Die `USE_*`-Varibalen

Variable	Bedeutung
USE_BZIP2	Der Tarball dieses Ports wird mit bzip2 komprimiert.
USE_ZIP	Der Tarball des Ports wird mit zip komprimiert.
USE_BISON	Der Port benutzt bison für die Erstellung.
USE_CDRTOOLS	Der Port erfordert cdrecord entweder von sysutils/cdrtools oder sysutils/cdrtools-cjk, abhängig davon, was der Nutzer vorgibt.
USE_GCC	Dieser Port benötigt eine bestimmte Version von gcc zur Erstellung. Die genaue Version kann festgelegt werden mit Werten wie 3.4. Mit 3.4+ kann die mindestens erforderliche Version spezifiziert werden. Der gcc aus dem Basissystem wird genutzt, wenn er die erforderliche Version erfüllt, andernfalls wird eine geeignete Version des gcc aus den Ports kompiliert und die Variablen CC und CXX werden angepasst.

Variablen zugehörig zu gmake und dem configure-Skript werden in Build-Mechanismen beschrieben, währenddessen autoconf, automake und libtool in Benutzung von GNU autotools beschrieben sind. Perl-spezifische Variablen werden in Die Benutzung von perl behandelt. X11-Variablen sind aufgelistet in Benutzung von X11. Benutzung von GNOME behandelt GNOME-bezogene Variablen und Benutzung von KDE KDE-bezogene Variablen. Benutzung von Java dokumentiert Java-Variablen, während WebanwendungenInformationen zu Apache, PHP und PEAR-Modulen enthält. Python wird in Python benutzen und Ruby in Ruby benutzen erörtert. SDL verwenden stellt Variablen für SDL-Programme zur Verfügung und Xfce verwenden enthält schliesslich Variablen für Xfce.

5.7.8. Minimale Version einer Abhängigkeit

Eine minimale Version einer Abhängigkeit kann in jeder *_DEPENDS-Variable festgelegt werden mit Ausnahme von LIB_DEPENDS durch Anwendung folgender Syntax:

```
p5-Spiffy>=0.26:${PORTSDIR}/devel/p5-Spiffy
```

Das erste Feld enthält einen abhängigen Paketnamen, welcher einem Eintrag in der Paketdatenbank entsprechen muss und einen Vergleich mit einer Paketversion. Die Abhängigkeit wird erfüllt, wenn p5-Spiffy-0.26 oder eine neuere Version auf dem System installiert ist.

5.7.9. Anmerkungen zu Abhängigkeiten

Wie vorstehend beschrieben ist das Vorgabe-Target DEPENDS_TARGET, wenn eine Abhängigkeit benötigt wird. Die Vorgabe hierfür ist install. Dies ist eine Nutzer-Variable; sie wird niemals im Makefile eines Ports definiert. Falls Ihr Port einen besonderen Weg benötigt, um mit einer

Abhängigkeit umzugehen, dann benutzen Sie bitte den `:target`-Teil der `*_DEPENDS`-Variablen, anstatt `DEPENDS_TARGET` zu ändern.

Falls Sie `make clean` schreiben, werden dessen Abhängigkeiten auch gesäubert. Falls Sie dies nicht wollen, definieren Sie die Variable `NOCLEANDEPENDS` in Ihrer Umgebung. Dies kann besonders erstrebenswert sein, wenn der Port etwas in seiner Liste von Abhängigkeiten hat, das sehr viel Zeit für einen rebuild benötigt wie KDE, GNOME oder Mozilla.

Um von einem anderen Port bedingungslos abhängig zu sein, benutzen Sie bitte die Variable `${NONEXISTENT}` als erstes Feld von `BUILD_DEPENDS` oder `RUN_DEPENDS`. Benutzen Sie dies nur, wenn Sie den Quelltext eines anderen Port benötigen. Sie können auch oft Kompilierzeit sparen, wenn Sie das Target festlegen. Zum Beispiel wird

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

immer zum `jpeg`-Port wechseln und ihn extrahieren.

5.7.10. Zirkuläre Abhängigkeiten sind fatal



Führen Sie niemals irgendwelche zirkulären Abhängigkeiten in der Ports-Sammlung ein!

Die Struktur für die Erstellung von Ports dulde keinerlei zirkuläre Abhängigkeiten. Falls Sie dennoch eine verwenden, wird es irgendjemanden irgendwo auf der Welt geben, dessen FreeBSD-Installation nahezu sofort zusammenbricht und vielen anderen wird es sehr schnell genauso ergehen. So etwas kann extrem schwer festzustellen sein. Falls Sie Zweifel haben vor einer Änderung, dann vergewissern Sie sich, dass Sie folgendes getan haben: `cd /usr/ports; make index`. Dieser Prozess kann auf alten Maschinen sehr langsam sein, aber Sie ersparen sich und einer Vielzahl von Menschen möglicherweise eine Menge Ärger.

5.8. MASTERDIR

Falls Ihr Port wegen einer Variable, die verschiedene Werte annimmt (z.B. Auflösung oder Papiergröße), leicht unterschiedliche Versione von Paketen erzeugen muss, dann legen Sie bitte ein Unterverzeichnis pro Paket an, um es für den Nutzer einfacher begreiflich zu machen, was zu machen ist. Aber versuchen Sie dabei so viele Dateien wie möglich zwischen diesen Ports gemeinsam zu nutzen. Normalerweise benötigen Sie nur ein sehr kurzes Makefile in allen ausser einem Unterverzeichnis, wenn Sie Variablen intelligent nutzen. In diesem einzigen Makefile können Sie `MASTERDIR` verwenden, um anzugeben, wo der Rest der Dateien liegt. Benutzen Sie bitte auch eine Variable für `PKGNAMESUFFIX`, damit die Pakete unterschiedliche Namen haben werden.

Wir demonstrieren dies am Besten an einem Beispiel. Es ist Teil von japanese/xdvi300/Makefile;

```
PORTNAME=       xdvi
PORTVERSION=    17
PKGNAMEPREFIX=  ja-
PKGNAMESUFFIX=  ${RESOLUTION}
```

```
 :
# default
RESOLUTION?=   300
.if ${RESOLUTION} != 118 && ${RESOLUTION} != 240 && \
       ${RESOLUTION} != 300 && ${RESOLUTION} != 400
       @${ECHO_MSG} "Error: invalid value for RESOLUTION: \"${RESOLUTION}\""
       @${ECHO_MSG} "Possible values are: 118, 240, 300 (default) and 400."
       @${FALSE}
.endif
```

japanese/xdvi300 verfügt ebenfalls über alle Patches, Paket-Dateien usw. Wenn Sie `make` eintippen, wird der Port die Standardvorgabe für die Auflösung nehmen (300) und den Port ganz normal erstellen.

Genauso wie für alle anderen Auflösungen ist dies das *vollständige* xdvi118/Makefile:

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

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

(xdvi240/Makefile und xdvi400/Makefile sind ähnlich). Die `MASTERDIR`-Definition teilt dem bsd.port.mk mit, dass die normalen Unterverzeichnisse wie `FILESDIR` und `SCRIPTDIR` unter xdvi300 gefunden werden können. Die `RESOLUTION=118`-Zeile wird die `RESOLUTION=300`-Zeile in xdvi300/Makefile überschreiben und der Port wird mit einer Auflösung von 118 erstellt.

5.9. Manualpages

Die Variablen `MAN[1-9LN]` werden automatisch jede Manualpage zur pkg-plist hinzufügen (dies bedeutet, dass Sie Manualpages *nicht* in der pkg-plist auflisten dürfen, lesen Sie bitte Erstellung der PLIST für weitere Details). Sie veranlassen zudem den Installationsabschnitt dazu, die Manualpages zu Komprimieren oder zu Dekomprimieren abhängig vom gesetzten Wert der Variable `NO_MANCOMPRESS` in /etc/make.conf.

Falls Ihr Port versucht verschiedene Namen für Manualpages unter Zuhilfenahme von Symlinks oder Hardlinks zu installieren, müssen Sie die Variable `MLINKS` nutzen, um diese zu identifizieren. Der von Ihrem Port installierte Link wird von bsd.port.mk gelöscht und wieder eingefügt, um sicherzustellen, dass er auf die korrekte Datei zeigt. Jede Manualpage, welche in `MLINKS` aufgeführt ist, darf nicht in der pkg-plist aufgenommen werden.

Falls die Manualpages während der Installation komprimiert werden sollen, müssen Sie die Variable `MANCOMPRESSED` setzen. Diese Variable kann drei Werte annehmen, `yes`, `no` und `maybe`. `yes` bedeutet, dass Manualpages bereits komprimiert installiert sind, bei `no` sind sie es nicht und `maybe` bedeutet, dass die Software bereits den Wert von `NO_MANCOMPRESS` beachtet, damit bsd.port.mk nichts Besonderes auszuführen hat.

`MANCOMPRESSED` wird automatisch auf `yes` gesetzt, wenn `USE_IMAKE` vorgegeben ist und gleichzeitig `NO_INSTALL_MANPAGES` nicht. Im umgekehrten Falle ist `MANCOMPRESSED` auf `no` gesetzt. Sie müssen es

nicht explizit angeben, außer die Standardvorgabe ist für Ihren Port nicht passend.

Wenn Ihr Port den man tree irgendwo anders als in der Variable PREFIX verankert, können Sie ihn mit MANPREFIX bestimmen. Sollten zudem Manualpages nur in bestimmten Abschnitten an einem nicht-standardkonformen Platz liegen, wie z.B. bestimmte Perl-Modul-Ports, dann können Sie mittels der Variable MAN_sect_PREFIX (wobei *sect* ein Wert aus 1-9, L oder N ist) individuelle Pfade zu den Manualpages festlegen.

Wenn Ihre Manualpages in sprachspezifische Unterverzeichnisse installiert werden, dann bestimmen Sie bitte den Namen der Sprache mit der Variable MANLANG. Der Wert dieser Variable ist mit "" vorgegeben (das bedeutet nur Englisch).

Hier ist ein Beispiel, welches alles zusammenfasst.

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

Dies zeigt an, dass sechs Dateien von diesem Port installiert werden;

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

${MANPREFIX}/man/man8/alt-name.8.gz kann zusätzlich von Ihrem Port installiert werden, oder auch nicht. Unabhängig davon wird ein Symlink erstellt, welcher die Manualpages foo(1) und alt-name(8) einbindet.

Falls nur manche Manualpages übersetzt sind, können Sie einige dynamisch vom MANLANG-Inhalt erzeugte Variablen nutzen:

```
MANLANG=       "" de ja
MAN1=          foo.1
MAN1_EN=       bar.1
MAN3_DE=       baz.3
```

Dies führt zu folgender Liste von Dateien:

```
${MANPREFIX}/man/man1/foo.1.gz
${MANPREFIX}/man/de/man1/foo.1.gz
```

```
${MANPREFIX}/man/ja/man1/foo.1.gz
${MANPREFIX}/man/man1/bar.1.gz
${MANPREFIX}/man/de/man3/baz.3.gz
```

5.10. Info-Dateien

Falls Ihr Paket GNU-Info-Dateien installiert, sollten diese in der INFO-Variablen augelistet sein (ohne das angehängte .info) mit einem Eintrag für jedes Dokument. Von diesen Dateien wird angenommen, dass sie nach PREFIX/INFO_PATH installiert werden. Sie können INFO_PATH ändern, falls Ihr Paket einen anderen Ort vorsieht. Jedoch wird dies nicht empfohlen. Die Einträge enthalten nur den relativen Pfad zu PREFIX/INFO_PATH. Zum Beispiel installiert lang/gcc34 Info-Dateien nach PREFIX/INFO_PATH/gcc34, wobei INFO etwa so aussieht:

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

Entsprechende Installations-/Deinstalltions-Codes werden vor der Paket-Registrierung automatisch der vorläufigen pkg-plist hinzugefügt.

5.11. Makefile-Optionen

Einige größere Applikationen können mit einer Reihe von Konfigurationen, die zusätzliche Funktionalitäten hinzufügen, erstellt werden, falls eine oder mehrere Bibliotheken oder Applikationen verfügbar sind. Dazu gehören die Auswahl von natürlichen Sprachen, GUI versus Kommandozeilen-Versionen oder die Auswahl aus mehreren Datenbank-Programmen. Da nicht alle Nutzer diese Bibliotheken oder Applikationen wollen, stellt das Ports-System hooks (Haken) zur Verfügung, damit der Autor des Ports bestimmen kann, welche Konfiguration erstellt werden soll.

5.11.1. KNOBS (Einstellungen)

WITH_* und WITHOUT_*

Diese Variablen sind entworfen worden, um vom System-Administrator gesetzt zu werden. Es gibt viele, die in ports/KNOBS standardisiert sind.

Benennen Sie Schalter bei der Erstellung eines Ports nicht programmspezifisch. Verwenden Sie zum Beispiel im Avahi-Port WITHOUT_MDNS anstelle von WITHOUT_AVAHI_MDNS.



Sie sollten nicht annehmen, dass ein WITH_* notwendigerweise eine korrespondierende WITHOUT_*-Variable hat oder umgekehrt. Im Allgemeinen wird diese Vorgabe einfach unterstellt.



Falls nicht anderweitig festgelegt, werden diese Variablen nur dahingehend überprüft, ob sie gesetzt sind oder nicht - nicht darauf, ob sie auf bestimmte Werte wie YES oder NO gesetzt sind.

Tabelle 3. Häufige WITH_* und WITHOUT_*-Variablen

Variable	Bedeutung
WITHOUT_NLS	Falls gesetzt, bedeutet sie, dass eine Internationalisierung nicht benötigt wird, was Kompilierzeit sparen kann. Als Vorgabe wird Internationalisierung gebraucht.
WITH_OPENSSL_BASE	Nutze die Version von OpenSSL aus dem Basissystem.
WITH_OPENSSL_PORT	Installiert die Version von OpenSSL aus security/openssl, auch wenn das Basissystem auf aktuellem Stand ist.
WITHOUT_X11	Falls der Port mit oder ohne Unterstützung für X erstellt werden kann, dann sollte normalerweise mit X-Unterstützung erstellt werden. Falls die Variable gesetzt ist, soll die Version ohne X-Unterstützung erstellt werden.

Benennung von Knobs (Einstellungen)

Um die Anzahl der Knobs niedrig zu halten und zum Vorteil des Anwenders, wird empfohlen, dass Porter ähnliche Namen für Knobs verwenden. Eine Liste der beliebtesten Knobs kann in der KNOBS-Datei eingesehen werden.

Knob-Namen sollten wiederspiegeln, was der Knob bedeutet und was er bewirkt. Wenn ein Port einen lib-Präfix im PORTNAME hat, dann soll das lib-Präfix im Knob-Namen entfallen.

5.11.2. OPTIONS

Hintergrund

Die OPTIONS-Variable gibt dem Nutzer, der diesen Port installiert, einen Dialog mit auswählbaren Optionen und speichert diese in /var/db/ports/portname/options. Bei der nächsten Neuerstellung des Ports werden diese Einstellungen wieder verwandt. Sie werden sich niemals mehr an all die zwanzig WITH_* und WITHOUT_*-Optionen erinnern müssen, die Sie benutzt haben, um diesen Port zu erstellen!

Wenn der Anwender make config benutzt (oder ein make build das erste Mal laufen lässt) wird das Framework auf /var/db/ports/portname/options die Einstellungen prüfen. Falls die Datei nicht existiert, werden die Werte von OPTIONS genutzt, um eine Dialogbox zu erzeugen, in welcher die Optionen an- oder abgeschaltet werden können. Dann wird die options-Datei gespeichert und die ausgewählten Variablen werden bei der Erstellung des Ports benutzt.

Falls eine neue Version des Ports OPTIONS hinzufügt, wird der Dialog mit den gespeicherten Werten dem Nutzer angezeigt.

Benutzen Sie make showconfig, um die gespeicherte Konfiguration zu betrachten. Benutzen Sie make rmconfig, um die gespeicherte Konfiguration zu Löschen.

Syntax

Die Syntax für die OPTIONS-Variable lautet:

```
OPTIONS=    OPTION    "descriptive text" default ...
```

Der Wert als Vorgabe ist entweder ON oder OFF. Wiederholungen dieser drei Felder sind erlaubt.

OPTIONS-Definitionen müssen vor der Einbindung von bsd.port.options.mk erscheinen. Die WITH_* und WITHOUT_*-Variablen können nur nach der Einbindung von bsd.port.options.mk getestet werden. bsd.port.pre.mk kann auch stattdessen eingebunden werden und wird immer noch von vielen Ports eingebunden, die vor der Einführung von bsd.port.options.mk erstellt wurden. Jedoch wirken manche Variablen nicht wie gewohnt nach der Einbindung von bsd.port.pre.mk, typischerweise USE_*-Optionen.

Beispiel 8. Einfache Anwendung von OPTIONS

```
OPTIONS=      FOO "Enable option foo" On \
              BAR "Support feature bar" Off

.include <bsd.port.options.mk>

.if defined(WITHOUT_FOO)
CONFIGURE_ARGS+=    --without-foo
.else
CONFIGURE_ARGS+=    --with-foo
.endif

.if defined(WITH_BAR)
RUN_DEPENDS+=    bar:${PORTSDIR}/bar/bar
.endif

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

Beispiel 9. Veraltete Anwendung von OPTIONS

```
OPTIONS=      FOO "Enable option foo" On

.include <bsd.port.pre.mk>

.if defined(WITHOUT_FOO)
CONFIGURE_ARGS+=    --without-foo
.else
CONFIGURE_ARGS+=    --with-foo
.endif
```

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

5.11.3. Automatische Aktivierung von Funktionen

Wenn Sie ein GNU-Konfigurationsskript benutzen, sollten Sie ein Auge darauf werfen, welche Funktionen durch die automatische Erkennung aktiviert werden. Schalten Sie Funktionen, die Sie nicht möchten, ausdrücklich durch Verwendung von `--without-xxx` oder `--disable-xxx` in der Variable `CONFIGURE_ARGS` einzeln ab.

Beispiel 10. Falsche Behandlung einer Option

```
.if defined(WITH_FOO)
LIB_DEPENDS+=        foo.0:${PORTSDIR}/devel/foo
CONFIGURE_ARGS+=    --enable-foo
.endif
```

Stellen Sie sich vor im obigen Beispiel ist eine Bibliothek libfoo auf dem System installiert. Der Nutzer will nicht, dass diese Applikation libfoo benutzt, also hat er die Option auf "off" im `make config`-Dialog umgestellt. Aber das Konfigurationsskript der Applikation hat erkannt, dass die Bibliothek auf dem System vorhanden ist und fügt ihre Funktionen in die Binärdatei ein. Falls der Nutzer sich nun entschliesst libfoo von seinem System zu entfernen, dann wird das Ports-System nicht protestieren (es wurde keine Abhängigkeit von libfoo eingetragen), aber die Applikation bricht ab.

Beispiel 11. Korrekte Behandlung einer Option

```
.if defined(WITH_FOO)
LIB_DEPENDS+=        foo.0:${PORTSDIR}/devel/foo
CONFIGURE_ARGS+=    --enable-foo
.else
CONFIGURE_ARGS+=    --disable-foo
.endif
```

Im zweiten Beispiel wird die Bibliothek libfoo explizit abgeschaltet. Das Konfigurationsskript aktiviert die entsprechenden Funktionen nicht in der Applikation trotz der Anwesenheit der Bibliothek auf dem System.

5.12. Die Festlegung des Arbeitsverzeichnisses

Jeder Port wird extrahiert in ein Arbeitsverzeichnis, welches beschreibbar sein muss. Das Ports-System gibt als Standard vor, dass die `DISTFILES` in einem Verzeichnis namens `${DISTNAME}` entpackt werden. Mit anderen Worten, wenn Sie:

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

festgelegt haben, dann enthalten die Distributions-Dateien des Ports ein Verzeichnis auf oberster Ebene, foo-1.0, und der Rest der Dateien befindet sich unter diesem Verzeichnis.

Es gibt eine Reihe von Variablen, die Sie überschreiben können, falls dies nicht der Fall sein sollte.

5.12.1. WRKSRC

Diese Variable listet den Namen des Verzeichnisses, welches erstellt wird, wenn die Distfiles der Applikation extrahiert werden. Wenn unser vorheriges Beispiel in einem Verzeichnis namens foo (und nicht foo-1.0) extrahiert wurde, würden Sie schreiben:

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

oder möglicherweise

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

5.12.2. NO_WRKSUBDIR

Wenn der Port überhaupt nicht in einem Unterverzeichnis extrahiert wird, sollten Sie dies mit dem Setzen von NO_WRKSUBDIR anzeigen.

```
NO_WRKSUBDIR= yes
```

5.13. Konfliktbehandlung

Es gibt drei verschiedene Variablen, um einen Konflikt zwischen Paketen und Ports zu dokumentieren: CONFLICTS, CONFLICTS_INSTALL sowie CONFLICTS_BUILD.



CONFLICTS setzt automatisch die Variable IGNORE, die ausführlicher in Einen Port durch BROKEN - FORBIDDEN oder IGNORE als nicht installierbar markieren beschrieben wird.

Beim Entfernen eines von mehreren in Konflikt stehenden Ports ist es ratsam, die CONFLICTS -Einträge in den anderen Ports für einige Monate beizubehalten, um Nutzer zu unterstützen, die ihre Ports nur sporadisch aktualisieren.

5.13.1. CONFLICTS_INSTALL

Falls Ihr Paket nicht mit anderen Paketen koexistieren kann (wegen Dateikonflikten, Laufzeit-Inkompatibilitäten usw.), führen Sie bitte die anderen Paketnamen in der Variable

CONFLICTS_INSTALL auf. Sie können hier Shell-Globs wie * und ? verwenden. Paketnamen sollten in der gleichen Weise aufgezählt werden, wie sie in /var/db/pkg auftauchen. Bitte stellen Sie sicher, dass CONFLICTS nicht mit dem Paket des Ports selbst übereinstimmt, da ansonsten das Erzwingen der Installation durch FORCE_PKG_REGISTER nicht länger funktionieren wird.

5.13.2. CONFLICTS_BUILD

Wenn Ihr Port nicht gebaut werden kann, wenn ein bestimmter Port bereits installiert ist, geben Sie diesen in der Variable CONFLICTS_BUILD an. Sie können hier Shell-Globs wie * und ? verwenden. Paketnamen sollten in der gleichen Weise aufgezählt werden, wie sie in /var/db/pkg auftauchen. Die CONFLICTS_BUILD-Prüfung erfolgt vor dem Bau des Ports. Baukonflikte werden im erzeugten Paket nicht verzeichnet.

5.13.3. CONFLICTS

Wenn Ihr Port nicht gebaut werden kann, wenn ein bestimmter Port bereits installiert ist und das aus dem Port erzeugte Paket nicht mit dem anderen Paket koexistieren kann, geben Sie das andere Paket in der Variable CONFLICTS an. Sie können hier Shell-Globs wie * und ? verwenden. Paketnamen sollten in der gleichen Weise aufgezählt werden, wie sie in /var/db/pkg auftauchen. Bitte stellen Sie sicher, dass CONFLICTS_INSTALL nicht mit dem Paket des Ports selbst übereinstimmt, da ansonsten das Erzwingen der Installation durch FORCE_PKG_REGISTER nicht länger funktionieren wird. Die CONFLICTS-Prüfung erfolgt vor dem Bau des Ports und vor der Installation des gebauten Ports.

5.14. Installation von Dateien

5.14.1. INSTALL_* macros

Nutzen Sie die Makros in bsd.port.mk, um korrekte Modi und Eigentümer von Dateien in Ihren *-install-Targets sicherzustellen.

- INSTALL_PROGRAM ist ein Befehl, um binäre Binärdateien zu installieren.
- INSTALL_SCRIPT ist ein Befehl, um ausführbare Skripte zu installieren.
- INSTALL_LIB ist ein Befehl zur Installation Shared-Libraries.
- INSTALL_KLD ist ein Befehl, mit dem Kernelmodule installiert werden können. Einige Architekturen haben Probleme mit stripped-Modulen. Daher sollten Sie diesen Befehl anstelle von INSTALL_PROGRAM verwenden.
- INSTALL_DATA ist ein Befehl, um gemeinsam nutzbare Daten zu installieren.
- INSTALL_MAN ist ein Befehl, um Manualpages oder andere Dokumentation zu installieren (es wird nichts komprimiert).

Das sind grundsätzlich alle install-Befehle mit ihren passenden Flags.

5.14.2. Zerlegen von Binärdateien und Shared-Libraries

Zerlegen Sie keine Binärdateien manuell, wenn Sie es nicht müssen. Alle Binaries sollten gestripped

werden; allerdings vermag das INSTALL_PROGRAM-Makro gleichzeitig eine Binärdatei zu installieren und zu strippen (beachten Sie den nächsten Abschnitt). Das Makro INSTALL_LIB erledigt das gleiche für Shared-Libraries.

Wenn Sie eine Datei strippen müssen, aber weder das INSTALL_PROGRAM- noch das INSTALL_LIB-Makro nutzen wollen, dann kann ${STRIP_CMD} Ihr Programm strippen. Dies wird typischerweise innerhalb des post-install-Targets gemacht. Zum Beispiel:

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

Nutzen Sie file(1) für die installierte Applikation, um zu überprüfen, ob eine Binärdatei gestripped ist oder nicht. Wenn es nicht meldet not stripped, dann ist es bereits gestripped. Zudem wird strip(1) nicht ein bereits gestripptes Programm nochmals versuchen zu strippen, sondern wird stattdessen einfach sauber beenden.

5.14.3. Installation eines ganzen Verzeichnisbaums inklusive Dateien

Manchmal muss man eine große Zahl von Dateien unter Erhalt ihrer hierarchischen Struktur installieren, d.h. Kopieraktionen über einen ganzen Verzeichnisbaum von WRKSRC zu einem Zielverzeichnis unter PREFIX.

Für diesen Fall gibt es zwei Makros. Der Vorteil der Nutzung dieser Makros anstatt cp ist, dass sie korrekte Besitzer und Berechtigungen auf den Zieldateien garantieren. Das erste Makro, COPYTREE_BIN, wird alle installierten Dateien ausführbar markieren und damit passend für die Installation in PREFIX/bin vorbereiten. Das zweite Makro, COPYTREE_SHARE, setzt keine Ausführungsberechtigungen auf Dateien und ist daher geeignet für die Installation von Dateien im Target von PREFIX/share.

```
post-install:
    ${MKDIR} ${EXAMPLESDIR}
    (cd ${WRKSRC}/examples/ && ${COPYTREE_SHARE} \* ${EXAMPLESDIR})
```

Dieses Beispiel wird den Inhalt des examples-Verzeichnisses im Distfile des Drittanbieters in das Beispielverzeichnis Ihres Ports kopieren.

```
post-install:
    ${MKDIR} ${DATADIR}/summer
    (cd ${WRKSRC}/temperatures/ && ${COPYTREE_SHARE} "June July August"
${DATADIR}/summer/)
```

Und dieses Beispiel wird die Daten der Sommermonate in das summer-Unterverzeichnis eines DATADIR installieren.

Zusätzliche find-Argumente können mit dem dritten Argument an die COPYTREE_*-Makros übergeben werden. Um zum Beispiel alle Dateien aus dem 1. Beispiel ohne die Makefiles zu

installieren, kann man folgenden Befehl benutzen.

```
post-install:
    ${MKDIR} ${EXAMPLESDIR}
    (cd ${WRKSRC}/examples/ && \
    ${COPYTREE_SHARE} \* ${EXAMPLESDIR} "! -name Makefile")
```

Beachten Sie bitte, dass diese Makros die installierten Dateien nicht zur pkg-plist hinzufügen, Sie müssen sie immer noch selbst auflisten.

5.14.4. Installation zusätzlicher Dokumentation

Falls Ihre Software zusätzlich zu den üblichen Manualpages und Info-Seiten weitere Dokumentation hat und Sie diese für nützlich halten, dann installieren Sie sie unter PREFIX/shared/doc. Dies kann wie vorstehend im Target des `post-install` geschehen.

Legen Sie ein neues Verzeichnis für Ihren Port an. Das Verzeichnis sollte wiederspiegeln, was der Port ist. Das bedeutet normalerweise `PORTNAME`. Wie auch immer, wenn Sie meinen, der Nutzer möchte verschiedene Versionen des Ports zur gleichen Zeit installiert haben, dann können Sie die gesamte Variable `PKGNAME` nutzen.

Machen Sie die Installation von der Variablen `NOPORTDOCS` abhängig, damit die Nutzer sie in /etc/make.conf abschalten können:

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

Hier einige praktische Variablen und wie sie standardmässig bei Verwendung im Makefile expandiert werden:

- `DATADIR` wird expandiert zu PREFIX/shared/PORTNAME.
- `DATADIR_REL` wird expandiert zu share/PORTNAME.
- `DOCSDIR` wird expandiert zu PREFIX/shared/doc/PORTNAME.
- `DOCSDIR_REL` wird expandiert zu share/doc/PORTNAME.
- `EXAMPLESDIR` wird expandiert zu PREFIX/shared/examples/PORTNAME.
- `EXAMPLESDIR_REL` wird expandiert zu share/examples/PORTNAME.



`NOPORTDOCS` behandelt nur zusätzliche Dokumentation, die in `DOCSDIR` installiert ist. Für normale Manualpages und Info-Seiten wird die Variable benutzt. Dinge, welche in `DATADIR` und `EXAMPLESDIR` installiert werden, legen die Variablen `NOPORTDATA` und `NOPORTEXAMPLES` fest.

Die Variablen werden nach PLIST_SUB exportiert. Ihre Werte erscheinen dort als Pfadnamen relativ zu PREFIX, falls möglich. Das bedeutet, dass share/doc/PORTNAME standardmässig ersetzt wird durch %%DOCSDIR%% in der Packliste usw. (mehr zur Ersetzung durch die pkg-plist finden Sie hier).

Alle installierten Dokumentationsdateien und -Verzeichnisse sollten in der pkg-plist dem %%PORTDOCS%%-Präfix enthalten sein, zum Beispiel:

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

Alternativ zur Auflistung der Dokumentationsdateien in der pkg-plist kann in einem Port auch die Variable PORTDOCS gesetzt werden für eine Liste von Dateien und Shell-Globs, um diese zur endgültigen Packliste hinzuzufügen. Die Namen werden relativ zur Variable DOCSDIR sein. Wenn Sie also einen Port haben, welcher PORTDOCS benutzt, und Sie haben eine vom Standard abweichenden Platz für seine Dokumentation, dann müssen Sie die Variable DOCSDIR entsprechend setzen. Wenn ein Verzeichnis in PORTDOCS aufgeführt ist, oder von einem Shell-Glob dieser Variable abgebildet wird, dann wird der komplette Verzeichnisbaum inklusive Dateien und Verzeichnissen in der endgültigen Packliste aufgenommen. Wenn die Variable NOPORTDOCS gesetzt ist, dann werden die Dateien und Verzeichnisse, die in PORTDOCS aufgelistet sind, nicht installiert und werden auch nicht zur Packliste des Ports hinzugefügt. Wie oben gezeigt bleibt es dem Port selbst überlassen, die Dokumentation in PORTDOCS zu installieren. Ein typisches Beispiel für den Gebrauch von PORTDOCS sieht wie folgt aus:

```
PORTDOCS=       README.* ChangeLog docs/*
```



Die Äquivalente zu PORTDOCS für unter DATADIR und EXAMPLESDIR installierte Dateien sind PORTDATA beziehungsweise PORTEXAMPLES.

Sie können auch pkg-message benutzen, um Meldungen während der Installation anzuzeigen. Lesen Sie diesen Abschnitt über den Gebrauch von pkg-message für weitere Details. Die pkg-message-Datei muss nicht zur pkg-plist hinzugefügt werden.

5.14.5. Unterverzeichnisse mit PREFIX

Lassen Sie den Port die Dateien in die richtigen Unterverzeichnisse von PREFIX verteilen. Einige Ports werfen alles in einen Topf und legen es im Unterverzeichnis mit dem Namen des Ports ab, was falsch ist. Ausserdem legen viele Ports alles ausser Binaries, Header-Dateien und Manualpages in ein Unterverzeichnis von lib, was natürlich auch nicht der BSD-Philosophie entspricht und nicht gut funktioniert. Viele der Dateien sollten in eines der folgenden Verzeichnisse geschoben werden: etc (Konfigurationsdateien), libexec (intern gestartete Binärdateien), sbin (Binärdateien für Superuser/Manager), info (Dokumentation für Info-Browser) oder share (Architektur-unabhängige Dateien). Lesen Sie hierzu hier(7); weitestgehend greifen die Regeln für /usr auch für /usr/local. Die Ausnahme sind Ports, welche mit "news" aus dem USENET arbeiten. In diesem Falle sollte PREFIX/news als Zielort für die Dateien benutzt werden.

Kapitel 6. Besonderheiten

Es gibt einige Dinge mehr, die zu beachten sind, wenn man einen Port erstellt. Dieser Abschnitt erklärt die wichtigsten.

6.1. Shared-Libraries

Wenn Ihr Port eine oder mehrere Shared-Libraries installiert, dann definieren Sie bitte eine `USE_LDCONFIG` make-Variable, die bsd.port.mk anweisen wird, `${LDCONFIG} -m` auf das Verzeichnis, in das die neue Library installiert wird (normalerweise PREFIX/lib), während des `post-install`-Targets anzuwenden, um sie im Shared-Library-Cache zu registrieren. Diese Variable, wenn definiert, wird auch dafür sorgen, dass ein entsprechendes `@exec /sbin/ldconfig -m` und `@unexec /sbin/ldconfig -R` -Paar zu Ihrer pkg-plist-Datei hinzugefügt wird, sodass ein Benutzer, der das Paket installiert, die Bibliothek danach sofort benutzen kann und das System nach deren Deinstallation nicht glaubt, die Bibliothek wäre noch da.

```
USE_LDCONFIG= yes
```

Wenn nötig, können Sie das Standardverzeichnis außer Kraft setzen, indem Sie den `USE_LDCONFIG` Wert auf eine Liste von Verzeichnissen setzen, in die Shared Libraries installiert werden sollen. Wenn Ihr Port z.B. diese Bibliotheken nach PREFIX/lib/foo und PREFIX/lib/bar installiert, könnten Sie folgendes in Ihrem Makefile benutzen:

```
USE_LDCONFIG= ${PREFIX}/lib/foo ${PREFIX}/lib/bar
```

Bitte überprüfen Sie dies genau. Oft ist das überhaupt nicht nötig oder kann durch `-rpath` oder das Setzen von `LD_RUN_PATH` während des Linkens umgangen werden (s. lang/moscow_ml für ein Beispiel), oder durch einen Shell-Wrapper, der `LD_LIBRARY_PATH` setzt, bevor er die Binärdatei ausführt, wie es www/seamonkey tut.

Wenn Sie 32-Bit Libraries auf 64-Bit Systemen installieren, benutzen Sie stattdessen `USE_LDCONFIG32`.

Versuchen Sie Shared-Library-Versionsnummern im libfoo.so.0 Format zu halten. Unser Runtime-Linker kümmert sich nur um die Major (erste) Nummer.

Wenn sich die Major-Library-Versionsnummer während der Aktualisierung zu einer neuen Portversion erhöht, sollte auch die `PORTREVISION` aller Ports, die die Shared-Library linken, erhöht werden, damit diese mit der neuen Version der Bibliothek neu kompiliert werden.

6.2. Ports mit beschränkter Verbreitung

Lizenzen variieren und manche geben Restriktionen vor, wie die Applikation gepackt werden oder ob sie gewinnorientiert verkauft werden kann, usw.



Es liegt in Ihrer Verantwortung als Porter die Lizenzbestimmungen der Software

> zu lesen und sicherzustellen, dass das FreeBSD-Projekt nicht haftbar gemacht wird für Lizenzverletzungen durch Weiterverbreitung des Quelltextes oder kompilierter Binaries über FTP/HTTP oder CD-ROM. Im Zweifelsfall kontaktieren Sie bitte die FreeBSD ports.

In solchen Situationen können die in den folgenden Abschnitten beschriebenen Variablen gesetzt werden.

6.2.1. NO_PACKAGE

Diese Variable zeigt an, dass wir keine binären Pakete dieser Applikation erzeugen dürfen - z.B. wenn die Lizenz die Weiterverteilung von binären Paketen oder Paketen verbietet, die aus verändertem Quelltext erzeugt wurden.

Die DISTFILES des Ports dürfen allerdings frei über FTP/HTTP Mirrors weiterverbreitet werden. Sie dürfen auch auf CD-ROM (oder ähnlichen Medien) weiterverbreitet werden - es sei denn, NO_CDROM ist ebenfalls gesetzt.

NO_PACKAGE sollte auch benutzt werden, wenn das binäre Paket nicht allgemein brauchbar ist und die Applikation immer aus dem Quelltext kompiliert werden sollte. Zum Beispiel, wenn die Applikation konfigurierte Informationen über den Rechner/Installationsort bei der Installation einkompiliert bekommt, setzen Sie NO_PACKAGE.

NO_PACKAGE sollte auf eine Zeichenkette gesetzt werden, die den Grund beschreibt, warum kein Paket erzeugt werden soll.

6.2.2. NO_CDROM

Diese Variable gibt an, dassobwohl wir binäre Pakete erzeugen dürfen - wir weder diese Pakete noch die DISTFILES des Ports auf einer CD-ROM (oder ähnlichen Medien) verkaufen dürfen. Die DISTFILES des Ports dürfen allerdings immer noch auf FTP/HTTP Mirrors.

Wenn diese Variable und auch NO_PACKAGE gesetzt ist, dann werden nur die DISTFILES des Ports erhältlich sein - und das nur mittels FTP/HTTP.

NO_CDROM sollte auf eine Zeichenkette gesetzt werden, die den Grund beschreibt, warum der Port nicht auf CD-ROM weiterverbreitet werden kann. Das sollte z.B. gemacht werden, wenn die Lizenz des Ports nur für "nichtkommerzielle Zwecke" gilt.

6.2.3. NOFETCHFILES

Dateien, die in der Variable NOFETCHFILES aufgelistet sind, sind von keiner der MASTER_SITES abrufbar. Ein Beispiel solch einer Datei ist eine selbige, welche vom Anbieter auf CD-ROM bereitgestellt wird.

Werkzeuge, die das Vorhandensein dieser Dateien auf den MASTER_SITES überprüfen, sollten diese Dateien ignorieren und sie nicht melden.

6.2.4. RESTRICTED

Setzen Sie diese Variable, wenn die Lizenz der Applikation weder das Spiegeln der DISTFILES der Applikation noch das Weiterverbreiten von binären Paketen in jedweder Art erlaubt.

NO_CDROM oder NO_PACKAGE sollten nicht zusammen mit RESTRICTED gesetzt werden, weil letztere Variable die anderen beiden impliziert.

RESTRICTED sollte auf eine Zeichenkette gesetzt werden, die den Grund beschreibt, warum der Port nicht weiterverbreitet werden kann. Typischerweise besagt dies, dass der Port proprietäre Software enthält und der Benutzer die DISTFILES manuell herunterladen muss - möglicherweise erst nachdem er sich für die Software registriert oder die Bedingungen eines Endbenutzer-Lizenzvertrags (EULA) akzeptiert hat.

6.2.5. RESTRICTED_FILES

Wenn RESTRICTED oder NO_CDROM gesetzt ist, ist diese Variable auf ${DISTFILES} ${PATCHFILES} voreingestellt, sonst ist sie leer. Wenn nicht jede dieser Dateien beschränkt ist, dann führen Sie die betroffenen Dateien in dieser Variable auf.

Beachten Sie, dass der Porter für jede aufgeführte Distributionsdatei einen Eintrag zu /usr/ports/LEGAL hinzufügen sollte, der genau beschreibt, was die Beschränkung mit sich bringt.

6.3. Build-Mechanismen

6.3.1. Paralleles Bauen von Ports

Das Ports-Framework von FreeBSD unterstützt das parallele Bauen von Ports, indem es mehrere make-Instanzen ausführt, damit SMP-Systeme ihre gesamte CPU-Rechenleistung ausnützen können und so das Bauen von Ports schneller und effektiver werden kann.

Dies ermöglicht der Parameter -jX an make(1), wenn Code von Drittanbietern kompiliert wird. Leider können nicht alle Ports wirklich gut mit dem Parallelbau umgehen. Deshalb ist es erforderlich, dass dieses Feature explizit durch MAKE_JOBS_SAFE=yes irgendwo unterhalb des Abschnitts für Abhängigkeiten im Makefile aktiviert wird.

Eine weitere Möglichkeit im Umgang mit dieser Option besteht für den Maintainer darin, MAKE_JOBS_UNSAFE=yes zu setzen. Diese Variable wird dann verwendet, wenn ein Port bekannterweise mit -jX nicht gebaut werden kann, der Benutzer jedoch für alle Ports den Mehrprozessorbau durch FORCE_MAKE_JOBS=yes in /etc/make.conf erzwingt.

6.3.2. make, gmake und imake

Wenn Ihr Port GNU make benutzt, dann setzen Sie bitte USE_GMAKE=yes.

Tabelle 4. Port-Variablen im Zusammenhang mit gmake

Variable	Bedeutung
USE_GMAKE	Der Port benötigt gmake für den Build.

Variable	Bedeutung
GMAKE	Der ganze Pfad zu gmake, wenn es nicht im PATH ist.

Wenn Ihr Port eine X-Applikation ist, die Makefile-Dateien aus Imakefile-Dateien mit imake erzeugt, dann setzen Sie USE_IMAKE=yes. Das sorgt dafür, dass die Konfigurationsphase automatisch ein xmkmf -a ausführt. Wenn das Flag -a ein Problem für Ihren Port darstellt, setzen Sie XMKMF=xmkmf. Wenn der Port imake benutzt, aber das install.man-Target nicht versteht, dann sollte NO_INSTALL_MANPAGES=yes gesetzt werden.

Wenn das Makefile im Quelltext Ihres Ports etwas anderes als all als Haupt-Build-Target hat, setzen Sie ALL_TARGET entsprechend. Das Gleiche gilt für install und INSTALL_TARGET.

6.3.3. configure Skript

Wenn Ihr Port ein configure-Skript benutzt, um Makefile-Dateien aus Makefile.in-Dateien zu erzeugen, setzen Sie GNU_CONFIGURE=yes. Wenn Sie dem configure-Skript zusätzliche Argumente übergeben wollen (das Vorgabeargument ist --prefix=${PREFIX} --infodir=${PREFIX}/${INFO_PATH} --mandir=${MANPREFIX}/man --build=${CONFIGURE_TARGET}), setzen Sie diese zusätzlichen Argumente in CONFIGURE_ARGS. Zusätzliche Umgebungsvariablen können überdie Variable CONFIGURE_ENV übergeben werden.

Tabelle 5. Variablen für Ports, die configure benutzen

Variable	Bedeutung
GNU_CONFIGURE	Der Port benutzt ein configure-Skript, um das Bauen vorzubereiten.
HAS_CONFIGURE	Wie GNU_CONFIGURE, nur dass kein Standard-Konfigurations-Target zu CONFIGURE_ARGS hinzugefügt wird.
CONFIGURE_ARGS	Zusätzliche Argumente für das configure-Skript.
CONFIGURE_ENV	Zusätzliche Umgebungsvariablen für die Abarbeitung des configure-Skriptes.
CONFIGURE_TARGET	Ersetzt das Standard-Konfigurations-Target. Vorgabewert ist ${MACHINE_ARCH}-portbld-freebsd${OSREL}.

6.3.4. Benutzung von scons

Wenn Ihr Port SCons benutzt, definieren Sie USE_SCONS=yes.

Tabelle 6. Variablen für Ports, die scons benutzen

Variable	Bedeutung
SCONS_ARGS	Port-spezifische SCons-Argumente, die der SCons-Umgebung übergeben werden.

Variable	Bedeutung
SCONS_BUILDENV	Variablen, die in der System-Umgebung gesetzt werden sollen.
SCONS_ENV	Variablen, die in der SCons-Umgebung gesetzt werden sollen.
SCONS_TARGET	Letztes Argument, das SCons übergeben wird - ähnlich MAKE_TARGET.

Um SConstruct im Quelltext alles, was SCons in SCONS_ENV übergeben wird, respektieren zu lassen (das ist hauptsächlich CC/CXX/CFLAGS/CXXFLAGS), patchen Sie SConstruct, sodass das Build Environment wie folgt konstruiert wird:

```
env = Environment(**ARGUMENTS)
```

Es kann dann mit env.Append und env.Replace modifiziert werden.

6.4. Benutzung von GNU autotools

6.4.1. Einführung

Die verschiedenen GNU autotools stellen einen Abstraktionsmechanismus bereit für das Kompilieren von Software für eine Vielfalt von Betriebssystemen und Maschinenarchitekturen. Innerhalb der Ports-Sammlung kann ein einzelner Port diese Werkzeuge mit Hilfe eines einfachen Konstrukts benutzen:

```
USE_AUTOTOOLS= tool:version[:operation] ...
```

Als dies geschrieben wurde konnte *tool* eins von libtool, libltdl, autoconf, autoheader, automake oder aclocal sein.

version gibt die einzelne Werkzeug-Revision an, die benutzt werden soll (siehe devel/{automake,autoconf,libtool}[0-9]+ für mögliche Versionen).

operation ist eine optionale Angabe, die modifiziert, wie das Werkzeug benutzt wird.

Es können auch mehrere Werkzeuge angegeben werden - entweder durch Angabe aller in einer einzigen Zeile oder durch Benutzung des += Makefile-Konstrukts.

Schliesslich gibt es das spezielle Tool, genannt autotools, das der Einfachheit dient indem es von alle verfügbaren Versionen der Autotools abhängt, was sinnvoll für Cross-Development ist. Dies kann auch erreicht werden, indem man den Port devel/autotools installiert.

6.4.2. libtool

Shared-Libraries, die das GNU Build-System benutzen, verwenden normalerweise libtool, um die

Kompilierung und Installation solcher Bibliotheken anzupassen. Die übliche Praxis ist, eine Kopie von `libtool`, die mit dem Quelltext geliefert wird, zu benutzen. Falls Sie ein externes `libtool` benötigen, können Sie die Version, die von der Ports-Sammlung bereitgestellt wird, benutzen:

```
USE_AUTOTOOLS= libtool:version[:env]
```

Ohne zusätzliche Angaben sagt `libtool:version` dem Build-System, dass es das Konfigurationsskript mit der auf dem System installierten Kopie von `libtool` patchen soll. Die Variable `GNU_CONFIGURE` ist impliziert. Außerdem werden einige make- und shell-Variablen zur weiteren Benutzung durch den Port gesetzt. Für Genaueres siehe bsd.autotools.mk.

Mit der Angabe `:env` wird nur die Umgebung vorbereitet.

Schließlich können optional `LIBTOOLFLAGS` und `LIBTOOLFILES` gesetzt werden, um die häufigsten Argumente und durch `libtool` gepatchten Dateien außer Kraft zu setzen. Die meisten Ports werden das aber nicht brauchen. Für Weiteres siehe bsd.autotools.mk.

6.4.3. `libltdl`

Einige Ports benutzen das `libltdl`-Bibliothekspaket, welches Teil der `libtool`-Suite ist. Der Gebrauch dieser Bibliothek macht nicht automatisch den Gebrauch von `libtool` selbst nötig, deshalb wird ein separates Konstrukt zur Verfügung gestellt.

```
USE_AUTOTOOLS= libltdl:version
```

Im Moment sorgt dies nur für eine `LIB_DEPENDS`-Abhängigkeit von dem entsprechenden `libltdl`-Port und wird zur Vereinfachung zur Verfügung gestellt, um Abhängigkeiten von den Autotools-Ports ausserhalb des `USE_AUTOTOOLS`-Systems zu eliminieren. Es gibt keine weiteren Angaben für dieses Werkzeug.

6.4.4. `autoconf` und `autoheader`

Manche Ports enthalten kein Konfigurationsskript, sondern eine autoconf-Vorlage in der configure.ac-Datei. Sie können die folgenden Zuweisungen benutzen, um `autoconf` das Konfigurationsskript erzeugen zu lassen, und auch `autoheader` Header-Vorlagen zur Benutzung durch das Konfigurationsskript erzeugen zu lassen.

```
USE_AUTOTOOLS=    autoconf:version[:env]
```

und

```
USE_AUTOTOOLS=    autoheader:version
```

welches auch die Benutzung von `autoconf:version` impliziert.

Ähnlich wie bei libtool, bereitet die Angabe des optionalen :env nur die Umgebung für weitere Benutzung vor. Ohne dieses wird der Port auch gepatched und erneut konfiguriert.

Die zusätzlichen optionalen Variablen AUTOCONF_ARGS und AUTOHEADER_ARGS können durch das Makefile des Ports ausser Kraft gesetzt werden, wenn erforderlich. Wie bei den libtool -Äquivalenten werden die meisten Ports dies aber nicht benötigen.

6.4.5. automake und aclocal

Manche Pakete enthalten nur Makefile.am-Dateien. Diese müssen durch automake in Makefile.in -Dateien konvertiert und dann durch configure weiterbearbeitet werden, um schließlich ein Makefile zu erzeugen.

Ähnliches gilt für Pakete, die gelegentlich keine aclocal.m4-Dateien mitliefern, welche ebenfalls zum Erstellen der Software benötigt werden. Diese können durch aclocal erzeugt werden, welches configure.ac oder configure.in durchsucht.

aclocal hat eine ähnliche Beziehung zu automake wie autoheader zu autoconf - beschrieben im vorherigen Abschnitt. aclocal impliziert die Benutzung von automake, also haben wir:

```
USE_AUTOTOOLS=    automake:version[:env]
```

und

```
USE_AUTOTOOLS=    aclocal:version
```

was auch die Benutzung von automake:version impliziert.

Ähnlich wie bei libtool und autoconf, bereitet die optionale Angabe :env nur die Umgebung zur weiteren Benutzung vor. Ohne sie wird der Port erneut konfiguriert.

Wie schon autoconf und autoheader, hat sowohl automake als auch aclocal eine optionale Argument-Variable AUTOMAKE_ARGS bzw. ACLOCAL_ARGS, die durch das Makefile des Ports, falls nötig, außer Kraft gesetzt werden kann.

6.5. Benutzung von GNU gettext

6.5.1. Grundlegende Benutzung

Wenn Ihr Port gettext benötigt, setzen Sie einfach USE_GETTEXT auf yes, und Ihr Port bekommt die Abhängigkeit von devel/gettext. Der Wert von USE_GETTEXT kann auch die benötigte Version der libintl-Bibliothek angeben, der grundlegenden Teil von gettext - jedoch wird von der Benutzung dieser Funktion *dringend abgeraten*: Ihr Port sollte einfach nur mit der aktuellen Version von devel/gettext funktionieren.

Ein ziemlich häufiger Fall ist, dass ein Port gettext und configure benutzt. Normalerweise sollte GNU configure gettext automatisch finden können. Sollte das einmal nicht funktionieren, können

Hinweise über den Ort von `gettext` in `CPPFLAGS` und `LDFLAGS` wie folgt übergeben werden:

```
USE_GETTEXT=    yes
CPPFLAGS+=      -I${LOCALBASE}/include
LDFLAGS+=       -L${LOCALBASE}/lib

GNU_CONFIGURE=  yes
CONFIGURE_ENV=  CPPFLAGS="${CPPFLAGS}" \
            LDFLAGS="${LDFLAGS}"
```

Natürlich kann der Code kompakter sein, wenn es keine weiteren Flags gibt, die `configure` übergeben werden müssen:

```
USE_GETTEXT=    yes
GNU_CONFIGURE=  yes
CONFIGURE_ENV=  CPPFLAGS="-I${LOCALBASE}/include" \
            LDFLAGS="-L${LOCALBASE}/lib"
```

6.5.2. Optionale Benutzung

Manche Softwareprodukte erlauben die Deaktivierung von NLS - z.B. durch Übergeben von `--disable-nls` an `configure`. In diesem Fall sollte Ihr Port `gettext` abhängig vom Status von `WITHOUT_NLS` benutzen. Für Ports mit niedriger bis mittlerer Komplexität können Sie sich auf das folgende Idiom verlassen:

```
GNU_CONFIGURE=          yes

.if !defined(WITHOUT_NLS)
USE_GETTEXT=            yes
PLIST_SUB+=             NLS=""
.else
CONFIGURE_ARGS+=        --disable-nls
PLIST_SUB+=             NLS="@comment "
.endif
```

Der nächste Punkt auf Ihrer Todo-Liste ist dafür zu sorgen, dass die Message-Catalog-Dateien nur bedingt in der Packliste aufgeführt werden. Der Makefile-Teil dieser Aufgabe ist schon durch obiges Idiom erledigt. Das wird im Abschnitt über Fortgeschrittene pkg-plist-Methoden erklärt. Kurz gesagt, jedes Vorkommen von `%%NLS%%` in pkg-plist wird durch “@comment”, wenn NLS abgeschaltet ist, oder durch eine leere Zeichenkette, wenn NLS aktiviert ist, ersetzt. Folglich werden die Zeilen, denen `%%NLS%%` vorangestellt ist, zu reinen Kommentaren in der endgültigen Packliste, wenn NLS abgeschaltet ist; andernfalls wird der Prefix einfach nur ausgelassen. Alles, was Sie jetzt noch machen müssen, ist `%%NLS%%` vor jedem Pfad zu einer Message-Catalog-Datei in pkg-plist einzufügen. Zum Beispiel:

```
%%NLS%%share/locale/fr/LC_MESSAGES/foobar.mo
```

```
%%NLS%%share/locale/no/LC_MESSAGES/foobar.mo
```

In sehr komplexen Fällen müssen Sie eventuell fortgeschrittenere Techniken als die hier vorgestellte benutzen - wie z.B. Dynamische Packlistenerzeugung.

6.5.3. Behandlung von Message-Catalog-Verzeichnissen

Bei der Installation von Message-Catalog-Dateien gibt es einen Punkt zu beachten. Ihr Zielverzeichnis, das unter LOCALBASE/shared/locale liegt, sollte nur selten von Ihrem Port erzeugt und gelöscht werden. Die Verzeichnisse für die gebräuchlichsten Sprachen sind in /etc/mtree/BSD.local.dist aufgelistet; das heisst, sie sind Teil des Systems. Die Verzeichnisse für viele andere Sprachen sind Teil des Ports devel/gettext. Sie wollen vielleicht dessen pkg-plist zur Hand nehmen, um festzustellen, ob Ihr Port eine Message-Catalog-Datei für eine seltene Sprache installiert.

6.6. Die Benutzung von `perl`

Wenn `MASTER_SITES` auf `MASTER_SITE_PERL_CPAN` gesetzt ist, dann ist der bevorzugte Wert von `MASTER_SITE_SUBDIR` der Top-Level-Name der Hierarchie. Zum Beispiel ist der empfohlene Wert für `p5-Module-Name-Module`. Die Top-Level-Hierarchie kann unter cpan.org angeschaut werden. Dies sorgt dafür, dass der Port weiter funktioniert, wenn sich der Autor des Moduls ändert.

Die Ausnahme dieser Regel ist, dass das entsprechende Verzeichnis selber oder das Distfile in diesem Verzeichnis nicht existiert. In solchen Fällen ist die Benutzung der Id des Autors als `MASTER_SITE_SUBDIR` erlaubt.

Jede der Einstellungen unten kann sowohl auf `YES` als auch auf eine Versionszeichenkette wie `5.8.0+` gesetzt werden. Wenn `YES` benutzt wird, bedeutet das, dass der Port mit jeder der unterstützten Perl-Versionen funktioniert. Falls ein Port nur mit einer bestimmten Perl-Version funktioniert, kann darauf mit einer Versionszeichenkette hingewiesen werden, die entweder eine Mindest- (z.B. `5.7.3+`), Maximal- (z.B. `5.8.0-`) oder Absolutversion (z.B. `5.8.3`) festlegt.

Tabelle 7. Variablen für Ports, die `perl` benutzen

Variable	Bedeutung
`USE_PERL5`	Bedeutet, dass der Port `perl 5` zum Erstellen und zum Ausführen benutzt.
`USE_PERL5_BUILD`	Bedeutet, dass der Port `perl 5` zum Erstellen benutzt.
`USE_PERL5_RUN`	Bedeutet, dass der Port `perl 5` zur Laufzeit benutzt.
`PERL`	Der gesamte Pfad zu `perl 5` - entweder im Basissystem oder nachinstalliert über einen Port - ohne die Versionsnummer. Benutzen Sie diese Variable, wenn Sie “#!”-Zeilen in Skripten ersetzen müssen.

Variable	Bedeutung
PERL_CONFIGURE	Perls MakeMaker für die Konfiguration benutzen. Dies impliziert USE_PERL5.
PERL_MODBUILD	Module::Build für configure, build und install benutzen. Dies impliziert PERL_CONFIGURE.



Ports von Perl-Modulen, die keine offizielle Webseite haben, sollen in der WWW-Zeile ihrer pkg-descr-Datei auf cpan.org verlinken. Die bevorzugte URL-Form ist http://search.cpan.org/dist/Module-Name/ (inklusive des Slash am Ende).

6.7. Benutzung von X11

6.7.1. X.Org-Komponenten

Die X11-Implementierung, welche die Ports-Sammlung bereitstellt, ist X.Org. Wenn Ihre Applikation von X-Komponenten abhängt, listen Sie die benötigten Komponenten in USE_XORG auf. Als dies geschrieben wurde, wurden die folgenden Komponenten bereitgestellt:

bigreqsproto compositeproto damageproto dmx dmxproto evieproto fixesproto fontcacheproto fontenc fontsproto fontutil glproto ice inputproto kbproto libfs oldx printproto randrproto recordproto renderproto resourceproto scrnsaverproto sm trapproto videoproto x11 xau xaw xaw6 xaw7 xaw8 xbitmaps xcmiscproto xcomposite xcursor xdamage xdmcp xevie xext xextproto xf86bigfontproto xf86dgaproto xf86driproto xf86miscproto xf86rushproto xf86vidmodeproto xfixes xfont xfontcache xft xi xinerama xineramaproto xkbfile xkbui xmu xmuu xorg-server xp xpm xprintapputil xprintutil xpr oto xproxymngproto xrandr xrender xres xscrnsaver xt xtrans xtrap xtst xv xvmc xxf86dga xxf86misc xxf86vm.

Die aktuelle Liste finden Sie immer in /usr/ports/Mk/bsd.xorg.mk.

Das Mesa Projekt ist ein Versuch, eine freie OpenGL Implementierung bereitzustellen. Sie können eine Abhängigkeit von verschiedenen Komponenten diese Projektes in der Variable USE_GL spezifizieren. ouml;gliche Optionen sind: glut, glu, glw, glew, gl und linux. Für Abwärtskompatibilität gilt der Wert yes als glu.

Beispiel 12. Beispiel für USE_XORG

```
USE_XORG=   xrender xft xkbfile xt xaw
USE_GL=     glu
```

Viele Ports definieren USE_XLIB, was dafür sorgt, dass der Port von allen (rund 50) Bibliotheken abhängt. Diese Variable existiert, um Abwärtskompatibilität sicherzustellen (sie stammt noch aus der Zeit vor dem modularem X.Org), und sollte bei neuen Ports nicht mehr benutzt werden.

Tabelle 8. Variablen für Ports, die X benutzen

USE_XLIB	Der Port benutzt die X-Bibliotheken. Soll nicht mehr verwendet werden - benutzen Sie stattdessen eine Liste von Komponenten in USE_XORG.
USE_X_PREFIX	Soll nicht mehr benutzt werden, ist jetzt äquivalent zu USE_XLIB und kann einfach durch letzteres ersetzt werden.
USE_IMAKE	Der Port benutzt imake. Impliziert USE_X_PREFIX.
XMKMF	Ist auf den Pfad zu xmkmf gesetzt, wenn nicht in PATH. Vorgabe ist xmkmf -a.

Tabelle 9. Variablen bei Abhängigkeit von einzelnen Teilen von X11

X_IMAKE_PORT	Ein Port, der imake und einige andere Werkzeuge, die zum Erstellen von X11 benutzt werden, bereitstellt.
X_LIBRARIES_PORT	Ein Port, der die X11-Bibliotheken bereitstellt.
X_CLIENTS_PORT	Ein Port, der X11-Clients bereitstellt.
X_SERVER_PORT	Ein Port, der den X11-Server bereitstellt.
X_FONTSERVER_PORT	Ein Port, der den Fontserver bereitstellt.
X_PRINTSERVER_PORT	Ein Port, der den Printserver bereitstellt.
X_VFBSERVER_PORT	Ein Port, der den virtuellen Framebuffer-Server bereitstellt.
X_NESTSERVER_PORT	Ein Port, der einen nested X-Server bereitstellt.
X_FONTS_ENCODINGS_PORT	Ein Port, der Kodierungen für Schriftarten bereitstellt.
X_FONTS_MISC_PORT	Ein Port, der verschiedene Bitmap-Schriftarten bereitstellt.
X_FONTS_100DPI_PORT	Ein Port, der 100dpi Bitmap-Schriftarten bereitstellt.
X_FONTS_75DPI_PORT	Ein Port, der 75dpi Bitmap-Schriftarten bereitstellt.
X_FONTS_CYRILLIC_PORT	Ein Port, der kyrillische Bitmap-Schriftarten bereitstellt.
X_FONTS_TTF_PORT	Ein Port, der TrueType®-Schriftarten bereitstellt.
X_FONTS_TYPE1_PORT	Ein Port, der Type1-Schriftarten bereitstellt.
X_MANUALS_PORT	Ein Port, der entwicklerorientierte Manualpages bereitstellt.

Beispiel 13. Benutzung von X11-bezogenen Variablen in einem Port

```
# Port benutzt X11-Bibliotheken und hängt vom Font-Server sowie
# von kyrillischen Schriftarten ab.
RUN_DEPENDS=   ${LOCALBASE}/bin/xfs:${X_FONTSERVER_PORT} \

${LOCALBASE}/lib/X11/fonts/cyrillic/crox1c.pcf.gz:${X_FONTS_CYRILLIC_PORT}

USE_XORG=      x11 xpm
```

6.7.2. Ports, die Motif benötigen

Wenn Ihr Port eine Motif-Bibliothek benötigt, definieren Sie `USE_MOTIF` im Makefile. Die Standard-Motif-Implementierung ist x11-toolkits/open-motif. Benutzer können stattdessen x11-toolkits/lesstif wählen, indem Sie die `WANT_LESSTIF`-Variable setzen.

Die Variable `MOTIFLIB` wird von bsd.port.mk auf die entsprechende Motif-Bibliothek gesetzt. Bitte patchen Sie den Quelltext Ihres Ports, sodass er überall `${MOTIFLIB}` benutzt, wo die Motif-Bibliothek im Original Makefile oder Imakefile referenziert wird.

Es gibt zwei verbreitete Fälle:

- Wenn sich der Port in seinem Makefile oder Imakefile auf die Motif-Bibliothek als `-lXm` bezieht, ersetzen Sie das einfach durch `${MOTIFLIB}`.
- Wenn der Port in seinem Imakefile`XmClientLibs` benutzt, ersetzen Sie das durch `${MOTIFLIB} ${XTOOLLIB} ${XLIB}`.

Anmerkung: `MOTIFLIB` expandiert (normalerweise) zu `-L/usr/X11R6/lib -lXm` oder `/usr/X11R6/lib/libXm.a` - d.h. Sie müssen kein `-L` oder `-l` davor einfügen.

6.7.3. X11 Schriftarten

Wenn Ihr Port Schriftarten für das X-Window-System installiert, legen Sie diese nach LOCALBASE/lib/X11/fonts/local.

6.7.4. Erzeugen eines künstlichen `DISPLAY` durch Xvfb

Manche Applikationen benötigen ein funktionierendes X11-Display, damit die Kompilierung funktioniert. Das stellt für Systeme, die ohne Display laufen, ein Problem dar. Wenn die folgende Variable benutzt wird, startet die Bauumgebung den virtuellen Framebuffer-X-Server, und ein funktionierendes `DISPLAY` wird dem Build übergeben.

```
USE_DISPLAY=  yes
```

6.7.5. Desktop-Einträge

Desktop-Einträge (Freedesktop Standard) können in Ihrem Port einfach über die DESKTOP_ENTRIES -Variable erzeugt werden. Diese Einträge erscheinen dann im Applikationsmenü von standardkonformen Desktop-Umgebungen wie GNOME oder KDE. Die .desktop-Datei wird dann automatisch erzeugt, installiert und der pkg-plist hinzugefügt. Die Syntax ist:

```
DESKTOP_ENTRIES=  "NAME" "COMMENT" "ICON" "COMMAND" "CATEGORY" StartupNotify
```

Die Liste der möglichen Kategorien ist auf der Freedesktop Webseite abrufbar. StartupNotify zeigt an, ob die Applikation den Status in Umgebungen, die Startup-Notifications kennen, löschen wird.

Beispiel:

```
DESKTOP_ENTRIES=  "ToME" "Roguelike game based on JRR Tolkien's work" \
              "${DATADIR}/xtra/graf/tome-128.png" \
              "tome -v -g" "Application;Game;RolePlaying;" \
              false
```

6.8. Benutzung von GNOME

Das FreeBSD/GNOME-Projekt benutzt seine eigene Gruppe von Variablen, um zu definieren, welche GNOME-Komponenten ein bestimmter Port benutzt. Eine umfassende Liste dieser Variablen existiert innerhalb der Webseite des FreeBSD/GNOME-Projektes.

6.9. Benutzung von Qt

6.9.1. Ports, die Qt benötigen

Tabelle 10. Variablen für Ports, die Qt benötigen

USE_QT_VER	Der Port benutzt das Qt-Toolkit. Mögliche Werte sind 3 und 4; diese spezifizieren die Major Version von Qt, die benutzt werden soll. Entsprechende Parameter werden an das configure-Skript und make übergeben.
QT_PREFIX	Enthält den Pfad, wohin Qt installiert ist (nur lesbare Variable).
MOC	Enthält den Pfad von moc (nur lesbare Variable). Voreingestellt entsprechend des USE_QT_VER -Werts.
QTCPPFLAGS	Zusätzliche Compiler-Flags, die über CONFIGURE_ENV an das Qt-Toolkit übergeben werden. Voreingestellt entsprechend des USE_QT_VER-Wertes.

QTCFGLIBS	Zusätzliche Bibliotheken, die über CONFIGURE_ENV für das Qt-Toolkit gelinkt werden sollen. Voreingestellt entsprechend des USE_QT_VER -Wertes.
QTNONSTANDARD	Änderungen von CONFIGURE_ENV, CONFIGURE_ARGS und MAKE_ENV sollen unterdrückt werden.

Tabelle 11. Zusätzliche Variablen für Ports, die Qt 4.xi benutzen

QT_COMPONENTS	Spezifiziert Tool- und Bibliothek-Abhängigkeiten für Qt4. Siehe unten für Details.
UIC	Enthält den Pfad von uic (nur lesbare Variable). Voreingestellt entsprechend des USE_QT_VER -Wertes.
QMAKE	Enthält den Pfad von qmake (nur lesbare Variable). Voreingestellt entsprechend des USE_QT_VER-Wertes.
QMAKESPEC	Enthält den Pfad der Konfigurationsdatei für qmake (nur lesbare Variable). Voreingestellt entsprechend des USE_QT_VER-Wertes.

Wenn USE_QT_VER gesetzt ist, werden dem configure-Skript einige nützliche Einstellungen übergeben:

```
CONFIGURE_ARGS+= --with-qt-includes=${QT_PREFIX}/include \
             --with-qt-libraries=${QT_PREFIX}/lib \
             --with-extra-libs=${LOCALBASE}/lib \
             --with-extra-includes=${LOCALBASE}/include
CONFIGURE_ENV+=  MOC="${MOC}" CPPFLAGS="${CPPFLAGS} ${QTCPPFLAGS}" LIBS="${QTCFGLIBS}"
\
             QTDIR="${QT_PREFIX}" KDEDIR="${KDE_PREFIX}"
```

Wenn USE_QT_VER auf 4 gesetzt ist, werden auch die folgenden Einstellungen übergeben:

```
CONFIGURE_ENV+= UIC="${UIC}" QMAKE="${QMAKE}" QMAKESPEC="${QMAKESPEC}"
MAKE_ENV+=      QMAKESPEC="${QMAKESPEC}"
```

6.9.2. Komponentenauswahl (nur bei Qt 4.x)

Wenn USE_QT_VER auf 4 gesetzt ist, können individuelle Qt4-Tool- und Bibliotheksabhängigkeiten in der Variable QT_COMPONENTS angegeben werden. An jede Komponente kann _build oder _run als Suffix angehängt werden, was eine Abhängigkeit zur Build- bzw. Laufzeit angibt. Ohne Suffix gilt die Abhängigkeit sowohl zur Build- als auch zur Laufzeit. Bibliothekskomponenten sollten normalerweise ohne Suffix angegeben werden, Tool-Komponenten mit _build und Plugin-Komponenten mit _run. Die gebräuchlichsten Komponenten werden im Folgenden angegeben (alle

verfügbaren Komponenten sind in `_QT_COMPONENTS_ALL` in /usr/ports/Mk/bsd.qt.mk aufgelistet):

Tabelle 12. Verfügbare Qt4-Bibliothekskomponenten

Name	Beschreibung
`corelib`	Kern-Bibliothek (kann weggelassen werden- es sei denn, der Port benutzt nichts außer `corelib`)
`gui`	Graphische Benutzeroberflächen-Bibliothek
`network`	Netzwerk-Bibliothek
`opengl`	OpenGL-Bibliothek
`qt3support`	Qt3-Kompatibilitäts-Bibliothek
`qtestlib`	Modultest-Bibliothek
`script`	Skript-Bibliothek
`sql`	SQL-Bibliothek
`xml`	XML-Bibliothek

Sie können herausfinden, welche Bibliotheken die Applikation benötigt, indem Sie nach erfolgreicher Kompilierung `ldd` auf die Hauptbinärdatei anwenden.

Tabelle 13. Verfügbare Qt4-Tool-Komponenten

Name	Beschreibung
`moc`	meta object compiler (wird zum Build fast jeder Qt-Applikation benötigt)
`qmake`	Makefile-Generator / Build-Werkzeug
`rcc`	Resource-Compiler (wird benötigt, falls die Applikation **.rc** oder.qrc Dateien enthält)
`uic`	User-Interface-Compiler (wird benötigt, falls die Applikation von Qt-Designer erzeugte *.ui Dateien enthält - gilt für praktisch jede Qt-Applikation mit einer GUI)

Tabelle 14. Verfügbare Qt4-Plugin-Komponenten

Name	Beschreibung
`iconengines`	SVG-Icon-Engine Plugin (wenn die Applikation SVG-Icons mitliefert)
`imageformats`	Bildformatplugins für GIF, JPEG, MNG und SVG (wenn die Applikation Bilddateien mitliefert)

Beispiel 14. Qt4-Komponenten auswählen

In diesem Beispiel benutzt die portierte Applikation die Qt4 GUI-Bibliothek, die Qt4-Core-Bibliothek, alle Qt4-Codeerzeugungstools und Qt4's Makefile Generator. Da die GUI-Bibliothek

eine Abhängigkeit von der Core-Bibliothek impliziert, muss corelib nicht angegeben werden. Die Qt4-Codeerzeugungstools moc, uic und rcc, sowie der Makefile Generator qmake werden nur für den Build benötigt, deshalb bekommen die den Suffix `_build`:

```
USE_QT_VER=    4
QT_COMPONENTS= gui moc_build qmake_build rcc_build uic_build
```

6.9.3. Zusätzliche Besonderheiten

Wenn die Applikation keine configure Datei, sondern eine .pro Datei hat, können Sie das Folgende benutzen:

```
HAS_CONFIGURE=    yes

do-configure:
    @cd ${WRKSRC} && ${SETENV} ${CONFIGURE_ENV} \
            ${QMAKE} -unix PREFIX=${PREFIX} texmaker.pro
```

Beachten Sie die Ähnlichkeit mit der `qmake`-Zeile im mitgelieferten BUILD.sh-Skript. Die Übergabe von `CONFIGURE_ENV` stellt sicher, dass `qmake` die `QMAKESPEC`-Variable übergeben bekommt, ohne die es nicht funktioniert. `qmake` erzeugt Standard-Makefiles, sodass es nicht nötig ist ein eigenes neues `build`-Target zu schreiben.

Qt-Applikationen sind oft so geschrieben, dass sie plattformübergreifend sind, und oft ist X11/Unix nicht die Plattform, auf der sie entwickelt werden. Das sorgt oft für bestimmte fehlende Kleinigkeiten wie z.B.:

- *Fehlende zusätzliche Include-Pfade.* Viele Applikationen kommen mit System-Tray-Icon Support- unterlassen es aber Includes oder Bibliotheken in den X11 Verzeichnissen zu suchen. Sie können `qmake` über die Kommandozeile sagen, es soll Verzeichnisse zu den Include- und Bibliotheks-Suchpfaden hinzufügen - z.B.:

  ```
  ${QMAKE} -unix PREFIX=${PREFIX} INCLUDEPATH+=${LOCALBASE}/include \
      LIBS+=-L${LOCALBASE}/lib sillyapp.pro
  ```

- *Falsche Installations-Pfade.* Manchmal werden Daten wie Icons oder .desktop-Dateien per Vorgabe in Verzeichnisse installiert, die nicht von XDG-kompatiblen Applikationen durchsucht werden. editors/texmaker ist hierfür ein Beispiel- siehe patch-texmaker.pro im files-Verzeichnis dieses Ports als eine Vorlage, die zeigt, wie man dies direkt in der Qmake Projektdatei löst.

6.10. Benutzung von KDE

6.10.1. Variablen-Definitionen (KDE 3)

Tabelle 15. Variablen für Ports, die KDE 3 benutzen

USE_KDELIBS_VER	Der Port benutzt KDE-Bibliotheken. Die Variable spezifiziert die Major Version von KDE, die benutzt werden soll, und impliziert USE_QT_VER der entsprechenden Version. Der einzig mögliche Wert ist 3.
USE_KDEBASE_VER	Der Port benutzt die KDE-Base. Die Variable spezifiziert die Major Version von KDE, die benutzt werden soll, und impliziert USE_QT_VER der entsprechenden Version. Der einzig mögliche Wert ist 3.

6.10.2. Variablen-Definitionen (KDE 4)

Falls Ihre Anwendung von KDE 4 abhängt, weisen Sie USE_KDE4 eine Liste mit benötigten Komponenten zu. Die am häufigsten gebrauchten sind unten aufgelistet (_USE_KDE4_ALL in /usr/ports/Mk/bsd.kde4.mk enthält stets die aktuelle Liste):

Tabelle 16. Verfügbare KDE 4-Komponenten

Name	Beschreibung
akonadi	Personal Information Management (PIM)-Speicherdienst
automoc4	Lässt den Port das Bauwerkzeug automoc4 verwenden.
kdebase	Grundlegende KDE-Anwendungen (Konqueror, Dolphin, Konsole)
kdeexp	Experimentelle KDE-Bibliotheken (mit einer API, die als non-stable eingestuft ist)
kdehier	Stellt allgemeine KDE-Verzeichnisse bereit
kdelibs	Die grundlegenden KDE-Bibliotheken
kdeprefix	Falls in der Liste vorhanden, wird der Port unter ${KDE4_PREFIX} statt ${LOCALBASE} installiert
pimlibs	PIM-Bibliotheken
workspace	Anwendungen und Bibliotheken, welche die Desktopumgebung gestalten (Plasma, KWin)

KDE 4-Ports werden unter ${KDE4_PREFIX}, zur Zeit /usr/local/kde4, installiert, um Konflikte mit KDE 3-Ports zu verhindern. Dies wird durch Auflisten der Komponente kdeprefix erreicht, welche die standardmäßig gesetzte Variable PREFIX überschreibt. Die Ports übernehmen jedoch, jeden über die Umgebungsvariable MAKEFLAGS oder make-Parameter festgelegten Wert für PREFIX.

Es könnte bei der Installation von KDE 4-Ports zu Konflikten mit KDE 3-Ports kommen, sodass diese

bei aktivierter kdeprefix-Komponente unter ${KDE4_PREFIX} installiert werden. Der Standardwert von KDE4_PREFIX ist zur Zeit /usr/local/kde4. Es ist auch möglich, KDE 4-Ports unter einem angepassten PREFIX zu installieren. Wenn PREFIX als MAKEFLAGS-Umgebungsvariable oder als make-Parameter gesetzt wird, überschreibt dies den von kdeprefix festgelegten Wert.

Beispiel 15. USE_KDE4-Beispiel

Dies ist ein einfaches Beispiel für einen KDE 4-Port. USE_CMAKE weist den Port an, CMake, ein unter KDE 4-Projekten weit verbreitetes Konfigurationswerkzeug, zu verwenden. USE_KDE4 legt die Abhängigkeit von KDE-Bibliotheken und die Verwendung von automoc4 während der Kompilierung fest. Mit Hilfe des configure-Protokolls können die KDE-Komponenten und andere Abhängigkeiten festgestellt werden. USE_KDE4 impliziert USE_QT_VER nicht. Falls der Port Qt 4-Komponenten benötigt, sollten USE_QT_VER gesetzt und verlangte Komponenten festgelegt werden.

```
USE_CMAKE=     yes
USE_KDE4=      automoc4 kdelibs kdeprefix
USE_QT_VER=    4
QT_COMPONENTS= qmake_build moc_build rcc_build uic_build
```

6.11. Benutzung von Java

6.11.1. Variablen-Definitionen

Wenn Ihr Port ein Java™ Development Kit (JDK™) benötigt, entweder zum Bauen, zur Laufzeit oder sogar, um das Distfile auszupacken, dann sollten Sie USE_JAVA setzen.

Es gibt mehrere JDKs in der Ports-Sammlung- von verschiedenen Anbietern und in verschiedenen Versionen. Wenn Ihr Port eine bestimmte dieser Versionen benötigt, können Sie definieren welche. Die aktuelle Version ist java/jdk16.

Tabelle 17. Variablen, die von Ports, die Java benutzen, gesetzt werden müssen

Variable	Bedeutung
USE_JAVA	Sollte definiert sein, damit die übrigen Variablen irgendeinen Effekt haben.
JAVA_VERSION	Durch Leerzeichen getrennte Liste von geeigneten Java-Versionen für den Port. Ein optionales "" ermöglicht die Angabe eines Bereiches von Versionen (mögliche Werte: 1.5[] 1.6[] 1.7[]).
JAVA_OS	Durch Leerzeichen getrennte Liste von geeigneten JDK-Port-Betriebssystemen für den Port. (erlaubte Werte: native linux).

Variable	Bedeutung
JAVA_VENDOR	Durch Leerzeichen getrennte Liste von geeigneten JDK-Port-Anbietern für den Port. (erlaubte Werte: freebsd bsdjava sun openjdk).
JAVA_BUILD	Bedeutet, falls gesetzt, dass der ausgewählte JDK-Port zu den Build-Abhängigkeiten des Ports hinzugefügt werden soll.
JAVA_RUN	Bedeutet, falls gesetzt, dass der ausgewählte JDK-Port zu den Laufzeit-Abhängigkeiten des Ports hinzugefügt werden soll.
JAVA_EXTRACT	Bedeutet, falls gesetzt, dass der ausgewählte JDK-Port zu den Extract-Abhängigkeiten des Ports hinzugefügt werden soll.

Das Folgende ist eine Liste aller Variablen, die ein Port bekommt, nachdem er USE_JAVA gesetzt hat:

Tabelle 18. Bereitgestellte Variablen für Ports, die Java benutzen

Variable	Wert
JAVA_PORT	Der Name des JDK-Ports (z.B. 'java/diablo-jdk16').
JAVA_PORT_VERSION	Die volle Version des JDK Ports (z.B. '1.6.0'). Wenn Sie nur die ersten beiden Stellen dieser Versionsnummer benötigen, benutzen Sie ${JAVA_PORT_VERSION:C/^([0-9])\.([0-9])(.*)$/\1.\2/}.
JAVA_PORT_OS	Das vom JDK-Port benutzte Betriebssystem (z.B. 'native').
JAVA_PORT_VENDOR	Der Anbieter des JDK-Ports (z.B. 'freebsd').
JAVA_PORT_OS_DESCRIPTION	Beschreibung des vom JDK-Port benutzten Betriebssystems (z.B. 'Native').
JAVA_PORT_VENDOR_DESCRIPTION	Beschreibung des Anbieters des JDK-Ports (z.B. 'FreeBSD Foundation').
JAVA_HOME	Pfad zum Installationsverzeichnis des JDK (z.B. '/usr/local/diablo-jdk1.6.0').
JAVAC	Pfad zum Java-Compiler, der benutzt werden soll (z.B. '/usr/local/diablo-jdk1.6.0/bin/javac'.
JAR	Pfad zum jar-Werkzeug, das benutzt werden soll (z.B."/usr/local/diablo-jdk1.6.0/bin/jar oder '/usr/local/bin/fastjar').
APPLETVIEWER	Pfad zum appletviewer-Werkzeug (z.B. '/usr/local/diablo-jdk1.6.0/bin/appletviewer').

Variable	Wert
JAVA	Pfad zur java Binärdatei. Benutzen Sie dies, um Java-Programme auszuführen (z.B.'/usr/local/diablo-jdk1.6.0/bin/java').
JAVADOC	Pfad zum javadoc-Werkzeug.
JAVAH	Pfad zum javah-Programm.
JAVAP	Pfad zum javap-Programm.
JAVA_KEYTOOL	Pfad zum keytool-Werkzeug.
JAVA_N2A	Pfad zum native2ascii-Werkzeug.
JAVA_POLICYTOOL	Pfad zum policytool Programm.
JAVA_SERIALVER	Pfad zum serialver-Werkzeug.
RMIC	Pfad zum RMI Stub/Skeleton-Generator, rmic.
RMIREGISTRY	Pfad zum RMI Registry-Werkzeug, rmiregistry.
RMID	Pfad zum RMI Daemon rmid.
JAVA_CLASSES	Pfad zum Archiv, das die JDK-Klassendateien enthält, ${JAVA_HOME}/jre/lib/rt.jar.

Sie können das java-debug make-Target benutzen, um Information zum Debuggen Ihres Ports zu erhalten. Es wird die Werte vieler der obenangegebenen Variablen anzeigen.

Zusätzlich sind die folgenden Konstanten definiert, damit alle Java-Ports auf eine konsistente Art installiert werden können:

Tabelle 19. Konstanten, die für Ports, welche Java benutzen, definiert sind

Konstante	Wert
JAVASHAREDIR	Das Basis-Verzeichnis für alles, was mit Java zusammenhängt. Standardmäßig ${PREFIX}/shared/java.
JAVAJARDIR	Das Verzeichnis, wohin JAR-Dateien installiert werden sollen. Standardmäßig ${JAVASHAREDIR}/classes.
JAVALIBDIR	Das Verzeichnis, in dem JAR-Dateien, die von anderen Ports installiert wurden, liegen. Standardmäßig ${LOCALBASE}/shared/java/classes.

Die entsprechenden Einträge sind sowohl in PLIST_SUB (dokumentiert in Änderungen an pkg-plist mit Hilfe von make-Variablen) als auch in SUB_LIST definiert.

6.11.2. Kompilieren mit Ant

Wenn der Port mit Apache Ant kompiliert werden soll, muss er USE_ANT setzen. Ant wird dann als

das sub-make-Kommando betrachtet. Wenn kein `do-build`-Target vom Port definiert ist, wird eine Standardvorgabe benutzt, die einfach Ant entsprechend `MAKE_ENV`, `MAKE_ARGS` und `ALL_TARGET` aufruft. Das ähnelt dem `USE_GMAKE`-Mechanismus, der in Build-Mechanismen dokumentiert ist.

6.11.3. Optimales Verfahren

Wenn Sie eine Java-Bibliothek portieren, sollte Ihr Port die JAR-Datei(en) in ${JAVAJARDIR} installieren, und alles andere unter ${JAVASHAREDIR}/${PORTNAME} (ausgenommen die Dokumentation - siehe unten). Um die Größe der Packlistendatei zu reduzieren, können die JAR-Datei(en) direkt im Makefile angegeben werden. Benutzen Sie einfach die folgende Anweisung (wobei myport.jar der Name der JAR-Datei ist, die als Teil des Ports installiert wird):

```
PLIST_FILES+= %%JAVAJARDIR%%/myport.jar
```

Beim Portieren einer Java-Applikation installiert der Port normalerweise alles unter einem einzigen Verzeichnis (inklusive seiner JAR-Abhängigkeiten). Die Benutzung von ${JAVASHAREDIR}/${PORTNAME} wird in dieser Beziehung dringend empfohlen. Es liegt im Entscheidungsbereich des Portierenden, ob der Port die zusätzlichen JAR-Abhängigkeiten unter diesem Verzeichnis installieren oder direkt die schon installierten (aus ${JAVAJARDIR}) benutzen soll.

Unabhängig von der Art Ihres Ports (Bibliothek oder Applikation), sollte die zusätzliche Dokumentation an die gleiche Stelle installiert werden wie bei jedem anderen Port auch. Das JavaDoc-Werkzeug ist dafür bekannt einen unterschiedlichen Satz von Dateien abhängig von der Version des benutzten JDKs zu erstellen. Für Ports, die nicht die Benutzung eines bestimmten JDKs vorgeben, ist es deshalb eine komplexe Aufgabe die Packliste (pkg-plist) festzulegen. Dies ist ein Grund, warum dringend angeraten wird, das `PORTDOCS`-Makro zu benutzen. Außerdem, selbst wenn Sie den Satz von Dateien, den `javadoc` erzeugen wird, voraussagen können, die Größe der resultierenden pkg-plist befürwortet die Benutzung von `PORTDOCS`.

Der Vorgabewert für `DATADIR` ist ${PREFIX}/shared/${PORTNAME}. Es ist eine gute Idee, `DATADIR` für Java-Ports stattdessen auf ${JAVASHAREDIR}/${PORTNAME} zu setzen. In der Tat wird `DATADIR` automatisch zu `PLIST_SUB` (dokumentiert in Änderungen an pkg-plist mit Hilfe von make-Variablen) hinzugefügt, d.h. Sie können `%%DATADIR%%` direkt in pkg-plist benutzen.

Zu der Frage, ob Java-Ports aus dem Quelltext gebaut werden, oder direkt bereitgestellte binäre Distributionen benutzt werden sollten, gab es, als dies geschrieben wurde, keine definierte Richtlinie. Allerdings ermutigen Mitglieder des FreeBSD Java-Projekts Porter dazu, Ihre Ports aus dem Quelltext kompilieren zu lassen, wann immer dies kein Problem darstellt.

Alle Eigenschaften, die in diesem Abschnitt präsentiert wurden sind in bsd.java.mk implementiert. Sollten Sie jemals der Meinung sein, dass Ihr Port ausgefeiltere Java-Unterstützung benötigt, schauen Sie bitte erst in das bsd.java.mk CVS Log, weil es normalerweise immer etwas Zeit braucht bis die neuesten Eigenschaften dokumentiert sind. Wenn Sie glauben, dass der fehlende Support auch für viele andere Java Ports nützlich sein könnte, wenden Sie sich bitte an die freebsd-java.

Obwohl es eine `java`-Kategorie für Fehlerberichte gibt, bezieht sich diese auf die JDK-Portierungsbemühungen des FreeBSD Java-Projektes. Deshalb sollten Sie Ihren Java-Port in der

ports-Kategorie einreichen wie bei jeden anderen Port auch - es sei denn, die Angelegenheit, die Sie zu klären versuchen, steht in Zusammenhang entweder mit einer JDK-Implementierung oder bsd.java.mk.

Gleichermaßen gibt es eine definierte Richtlinie für die CATEGORIES eines Java-Ports, die in Kategorisierung erklärt wird.

6.12. Webanwendungen, Apache und PHP

6.12.1. Apache

Tabelle 20. Variablen für Ports, die Apache verwenden

USE_APACHE	Der Port benötigt Apache. Mögliche Werte: yes (beliebige Version), 1.3, 2.0, 2.2, 2.0+, etc. - Standard ist Version 1.3.
WITH_APACHE2	Der Port benötigt Apache 2.0. Ist diese Variable nicht gesetzt, so benötigt der Port Apache 1.3. Diese Variable ist veraltet und sollte nicht mehr verwendet werden.
APXS	Vollständiger Pfad zu der apxs Binärdatei. Die Variable kann neu gesetzt werden.
HTTPD	Vollständiger Pfad zu der httpd Binärdatei. Die Variable kann neu gesetzt werden.
APACHE_VERSION	Beinhaltet die Versionsnummer des aktuell installierten Apache (nur lesbare Variable). Diese Variable ist nach Einbinden der Datei bsd.port.pre.mk verfügbar. Mögliche Werte: 13, 20, 22.
APACHEMODDIR	Verzeichnis der Apache-Module. Diese Variable wird automatisch in pkg-plist ersetzt.
APACHEINCLUDEDIR	Verzeichnis der Apache Header-Dateien. Diese Variable wird automatisch in pkg-plist ersetzt.
APACHEETCDIR	Verzeichnis der Apache-Konfigurationsdateien. Diese Variable wird automatisch in pkg-plist ersetzt.

Tabelle 21. Nützliche Variablen für Ports von Apache-Modulen

MODULENAME	Name des Moduls. Standardwert ist PORTNAME. Beispiel: mod_hello
SHORTMODNAME	Der gekürzte Name des Moduls. Standardmäßig wird der Wert von MODULENAME übernommen. Beispiel: hello

AP_FAST_BUILD	Verwende apxs zum Kompilieren und Installieren des Moduls.
AP_GENPLIST	Eine pkg-plist wird automatisch erzeugt.
AP_INC	Verzeichnis für zusätzliche Header-Dateien, die beim Kompilieren mitverwendet werden.
AP_LIB	Verzeichnis für zusätzliche Bibliothek-Dateien, welche beim Kompilieren mitverwendet werden.
AP_EXTRAS	Zusätzliche Flags für apxs.

6.12.2. Webanwendungen

Webanwendungen sollten nach PREFIX/www/programmname installiert werden. Der Einfachheit halber ist dieser Pfad sowohl im Makefile als auch in pkg-plist als WWWDIR verfügbar. Der relative Pfad PREFIX ist hingegen im Makefile durch die Variable WWWDIR_REL festgelegt.

Der Benutzername und die Benutzergruppe, mit deren Rechte Webanwendungen laufen, sind in WWWOWN und WWWGRP festgelegt. Standardwert ist bei beiden www. Falls ein Port mit anderen Rechten gestartet werden soll, so sollte die Anweisung WWWOWN?= myuser verwendet werden. Dies vereinfacht dem Benutzer eine Anpassung dieser Werte.

Falls die Webanwendung nicht explizit Apache benötigt, so sollte dieser auch nicht als Abhängigkeit des Ports aufgeführt werden. Dadurch bleibt es dem Benutzer überlassen Apache oder einen anderen Webserver zu verwenden.

6.12.3. PHP

Tabelle 22. Variablen für Ports, die PHP verwenden

USE_PHP	Der Port benötigt PHP. Der Wert yes bewirkt eine Abhängigkeit des Ports von PHP. Es kann auch eine Liste der benötigten PHP-Erweiterungen angegeben werden. Beispiel: pcre xml gettext
DEFAULT_PHP_VER	Legt die Version von PHP fest, die standardmäßig installiert wird, falls noch kein PHP vorhanden ist. Standardwert ist 4. Mögliche Werte sind: 4,5
IGNORE_WITH_PHP	Der Port funktioniert nicht mit der angegebenen Version von PHP. Mögliche Werte: 4, 5
USE_PHPIZE	Der Port wird als PHP-Erweiterung gebaut.
USE_PHPEXT	Der Port wird wie eine PHP-Erweiterung behandelt - Installation und Eintragung in die PHP-Registry für Erweiterungen.
USE_PHP_BUILD	Setzt PHP als build-Anhängigkeit.

WANT_PHP_CLI	Benötigt die Kommandozeilen-Version von PHP.
WANT_PHP_CGI	Benötigt die CGI-Version von PHP.
WANT_PHP_MOD	Benötigt das Apache-Modul von PHP.
WANT_PHP_SCR	Benötigt die Kommandozeilen- oder die CGI-Version von PHP.
WANT_PHP_WEB	Benötigt das Apache-Modul oder die CGI-Version von PHP.

6.12.4. PEAR Module

Das Portieren von PEAR-Modulen ist sehr einfach.

Mit Hilfe der Variablen FILES, TESTS, DATA, SQLS, SCRIPTFILES, DOCS und EXAMPLES können die zu installierenden Dateien angegeben werden. Alle aufgeführten Dateien werden automatisch in die jeweiligen Verzeichnisse installiert und der Datei pkg-plist hinzugefügt.

Die Datei ${PORTSDIR}/devel/pear/bsd.pear.mk muss am Ende des Makefiles eingebunden werden.

Beispiel 16. Beispiel eines Makefiles für eine PEAR Klasse

```
PORTNAME=       Date
PORTVERSION=    1.4.3
CATEGORIES=     devel www pear

MAINTAINER=     example@domain.com
COMMENT=        PEAR Date and Time Zone Classes

BUILD_DEPENDS=  ${PEARDIR}/PEAR.php:${PORTSDIR}/devel/pear-PEAR
RUN_DEPENDS=    ${BUILD_DEPENDS}

FILES=          Date.php Date/Calc.php Date/Human.php Date/Span.php     \
            Date/TimeZone.php
TESTS=          test_calc.php test_date_methods_span.php testunit.php   \
            testunit_date.php testunit_date_span.php wknotest.txt   \
            bug674.php bug727_1.php bug727_2.php bug727_3.php       \
            bug727_4.php bug967.php weeksinmonth_4_monday.txt       \
            weeksinmonth_4_sunday.txt weeksinmonth_rdm_monday.txt   \
            weeksinmonth_rdm_sunday.txt
DOCS=           TODO
_DOCSDIR=       .

.include <bsd.port.pre.mk>
.include "${PORTSDIR}/devel/pear/bsd.pear.mk"
.include <bsd.port.post.mk>
```

6.13. Python benutzen

Die Ports unterstützen parallele Installationen mehrerer Python-Versionen. Ports sollten sicherstellen, dass der richtige `python`-Interpreter verwendet wird - entsprechend der durch den Benutzer definierbaren Variable `PYTHON_VERSION`. Häufig bedeutet dies, dass der Pfad zum `python` -Interpreter durch den Wert der Variablen `PYTHON_CMD` ersetzt werden muss.

Ports, die Dateien unter `PYTHON_SITELIBDIR` installieren, sollten `pyXY-` als Präfix des Paketnamens haben, sodass in deren Paketname die zugehörige Python Version aufgeführt wird.

```
PKGNAMEPREFIX= ${PYTHON_PKGNAMEPREFIX}
```

Tabelle 23. Nützliche Variablen für Ports, die Python verwenden

`USE_PYTHON`	Der Port benötigt Python. Die minimal benötigte Version kann durch Werte wie `2.3+` angegeben werden. Bereiche von Versionsnummern können durch Angabe der minimalen und maximalen Versionsnummer, getrennt durch einen Gedankenstrich, festgelegt werden, z.B.: `2.1-2.3`
`USE_PYDISTUTILS`	Verwende Python-distutils zum Konfigurieren, Kompilieren und Installieren. Dies ist erforderlich, falls der Port eine setup.py-Datei beinhaltet. Dadurch werden die `do-build` und `do-install`-Ziele und eventuell auch das `do-configure`-Ziel übergangen, falls `GNU_CONFIGURE` nicht definiert ist.
`PYTHON_PKGNAMEPREFIX`	Wird als `PKGNAMEPREFIX` verwendet, um Pakete für unterschiedliche Python-Versionen zu trennen. Beispiel: `py24-`
`PYTHON_SITELIBDIR`	Verzeichnis des site-Pakete Baums, der das Installationsverzeichnis von Python (üblicherweise `LOCALBASE`) beinhaltet. Die `PYTHON_SITELIBDIR`-Variable kann sehr nützlich bei der Installation von Python-Modulen sein.
`PYTHONPREFIX_SITELIBDIR`	Die präfix-freie Variante von `PYTHON_SITELIBDIR`. Benutzen Sie immer `%%PYTHON_SITELIBDIR%%` in pkg-plist, wenn möglich. Der Standardwert von `%%PYTHON_SITELIBDIR%%` ist `lib/python%%PYTHON_VERSION%%/site-packages`
`PYTHON_CMD`	Kommandozeilen-Interpreter für Python mit Versionsnummer.
`PYNUMERIC`	Liste der Abhängigkeiten für numerische Erweiterungen.

PYNUMPY	Liste der Abhängigkeiten für die neue numerische Erweiterung numpy. (PYNUMERIC ist vom Anbieter als veraltet deklariert)
PYXML	Liste der Abhängigkeiten für XML-Erweiterungen (wird ab Python 2.0 nicht mehr benötigt, da im Basispaket enthalten).
USE_TWISTED	Setzt die Abhängigkeit des Ports von twistedCore. Die Liste der erforderlichen Komponenten kann als Wert spezifiziert werden. Beispiel: `web lore pair flow`
USE_ZOPE	Setzt Zope, eine Plattform für Webanwendungen, als Abhängigkeit des Ports. Setzt die Versionsabhängigkeit von Python auf 2.3. Setzt ZOPEBASEDIR auf das Verzeichnis, in welches Zope installiert wurde.

Eine vollständige Liste aller verfügbaren Variablen ist in /usr/ports/Mk/bsd.python.mk zu finden.

6.14. Benutzung von Tcl/Tk

Die Ports-Sammlung unterstützt die parallele Installation mehrerer Tcl/Tk-Versionen. Ports sollten mindestens die vorgegebene Tcl/Tk-Version oder höher zu unterstützen versuchen anhand der Variablen USE_TCL und USE_TK. Es ist möglich, die gewünschte Version von tcl mit der Variable WITH_TCL_VER vorzuschreiben.

Tabelle 24. Äußerst nützliche Variablen für Ports, die Tcl/Tk benutzen

USE_TCL	Der Port benötigt die Tcl-Bibliothek (nicht die Shell). Eine notwendige Mindestversion kann mit Werten wie 84+ angegeben werden. Einzelne nicht unterstützte Versionen können mit der Variable INVALID_TCL_VER festgelegt werden.
USE_TCL_BUILD	Der Port benötigt Tcl nur während der Zeit, in der er gebaut wird.
USE_TCL_WRAPPER	Ports, welche zwar die Tcl-Shell, aber nicht eine bestimmte Version von tclsh verlangen, sollten diese neue Variable verwenden. Ein Wrapperskript für tclsh wird auf dem System installiert. Der Benutzer kann festlegen, welche tcl-Shell gewünscht ist bzw. verwendet werden soll.
WITH_TCL_VER	Benutzerdefinierte Variable, welche die gewünschte Tcl-Version bestimmt.
_PORTNAME__WITH_TCL_VER	Gleich wie WITH_TCL_VER, nur portspezifisch.
USE_TCL_THREADS	Fordere threadfähiges Tcl/Tk.

USE_TK	Der Port benötigt die Tk-Bibliothek (nicht die Wish-Shell). Impliziert USE_TCL mit dem gleichen Wert. Für weitere Informationen siehe die Beschreibung der Variable USE_TCL.
USE_TK_BUILD	Analog zur Variable USE_TCL_BUILD.
USE_TK_WRAPPER	Analog zur Variable USE_TCL_WRAPPER.
WITH_TK_VER	Analog zur Variable WITH_TCL_VER und impliziert WITH_TCL_VER mit dem gleichen Wert.

Eine vollständige Liste der zur Verfügung stehenden Variablen befindet sich in /usr/ports/Mk/bsd.tcl.mk.

6.15. Emacs benutzen

Dieser Abschnitt muss noch geschrieben werden.

6.16. Ruby benutzen

Tabelle 25. Nützliche Variablen für Ports, die Ruby verwenden

Variable	Description
USE_RUBY	Der Port benötigt Ruby.
USE_RUBY_EXTCONF	Der Port verwendet extconf.rb für die Konfiguration.
USE_RUBY_SETUP	Der Port verwendet setup.rb für die Konfiguration.
RUBY_SETUP	Legt den alternativen Namen von setup.rb fest. Üblich ist der Wert install.rb.

Die folgende Tabelle listet ausgewählte Variablen auf, die Portautoren über die Port-Infrastruktur zur Verfügung stehen. Diese Variablen sollten für die Installation von Dateien in die entsprechenden Verzeichnisse verwendet werden. Sie sollten in pkg-plist so häufig wie möglich verwendet und in einem Port nicht neu definiert werden.

Tabelle 26. Ausgewählte read-only-Variablen für Ports, die Ruby verwenden

Variable	Beschreibung	Beispiel
RUBY_PKGNAMEPREFIX	Wird als PKGNAMEPREFIX verwendet, um Pakete für verschiedene Versionen von Ruby zu unterscheiden.	ruby18-
RUBY_VERSION	Vollständige Version von Ruby in der Form x.y.z.	1.8.2

Variable	Beschreibung	Beispiel
RUBY_SITELIBDIR	Installationsverzeichnis der von der Rechnerarchitektur unabhängigen Bibliotheken.	/usr/local/lib/ruby/site_ruby/1.8
RUBY_SITEARCHLIBDIR	Installationsverzeichnis der von der Rechnerarchitektur abhängigen Bibliotheken.	/usr/local/lib/ruby/site_ruby/1.8/amd64-freebsd6
RUBY_MODDOCDIR	Installationsverzeichnis für die Dokumentation der Module.	/usr/local/shared/doc/ruby18/patsy
RUBY_MODEXAMPLESDIR	Installationsverzeichnis für die Beispiele der Module.	/usr/local/shared/examples/ruby18/patsy

Eine vollständige Liste der verfügbarenVariablen kann in /usr/ports/Mk/bsd.ruby.mk eingesehen werden.

6.17. SDL verwenden

Die Variable USE_SDL wird für die automatische Konfiguration der Abhängigkeiten für Ports benutzt, die auf SDL basierende Bibliotheken wie devel/sdl12 und x11-toolkits/sdl_gui verwenden.

Die folgenden SDL-Bibliotheken sind derzeit bekannt:

- 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

Falls ein Port z.B. von net/sdl_net und audio/sdl_mixer abhängt, so wäre die Syntax:

```
USE_SDL=        net mixer
```

Die Abhängigkeit von devel/sdl12, die durch net/sdl_net und audio/sdl_mixer entsteht, wird automatisch zum Port hinzugefügt.

Falls USE_SDL im Port verwendet wird, so wird automatisch:

- die Abhängigkeit von sdl12-config zu BUILD_DEPENDS hinzugefügt

- die Variable SDL_CONFIG zu CONFIGURE_ENV hinzugefügt
- die Abhängigkeit der ausgewählten Bibliotheken zu LIB_DEPENDS hinzugefügt

Um zu überprüfen, ob die SDL-Bibliotheken verfügbar sind, kann die Variable WANT_SDL verwendet werden:

```
WANT_SDL=yes

.include <bsd.port.pre.mk>

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

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

6.18. wxWidgets verwenden

Dieser Abschnitt beschreibt den Status der wxWidgets-Bibliotheken in den Ports und deren Einbindung in das Ports-System.

6.18.1. Einführung

Es gibt viele Probleme bei der gleichzeitigen Verwendung unterschiedlicher Versionen von wxWidgets-Bibliotheken (Dateien unterschiedlicher wxWidgets-Versionen haben denselben Dateinamen). In den Ports wurde das Problem dadurch gelöst, dass jede Version unter einem eigenen Namen installiert wird, der die Versionsnummer als Suffix beinhaltet.

Der offensichtliche Nachteil dabei ist, dass jede Anwendung so verändert werden muss, dass sie die erwartete Version vorfindet. Die meisten solcher Anwendungen benutzen das wx-config-Skript, um die benötigten Compiler- und Linkerflags zu erhalten. Dieses Skript hat für jede verfügbare Version einen anderen Namen. Die meisten Anwendungen beachten eine Umgebungsvariable oder ein Argument beim configure-Skript, um das gewünschte wx-config-Skript festzulegen. Ansonsten müssen sie gepatcht werden.

6.18.2. Auswahl der Version

Um festzulegen, welche Version der wxWidgets verwendet werden soll, gibt es zwei Variablen (falls nur eine der beiden definiert wird, so wird die andere auf einen Standardwert gesetzt):

Tabelle 27. Variablen, um die wxWidgets-Version festzulegen

Variable	Beschreibung	Standardwert
USE_WX	Liste der Versionen, die der Port verwenden kann	Alle verfügbaren Versionen
USE_WX_NOT	Liste der Versionen, die der Port nicht verwenden kann	Nichts

Es folgt eine Liste an möglichen wxWidgets-Versionen und deren zugehöriger Port:

Tabelle 28. Verfügbare wxWidgets-Versionen

Version	Port
2.4	x11-toolkits/wxgtk24
2.6	x11-toolkits/wxgtk26
2.8	x11-toolkits/wxgtk28



Ab Version 2.5 werden auch Versionen in Unicode unterstützt und über einen Unterport mit dem Suffix -unicode installiert. Dies kann aber auch über Variablen gehandhabt werden (siehe Unicode).

Die Variablen in Variablen, um die wxWidgets-Version festzulegen können auf einen oder mehrere (durch Leerzeichen getrennt) der folgenden Werte gesetzt werden:

Tabelle 29. Spezifikationen der wxWidgets-Versionen

Beschreibung	Beispiel
Einzelne Version	2.4
Aufsteigende Versionsnummern	2.4+
Absteigende Versionsnummern	2.6-
Versionsinterval (muss aufsteigend sein)	2.4-2.6

Desweiteren gibt es Variablen, über die eine bevorzugte Version festgelegt werden kann. Die Versionen können als Liste angegeben werden, wobei die Reihenfolge der Priorisierung entspricht.

Tabelle 30. Variablen zur Festlegung der bevorzugten wxWidgets-Version

Name	Bestimmt für
WANT_WX_VER	den Port
WITH_WX_VER	den Benutzer

6.18.3. Komponentenauswahl

Desweiteren gibt es Anwendungen, die nicht direkt wxWidgets-Bibliotheken sind, aber trotzdem mit diesen zusammenhängen. Diese Anwendungen können über die Variable WX_COMPS festgelegt werden. Die folgenden Komponenten sind verfügbar:

Tabelle 31. Verfügbare wxWidgets-Komponenten

Name	Beschreibung	Versionsbeschränkungen
wx	Hauptbibliothek	Nichts
contrib	Beigesteuerte Bibliothek	Nichts
python	wxPython (Python-Bindungen)	2.4-2.6
mozilla	wxMozilla	2.4

Name	Beschreibung	Versionsbeschränkungen
svg	wxSVG	2.6

Der Typ der Abhängigkeit kann für jede Komponente durch hinzufügen eines Suffix (durch Strichpunkt getrennt) festgelegt werden. Falls der Typ nicht angegeben wird, wird ein Standardwert verwendet (siehe Standardtypen der wxWidgets-Abhängigkeiten). Die folgenden Typen sind verfügbar:

Tabelle 32. Verfügbare Typen von wxWidgets-Abhängigkeiten

Name	Beschreibung
build	Komponente wird zum Bau benötigt - äquivalent zu BUILD_DEPENDS
run	Komponente wird zum Ausführen benötigt - äquivalent zu RUN_DEPENDS
lib	Komponente wird zum Bau und Ausführen benötigt - äquivalent zu LIB_DEPENDS

Die Standardwerte für die einzelnen Komponenten sind in der folgenden Tabelle aufgeführt:

Tabelle 33. Standardtypen der wxWidgets-Abhängigkeiten

Komponente	Typ der Abhängigkeit
wx	lib
contrib	lib
python	run
mozilla	lib
svg	lib

Beispiel 17. Auswahl von wxWidgets-Komponenten

Der folgende Ausschnitt entspricht einem Port, der die wxWidgets-Version 2.4 und die zugehörigen Bibliotheken verwendet.

```
USE_WX=       2.4
WX_COMPS=     wx contrib
```

6.18.4. Unicode

Die wxWidgets-Bibliotheken unterstützen Unicode seit der Version 2.5. In den Ports sind beide Versionen verfügbar und können über die folgenden Variablen ausgewählt werden:

Tabelle 34. Variablen, um Unicode in den wxWidgets-Versionen auszuwählen

Variable	Beschreibung	Bestimmt für
WX_UNICODE	Der Port funktioniert *ausschließlich* mit der Unicode-Version	den Port
WANT_UNICODE	Der Port funktioniert in beiden Versionen - bevorzugt wird jedoch Unicode	den Port
WITH_UNICODE	Der Port verwendet die Unicode-Version	den Benutzer
WITHOUT_UNICODE	Der Port verwendet, falls unterstützt, die normale Version (falls WX_UNICODE nicht definiert ist)	den Benutzer



Die Variable WX_UNICODE darf nicht bei Ports benutzt werden, die sowohl die Version mit als auch ohne Unterstützung für Unicode verwenden können. Falls der Port standardmäßig Unterstützung für Unicode bieten soll, verwenden Sie WANT_UNICODE stattdessen.

6.18.5. Feststellen der installierten Version

Um eine bereits installierte Version zu finden, muss WANT_WX definiert werden. Falls diese Variable nicht auf eine bestimmte Versionsnummer gesetzt wird, werden die Komponenten einen Suffix mit der Versionsnummer tragen. Die Variable HAVE_WX wird gesetzt, falls eine installierte Version vorgefunden wurde.

Beispiel 18. Installierte wxWidgets-Versionen und -Komponenten feststellen

Der folgende Ausschnitt kann in einem Port verwendet werden, der wxWidgets verwendet, falls es installiert ist, oder falls eine Option dafür ausgewählt wurde.

```
WANT_WX=        yes

.include <bsd.port.pre.mk>

.if defined(WITH_WX) || ${HAVE_WX:Mwx-2.4} != ""
USE_WX=         2.4
CONFIGURE_ARGS+=--enable-wx
.endif
```

Der folgende Ausschnitt kann verwendet werden, um die Unterstützung für wxPython zusätzlich zu der von wxWidgets zu aktivieren (beide in Version 2.6), wenn das installiert ist, oder die Option ausgewählt wurde.

```
USE_WX=         2.6
```

```
WX_COMPS=       wx
WANT_WX=        2.6

.include <bsd.port.pre.mk>

.if defined(WITH_WXPYTHON) || ${HAVE_WX:Mpython} != ""
WX_COMPS+=      python
CONFIGURE_ARGS+=--enable-wxpython
.endif
```

6.18.6. Vordefinierte Variablen

Die folgenden Variablen sind in den Ports verfügbar (nachdem sie entsprechend Variablen, um die wxWidgets-Version festzulegen definiert wurden).

Tabelle 35. Vordefinierte Variablen für Ports, die wxWidgets verwenden

Name	Beschreibung
WX_CONFIG	Pfad zum wxWidgets wx-config-Skript (mit unterschiedlichem Namen)
WXRC_CMD	Pfad zum wxWidgets wxrc-Programm (mit unterschiedlichem Namen)
WX_VERSION	Version der wxWidgets, die verwendet werden soll (z.B. 2.6)
WX_UNICODE	Falls Unterstützung für Unicode nicht explizit definiert, jedoch verwendet wird, dann wird die Unterstützung automatisch aktiviert.

6.18.7. Verarbeitung in bsd.port.pre.mk

Falls die Variablen gleich nach dem Importieren von bsd.port.pre.mk benutzt werden sollen, so muss die Variable WX_PREMK definiert werden.



Falls WX_PREMK definiert ist, so werden Version, Abhängigkeiten, Komponenten und vordefinierte Variablen nicht geändert, wenn die Variablen des wxWidgets-Ports *nach* dem Einbinden von bsd.port.pre.mk geändert werden.

Beispiel 19. Verwendung von wxWidgets-Variablen in Kommandos

Der folgende Ausschnitt zeigt die Verwendung von WX_PREMK durch Ausführen des wx-config -Skriptes, um die vollständige Version als Zeichenkette zu erhalten, diese dann einer Variablen zuzuweisen und die Variable anschließend einem Programm zu übergeben.

```
USE_WX=         2.4
WX_PREMK=       yes
```

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

.if exists(${WX_CONFIG})
VER_STR!=       ${WX_CONFIG} --release

PLIST_SUB+=     VERSION="${VER_STR}"
.endif
```



Die wxWidgets-Variablen können problemlos in Kommandos benutzt werden, falls diese in Targets ohne gesetztes `WX_PREMK` verwendet werden.

6.18.8. Weitere `configure`-Argumente

Einige GNU `configure`-Skripte können wxWidgets nicht auffinden, falls nur die Umgebungsvariable `WX_CONFIG` gesetzt ist, sondern benötigen zusätzliche Argumente. Dafür kann die Variable `WX_CONF_ARGS` benutzt werden.

Tabelle 36. Zulässige Werte für `WX_CONF_ARGS`

Möglicher Wert	Resultierendes Argument
`absolute`	`--with-wx-config=${WX_CONFIG}`
`relative`	`--with-wx=${LOCALBASE} --with-wx-config=${WX_CONFIG:T}`

6.19. Verwendung von Lua

Dieser Abschnitt beschreibt den Status der Lua-Bibliotheken in den Ports und deren Einbindung in das Ports System.

6.19.1. Einführung

Es gibt viele Probleme bei der gleichzeitigen Verwendung unterschiedlicher Versionen von Lua-Bibliotheken (Dateien unterschiedlicher Versionen haben denselben Dateinamen). In den Ports wurde das Problem gelöst, indem jede Version unter einem eigenen Namen mit der Versionsnummer als Suffix installiert wird.

Der offensichtliche Nachteil dabei ist, dass jede Anwendung so verändert werden muss, dass sie die erwartete Version vorfindet. Dies kann jedoch durch zusätzliche Flags für Compiler und Linker gelöst werden.

6.19.2. Auswahl der Version

Um festzulegen, welche Version von Lua verwendet werden soll, gibt es zwei Variablen (falls nur eine der beiden definiert ist, so wird die andere auf einen Standardwert gesetzt):

Tabelle 37. Variablen, um die Lua-Version festzulegen

Variable	Beschreibung	Standardwert
USE_LUA	Liste der Versionen, welche der Port verwenden kann	Alle verfügbaren Versionen
USE_LUA_NOT	Liste der Versionen, die der Port nicht verwenden kann	Nichts

Es folgt eine Liste an möglichen Lua-Versionen und deren zugehöriger Port:

Tabelle 38. Verfügbare Lua-Versionen

Version	Port
4.0	lang/lua4
5.0	lang/lua50
5.1	lang/lua

Die Variablen in Variablen, um die Lua-Version festzulegen können auf einen oder mehrere (durch Leerzeichen getrennt) der folgenden Werte gesetzt werden:

Tabelle 39. Spezifikationen der Lua-Versionen

Beschreibung	Beispiel
Spezielle Version	4.0
Aufsteigende Versionen	5.0+
Absteigende Versionen	5.0-
Versionenintervall (muss aufsteigend sein)	5.0-5.1

Desweiteren gibt es Variablen, über die eine bevorzugte Version festgelegt werden kann. Die Versionen können als Liste angegeben werden, wobei die Reihenfolge der Priorisierung entspricht.

Tabelle 40. Variablen zur Festlegung der bevorzugten Lua-Version

Name	Bestimmt für
WANT_LUA_VER	den Port
WITH_LUA_VER	den Benutzer

Beispiel 20. Auswahl der Lua-Version

Der folgende Ausschnitt entspricht einem Port, der Lua in den Versionen 5.0 oder 5.1 verwenden kann und standardmäßig 5.0 verwendet. Diese Einstellung kann durch die benutzerdefinierte Variable WITH_LUA_VER überschrieben werden.

```
USE_LUA=      5.0-5.1
WANT_LUA_VER= 5.0
```

6.19.3. Komponentenauswahl

Desweiteren gibt es Anwendungen, die nicht direkt Lua-Bibliotheken sind, aber trotzdem mit diesen zusammenhängen. Diese Anwendungen können über die Variable `LUA_COMPS` festgelegt werden. Die folgenden Komponenten sind verfügbar:

Tabelle 41. Verfügbare Lua-Komponenten

Name	Beschreibung	Versionseinschränkungen
`lua`	Hauptbibliothek	Keine
`tolua`	Bibliothek für die Unterstützung von C/C++-Code	`4.0-5.0`
`ruby`	Ruby-Bindungen	`4.0-5.0`



Es gibt weitere Komponenten, die jedoch Module für den Interpreter sind und nicht von Anwendungen benutzt werden (nur von anderen Modulen).

Der Typ der Abhängigkeit kann für jede Komponente durch Hinzufügen eines Suffix (durch Strichpunkt getrennt) festgelegt werden. Falls der Typ nicht angegeben wird, wird ein Standardwert verwendet (siehe Standardtypen für Lua-Abhängigkeiten). Die folgenden Typen sind verfügbar:

Tabelle 42. Verfügbare Typen von Lua-Abhängigkeiten

Name	Beschreibung
`build`	Komponente wird zum Bau benötigt - äquivalent zu `BUILD_DEPENDS`
`run`	Komponente wird zum Ausführen benötigt - äquivalent zu `RUN_DEPENDS`
`lib`	Komponente wird zum Bau und zum Ausführen benötigt - äquivalent zu `LIB_DEPENDS`

Die Standardwerte für die einzelnen Komponenten sind in der folgenden Tabelle aufgeführt:

Tabelle 43. Standardtypen für Lua-Abhängigkeiten

Komponente	Typ der Abhängigkeit
`lua`	`lib` für `4.0-5.0` (shared) und `build` für `5.1` (static)
`tolua`	`build` (static)
`ruby`	`lib` (shared)

Beispiel 21. Auswahl von Lua-Komponenten

Der folgende Ausschnitt entspricht einem Port, welcher die Lua-Version `4.0` und die zugehörigen Ruby-Bindungen verwendet.

```
USE_LUA=      4.0
LUA_COMPS=    lua ruby
```

6.19.4. Feststellen der installierten Version

Um eine bereits installierte Version zu finden, muss `WANT_LUA` definiert werden. Falls diese Variable nicht auf eine bestimmte Versionsnummer gesetzt wird, werden die Komponenten einen Suffix mit der Versionsnummer tragen. Die Variable `HAVE_LUA` wird gesetzt, falls eine installierte Version vorgefunden wurde.

Beispiel 22. Installierte Lua-Versionen und- Komponenten feststellen

Der folgende Ausschnitt kann in einem Port verwendet werden, der Lua benutzt, falls es installiert ist oder eine Option dafür ausgewählt wurde.

```
WANT_LUA=       yes

.include <bsd.port.pre.mk>

.if defined(WITH_LUA5) || ${HAVE_LUA:Mlua-5.[01]} != ""
USE_LUA=        5.0-5.1
CONFIGURE_ARGS+=--enable-lua5
.endif
```

Der folgende Ausschnitt kann verwendet werden, um die Unterstützung für tolua zusätzlich zu der von Lua zu aktivieren (beide in Version 4.0), wenn dies installiert ist oder die Option ausgewählt wurde.

```
USE_LUA=        4.0
LUA_COMPS=      lua
WANT_LUA=       4.0

.include <bsd.port.pre.mk>

.if defined(WITH_TOLUA) || ${HAVE_LUA:Mtolua} != ""
LUA_COMPS+=     tolua
CONFIGURE_ARGS+=--enable-tolua
.endif
```

6.19.5. Vordefinierte Variablen

Die folgenden Variablen sind in den Ports verfügbar (nachdem sie entsprechend Variablen, um die Lua-Version festzulegen definiert wurden).

Tabelle 44. Vordefinierte Variablen für Ports, die Lua verwenden

Name	Beschreibung
LUA_VER	Die Lua-Version, die verwendet wird (z.B. 5.1)
LUA_VER_SH	Die Hauptversion für shared-Lua-Bibliotheken (z.B. 1)
LUA_VER_STR	Die Lua-Version ohne die Punkte (z.B. 51)
LUA_PREFIX	Der Präfix, unter dem Lua (und Komponenten) installiert ist
LUA_SUBDIR	Das Verzeichnis unter ${PREFIX}/bin, ${PREFIX}/share und ${PREFIX}/lib, in welchem Lua installiert ist
LUA_INCDIR	Das Verzeichnis, in dem Lua- und tolua-Header-Dateien installiert sind
LUA_LIBDIR	Das Verzeichnis, in dem Lua- und tolua-Bibliotheken installiert sind
LUA_MODLIBDIR	Das Verzeichnis, in dem Lua Modul-Bibliotheken (.so) installiert sind
LUA_MODSHAREDIR	Das Verzeichnis, in dem Lua-Module (.lua) installiert sind
LUA_PKGNAMEPREFIX	Der Paketnamen-Präfix, der von Lua-Modulen verwendet wird
LUA_CMD	Das Verzeichnis, in dem der Lua-Interpreter liegt
LUAC_CMD	Das Verzeichnis, in dem der Lua-Compiler liegt
TOLUA_CMD	Das Verzeichnis, in dem das tolua-Programm liegt

Beispiel 23. Einem Port mitteilen, in welchem Verzeichnis Lua liegt

Der folgende Ausschnitt zeigt, wie einem Port, welcher ein configure-Skript verwendet, mitgeteilt werden kann, wo die Lua-Header-Dateien und Bibliotheken liegen.

```
USE_LUA=        4.0
GNU_CONFIGURE=  yes
CONFIGURE_ENV=  CPPFLAGS="-I${LUA_INCDIR}" LDFLAGS="-L${LUA_LIBDIR}"
```

6.19.6. Verarbeitung in bsd.port.pre.mk

Falls die Variablen gleich nach dem Einbinden von bsd.port.pre.mk benutzt werden sollen, so muss die Variable LUA_PREMK definiert werden.



Falls LUA_PREMK definiert ist, so werden Version, Abhängigkeiten, Komponenten und vordefinierte Variablen nicht geändert, wenn die Variablen des Lua-Ports *nach*

dem Einbinden von bsd.port.pre.mk geändert werden.

Beispiel 24. Verwendung von Lua-Variablen in Kommandos

Der folgende Ausschnitt zeigt die Verwendung von LUA_PREMK durch Ausführen des Lua-Interpreters, um die vollständige Version als Zeichenkette zu erhalten, diese dann einer Variablen zuzuweisen und die Variable schließlich einem Programm zu übergeben.

```
USE_LUA=        5.0
LUA_PREMK=      yes

.include <bsd.port.pre.mk>

.if exists(${LUA_CMD})
VER_STR!=       ${LUA_CMD} -v

CFLAGS+=        -DLUA_VERSION_STRING="${VER_STR}"
.endif
```



Die Lua-Variablen können problemlos in Befehlen benutzt werden, falls diese in Targets ohne gesetztes LUA_PREMK verwendet werden.

6.20. Xfce verwenden

Die USE_XFCE-Variable wird für die automatische Konfiguration der Abhängigkeiten eingesetzt, welche die Xfce-Basisbibliotheken oder Anwendungen wie x11-toolkits/libxfce4gui und x11-wm/xfce4-panel verwenden.

Die folgenden Xfce-Bibliotheken und -Anwendungen werden derzeit unterstützt:

- libexo: x11/libexo
- libgui: x11-toolkits/libxfce4gui
- libutil: x11/libxfce4util
- libmcs: x11/libxfce4mcs
- mcsmanager: sysutils/xfce4-mcs-manager
- panel: x11-wm/xfce4-panel
- thunar: x11-fm/thunar
- wm: x11-wm/xfce4-wm
- xfdev: dev/xfce4-dev-tools

Die folgenden zusätzlichen Parameter werden unterstützt:

- configenv: Benutzen Sie dies, wenn Ihr Port eine speziell angepasste CONFIGURE_ENV-Variable benötigt, um seine erforderlichen Bibliotheken zu finden.

```
-I${LOCALBASE}/include
        -L${LOCALBASE}/lib
```

wird CPPFLAGS hinzugefügt und ergibt `CONFIGURE_ENV`.

Wenn also ein Port von sysutils/xfce4-mcs-manager abhängt und die speziellen CPPFLAGS in seiner configure-Umgebung verlangt, dann würde die Syntax wie folgt aussehen:

```
USE_XFCE=       mcsmanager configenv
```

6.21. Mozilla verwenden

Tabelle 45. Variablen für Ports, die Mozilla verwenden

`USE_GECKO`	Vom Port unterstützte Gecko-Backends. Mögliche Werte sind: `libxul` (libxul.so), `seamonkey` (libgtkembedmoz.so, (veraltet, sollte daher nicht mehr verwendet werden).
`USE_FIREFOX`	Der Port benötigt Firefox, um korrekt zu funktionieren. Mögliche Werte sind: `yes` (verwendet die Standardversion), `40`, `36`, `35`. Die Standardversion ist derzeit `40`.
`USE_FIREFOX_BUILD`	Um den Port zu bauen, muss Firefox installiert sein. Wird diese Variable gesetzt, wird automatisch auch `USE_FIREFOX` gesetzt.
`USE_SEAMONKEY`	Der Port benötigt Seamonkey, um korrekt zu funktionieren. Mögliche Werte sind: `yes` (verwendet die Standardversion), `20`, `11` (veraltet, sollte daher nicht mehr verwendet werden). Die Standardversion ist `20`.
`USE_SEAMONKEY_BUILD`	Um den Port zu bauen, muss Seamonkey installiert sein. Wird diese Variable gesetzt, wird automatisch auch `USE_SEAMONKEY` gesetzt.
`USE_THUNDERBIRD`	Dieser Port benötigt Thunderbird, um korrekt zu funktionieren. Mögliche Werte sind: `yes` (verwendet die Standardversion), `31`, `30` (veraltet, sollte daher nicht mehr verwendet werden). Die Standardversion ist `31`.
`USE_THUNDERBIRD_BUILD`	Um den Port zu bauen, muss Thunderbird installiert sein. Wird diese Variable gesetzt, wird automatisch auch `USE_THUNDERBIRD` gesetzt.

Eine komplette Liste aller verfügbaren Variablen finden Sie in der Datei /usr/ports/Mk/bsd.gecko.mk.

6.22. Benutzung von Datenbanken

Tabelle 46. Variablen für Ports, die Datenbanken benutzen

Variable	Bedeutung
USE_BDB	Falls die Variable auf yes gesetzt ist, füge eine Abhängigkeit von databases/db41 hinzu. Die Variable kann auch folgende Werte annehmen: 40, 41, 42, 43, 44, 46, 47, 48 oder 51. Sie können eine Folge akzeptierter Werte angeben - USE_BDB=42+ stellt die höchste installierte Version fest und greift auf 42 zurück, falls sonst nichts installiert ist.
USE_MYSQL	Falls die Variable auf yes gesetzt ist, füge databases/mysql55-server als Abhängigkeit hinzu. Die damit verknüpfte Variable WANT_MYSQL_VER kann Werte wie 323, 40, 41, 50, 51, 52, 55, oder 60 annehmen.
USE_PGSQL	Falls die Variable auf yes gesetzt ist, füge eine Abhängigkeit von databases/postgresql84 hinzu. Die damit verknüpfte Variable WANT_PGSQL_VER kann Werte wie 73, 74, 80, 81, 82, 83, oder 90 annehmen.

Weitere Informationen zu diesem Thema finden sich in der Datei bsd.database.mk.

6.23. Starten und Anhalten von Diensten (rc Skripten)

rc.d-Skripten werden zum Starten von Diensten während des Systemstarts verwendet und um den Administratoren einen Standardweg zum Anhalten und Starten von Diensten zu bieten. Ports halten sich an dieses systemweite rc.d-Framework. Details zu deren Benutzung können im rc.d Kapitel des Handbuchs nachgelesen werden. Ausführliche Beschreibungen der verfügbaren Befehle stehen in rc(8) und rc.subr(8). Desweiteren gibt es einen Artikel zu praktischen Aspekten bezüglich rc.d-Skripten.

Ein oder mehrere rc.d-Skripten können installiert werden mittels:

```
USE_RC_SUBR=    doormand
```

Skripten müssen im Unterverzeichnis files abgelegt und jeder Skript-Datei muss ein .in-Suffix hinzugefügt werden. Standardmäßige SUB_LIST-Ersetzungen werden für diese Dateien unterstützt. Die Verwendung von %%PREFIX%% und %%LOCALBASE%% wird dringend empfohlen. Näheres zu SUB_LIST kann im zugehörigen Kapitel nachgelesen werden.

Für FreeBSD-Versionen, die älter als 6.1-RELEASE sind, ist die Integration mittels rcorder(8) möglich, indem USE_RCORDER anstatt USE_RC_SUBR verwendet wird. Die Verwendung dieser Methode

ist jedoch nur notwendig, wenn der Port in die Verzeichnisstruktur des Basissystems installiert werden kann oder der Dienst vor den FILESYSTEMS-Skripten in rc.d des Basissystems gestartet sein muss.

Seit FreeBSD 6.1-RELEASE sind lokale rc.d-Skripten (inklusive der durch Ports installierten) im allgemeinen rcorder(8) des Basissystems.

Beispiel eines einfachen rc.d-Skripts:

```
#!/bin/sh

# $FreeBSD$
#
# PROVIDE: doormand
# REQUIRE: LOGIN
# KEYWORD: shutdown
#
# Add the following lines to /etc/rc.conf.local or /etc/rc.conf
# to enable this service:
#
# doormand_enable (bool):    Set to NO by default.
#                Set it to YES to enable doormand.
# doormand_config (path):    Set to %%PREFIX%%/etc/doormand/doormand.cf
#                by default.
#

. /etc/rc.subr

name="doormand"
rcvar=${name}_enable

command=%%PREFIX%%/sbin/${name}
pidfile=/var/run/${name}.pid

load_rc_config $name

: ${doormand_enable="NO"}
: ${doormand_config="%%PREFIX%%/etc/doormand/doormand.cf"}

command_args="-p $pidfile -f $doormand_config"

run_rc_command "$1"
```

Solange kein guter Grund dafür besteht, einen Dienst früher starten zu lassen, sollten alle Ports-Skripten

```
REQUIRE: LOGIN
```

verwenden. Falls der Port von einem bestimmten Benutzer (außer root) ausgeführt wird, ist dies zwingend.

```
KEYWORD: shutdown
```

ist im Skript oben deswegen vorhanden, weil der frei erfundene Beispiel-Port einen Dienst startet und dieser beim Herunterfahren des Systems sauber beendet werden sollte. Startete das Skript keinen persistenten Dienst, wäre dies nicht notwendig.

Für die Wertzuweisung von Variablen sollte "=" anstatt ":=" verwendet werden, da bei Ersterem nur auf einen Standardwert gesetzt wird, wenn die Variable vorher noch nicht gesetzt war, und bei Letzterem dieser gesetzt wird, auch wenn der Wert vorher Null gewesen ist. Ein Benutzer kann durchaus einen Ausdruck wie

```
doormand_flags=""
```

in seiner rc.conf.local-Datei stehen haben, und eine Variablenzuweisung mittels ":=" würde in diesem Fall die Benutzerdefinition überschreiben.



Es sollten keine weiteren Skripten mit der .sh-Endung hinzugefügt werden. Irgendwann wird es ein Massenumbenennen aller Skripten im Repository geben, die immer noch diese Endung haben.

6.23.1. Anhalten und Deinstallieren von Diensten

Es ist möglich, dass ein Dienst während der Deinstallation automatisch angehalten wird. Es wird empfohlen dieses Verhalten nur zu implementieren, wenn es unbedingt erforderlich ist zuerst den Dienst anzuhalten und dann die Dateien zu entfernen. Normalerweise sollte es dem Administrator überlassen werden, ob ein Dienst durch Deinstallieren angehalten werden soll. Dies betrifft auch den Vorgang des Aktualisierens.

Der Datei pkg-plist sollte eine Zeile wie folgt zugefügt werden:

```
@stopdaemon doormand
```

Das Argument muss dabei mit dem Inhalt der USE_RC_SUBR-Variablen übereinstimmen.

6.24. Hinzufügen von Benutzern und Gruppen

Manche Ports setzen voraus, dass ein bestimmter Benutzer auf dem System angelegt ist. Wählen Sie in einem solchen Fall eine freie Kennnummer zwischen 50 und 999 aus und tragen Sie diese in ports/UIDs (für Benutzer) oder ports/GIDs (für Gruppen) ein. Stellen Sie dabei sicher, dass Sie keine Kennnummer auswählen, die bereits vom System oder von anderen Ports verwendet wird.

Erstellen Sie bitte eine entsprechende Patch-Datei für diese beiden Dateien, wenn für Ihren Port ein

neuer Benutzer oder eine neue Gruppe angelegt werden muss.

Sie können dann die Variablen USERS und GROUPS im Makefile benutzen, um bei der Port-Installation das automatische Anlegen des Benutzers zu veranlassen.

```
USERS=  pulse
GROUPS= pulse pulse-access pulse-rt
```

Die Liste mit den momentan belegten UIDs (GIDs) befindet sich in ports/UIDs (ports/GIDs).

6.25. Von Kernelquellen abhängige Ports

Einige Ports (beispielsweise vom Kernel ladbare Module) benötigen die Kernelsourcen, damit sie gebaut werden können. Die folgenden Zeilen beschreiben den korrekten Weg, wie Sie feststellen können, ob der Benutzer die Kernelsourcen installiert hat:

```
.if !exists(${SRC_BASE}/sys/Makefile)

IGNORE=         requires kernel sources to be installed
.endif
```

Kapitel 7. Fortgeschrittene pkg-plist -Methoden

7.1. Änderungen an pkg-plist mit Hilfe von make-Variablen

Einige Ports, insbesondere die `p5-`-Ports, müssen, abhängig von ihren Konfigurationsoptionen (oder im Falle der p5-Ports von der `perl`-Version), die pkg-plist verändern. Um dies zu vereinfachen, werden für jeden Eintrag in pkg-plist die Variablen `%%OSREL%%`, `%%PERL_VER%%` und `%%PERL_VERSION%%` durch die jeweiligen Werte ersetzt. Der Wert von `%%OSREL%%` ist die Revisionsnummer des Betriebssystems (z.B. `4.9`). `%%PERL_VERSION%%` und `%%PERL_VER%%` geben die vollständige Versionsnummer von `perl` (z.B. `5.8.9`) an. Weitere, die Dokumentationsdateien des Ports betreffende `%%VARS%%`, werden im entsprechenden Abschnitt erläutert.

Falls Sie weitere Ersetzungen von Variablen durchführen müssen, können Sie in der Variable `PLIST_SUB` eine Liste von `VAR=VALUE`-Paaren angeben, wobei in der pkg-plist`%%VAR%%` durch *VALUE* ersetzt wird.

Wenn Sie z.B. einen Port haben, der viele Dateien in ein versionsspezifisches Unterverzeichnis installiert, dann können Sie etwas wie

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

in das Makefile schreiben und `%%OCTAVE_VERSION%%` verwenden, unabhängig davon, wo die Variable in pkg-plist verwendet wird. In diesem Fall müssen Sie bei einem Upgrade des Ports nicht dutzende (oder manchmal sogar hunderte) Zeilen in pkg-plist anpassen.

Falls Ihr Port in Abhängigkeit von den ausgewählten Optionen Dateien installiert, ist es üblich, den entsprechenden Zeilen in der pkg-plist eine Zeichenfolge `%%TAG%%` voranzustellen, wobei der Platzhalter `TAG` der Variablen `PLIST_SUB` im Makefile bei gleichzeitiger Zuweisung des speziellen Werts `@comment` hinzugefügt wird, der die Paket-Werkzeuge die Zeile ignorieren lässt:

```
.if defined(WITH_X11)
PLIST_SUB+= X11=""
.else
PLIST_SUB+= X11="@comment "
.endif
```

und in der pkg-plist:

```
%%X11%%bin/foo-gui
```

Diese Ersetzung (ebenso wie das Hinzufügen weiterer Manualpages) wird zwischen den pre-install- und do-install-Targets ausgeführt, indem aus PLIST gelesen und in TMPPLIST geschrieben wird (Standard: WRKDIR/.PLIST.mktmp). Falls Ihr Port also PLIST während dem Erstellen generiert, so sollte dies vor oder in pre-install geschehen. Muss Ihr Port die resultierende Datei verändern, so sollte dies in post-install mit der Ausgabedatei TMPPLIST erfolgen.

Eine weitere Möglichkeit, die Paketliste eines Ports zu verändern, besteht darin die Variablen PLIST_FILES und PLIST_DIRS zu setzen. Der Wert jeder der beiden Variablen stellt eine Liste von Pfadnamen dar, die zusammen mit dem Inhalt von PLIST in TMPPLIST geschrieben wird. Dabei unterliegen die Namen in PLIST_FILES und PLIST_DIRS der weiter oben beschriebenen Substitution von %%VAR%%. Die Namen aus PLIST_FILES werden ansonsten unverändert in die endgültige Paketliste übernommen, während den Namen aus PLIST_DIRS noch der Wert von @dirrm vorangestellt wird. Damit die Verwendung von PLIST_FILES und PLIST_DIRS überhaupt möglich ist, müssen diese gesetzt werden, bevor TMPPLIST geschrieben wird - z.B. in pre-install oder vorher.

7.2. Leere Verzeichnisse

7.2.1. Aufräumen leerer Verzeichnisse

Bitte sorgen Sie dafür, dass ihre Ports bei der Deinstallation leere Verzeichnisse löschen. Dazu wird für jedes Verzeichnis, das der Port erzeugt hat, eine @dirrm-Zeile angegeben. Um ein Verzeichnis zu löschen müssen Sie zuerst dessen Unterverzeichnisse entfernen.

```
 :
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
```

Es kann allerdings auch vorkommen, dass @dirrm Fehler ausgibt, da andere Ports ein Verzeichnis ebenfalls nutzen. Deshalb können Sie @dirrmtry verwenden, um nur Verzeichnisse zu löschen, die wirklich leer sind, und damit Warnhinweise vermeiden.

```
@dirrmtry share/doc/gimp
```

Dadurch wird es weder eine Fehlermeldung geben noch wird pkg_delete(1) abnormal beendet werden - auch dann nicht, wenn ${PREFIX}/shared/doc/gimp nicht leer ist, da andere Ports hier ebenfalls Dateien installiert haben.

7.2.2. Erstellen leerer Verzeichnisse

Um leere Verzeichnisse während der Installation eines Ports zu erstellen, bedarf es etwas Aufmerksamkeit. Diese Verzeichnisse werden nicht erstellt, wenn das Paket installiert wird, da Pakete nur die Dateien speichern und pkg_add(1) nur die Verzeichnisse erstellt, die dafür benötigt

werden. Um sicher zu gehen, dass das leere Verzeichnis erstellt wird, wenn ein Paket installiert wird, muss die folgende Zeile in pkg-plist über der entsprechenden `@dirrm` Zeile eingetragen werden:

```
@exec mkdir -p %D/shared/foo/templates
```

7.3. Konfigurationsdateien

Sollte Ihr Port Konfigurationsdateien in PREFIX/etc benötigen, so sollten Sie diese *nicht* einfach installieren und in pkg-plist auflisten. Dies würde pkg_delete(1) veranlassen, diese Dateien zu löschen, selbst wenn wenn sie vom Benutzer editiert wurden.

Stattdessen sollten Beispieldateien mit einem entsprechenden Suffix (beispielsweise filename.sample) versehen werden. Ist die Konfigurationsdatei nicht vorhanden, so sollte die Beispieldatei an deren Platz kopiert werden. Bei der Deinstallation sollte die Konfigurationsdatei gelöscht werden, aber nur, wenn sie nicht vom Benutzer verändert wurde. Das alles muss sowohl im Makefile des Ports als auch in der pkg-plist (für die Installation aus einem Paket) sichergestellt werden.

Beispiel aus einem Makefile:

```
post-install:
    @if [ ! -f ${PREFIX}/etc/orbit.conf ]; then \
    ${CP} -p ${PREFIX}/etc/orbit.conf.sample ${PREFIX}/etc/orbit.conf ; \
    fi
```

Beispiel aus einer pkg-plist:

```
@unexec if cmp -s %D/etc/orbit.conf.sample %D/etc/orbit.conf; then rm -f
%D/etc/orbit.conf; fi
etc/orbit.conf.sample
@exec if [ ! -f %D/etc/orbit.conf ] ; then cp -p %D/%F %B/orbit.conf; fi
```

Wahlweise können Sie auch eine Nachricht ausgegeben lassen, in der Sie den Nutzer auffordern, die Datei an die richtige Stelle zu kopieren und zu bearbeiten, bevor das Programm ausgeführt werden kann.

7.4. Dynamische oder statische Paketliste

Eine *statische Paketliste* ist eine Paketliste, die in der Ports-Sammlung, entweder in Form der pkg-plist (mit oder ohne der Ersetzung von Variablen) oder durch `PLIST_FILES` und `PLIST_DIRS` im Makefile eingebettet, verfügbar ist. Selbst wenn der Inhalt durch ein Werkzeug oder ein Target im Makefile automatisch erzeugt wird, *bevor* die Datei von einem Committer in die Ports-Sammlung aufgenommen wird, so ist dies immer noch eine statische Liste, da es möglich ist den Dateiinhalt zu betrachten ohne ein Distfile Herunterladen oder Kompilieren zu müssen.

Eine *dynamische Paketliste* ist eine Paketliste, die beim Kompilieren des Ports erstellt wird, abhängig davon, welche Dateien und Verzeichnisse installiert werden. Es ist nicht möglich diese Liste zu betrachten, bevor der Quelltext heruntergeladen und kompiliert oder nachdem ein `make clean` ausgeführt wurde.

Der Einsatz dynamischer Paketlisten ist zwar nicht untersagt, aber Sie sollten, wann immer das möglich ist, statische Paketlisten verwenden, da die Nutzer dann grep(1) auf alle verfügbaren Ports anwenden können, um z.B. herauszufinden, von welchem eine bestimmte Datei installiert wurde. Dynamische Paketlisten sollten für komplexe Ports verwendet werden, bei denen sich die Liste abhängig von den gewählten Funktionen sehr stark ändern kann (wodurch die Pflege von statischen Listen unmöglich wird), oder Ports, welche die Paketliste abhängig von den Versionen verwendeter Abhängigkeiten verändern (z.B. Ports, die Ihre Dokumentation mit Javadoc erzeugen).

Maintainer, die dynamische Paketlisten bevorzugen, werden dazu aufgefordert, neue Targets zu Ihren Ports hinzuzufügen, welche die pkg-plist-Datei erzeugen, sodass Benutzer den Inhalt überprüfen können.

7.5. Automatisiertes Erstellen von Paketlisten

Als Erstes sollten Sie sich vergewissern, dass der Port bis auf pkg-plist vollständig ist.

Als Nächstes erstellen Sie einen temporären Verzeichnisbaum, in welchem Ihr Port installiert werden kann, und installieren Sie alle Abhängigkeiten.

```
# mkdir /var/tmp/`make -V PORTNAME`
# mtree -U -f `make -V MTREE_FILE` -d -e -p /var/tmp/`make -V PORTNAME`
# make depends PREFIX=/var/tmp/`make -V PORTNAME`
```

Speichern Sie die Verzeichnisstruktur in einer neuen Datei.

```
# (cd /var/tmp/`make -V PORTNAME` && find -d * -type d) | sort > OLD-DIRS
```

Erstellen Sie eine leere pkg-plist-Datei:

```
# :>pkg-plist
```

Wenn Ihr Port auf `PREFIX` achtet (was er machen sollte), so kann der Port nun installiert und die Paketliste erstellt werden.

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

Sie müssen auch alle neu erstellten Verzeichnisse in die Paketliste aufnehmen.

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

Zu guter Letzt muss die Paketliste noch manuell aufgeräumt werden - es funktioniert eben nicht *alles* automatisch. Manualpages sollten im Makefile des Ports unter `MANn` aufgeführt sein und nicht in der Paketliste. Konfigurationsdateien des Benutzers sollten entfernt oder als filename.sample installiert werden. Die info/dir-Datei sollte nicht aufgeführt sein und die zugehörigen install-info -Zeilen sollten hinzugefügt werden, wie im info files-Abschnitt beschrieben. Alle Bibliotheken, die der Port installiert, sollten aufgelistet werden, wie es im Shared Libraries-Abschnitt festgelegt ist.

Alternativ dazu können Sie das `plist`-Skript in /usr/ports/Tools/scripts/ verwenden, um die Paketliste automatisch zu erstellen. Das plist-Skript ist ein Ruby-Skript, das die meisten der in den vorangehenden Absätzen kurz dargestellten manuellen Schritte automatisiert.

Der erste Schritt ist derselbe wie oben: Nehmen Sie die ersten drei Zeilen, also `mkdir`, `mtree` und `make depends`. Installieren und bauen Sie dann den Port:

```
# make install PREFIX=/var/tmp/`make -V PORTNAME`
```

Und lassen Sie `plist` die pkg-plist-Datei erstellen:

```
# /usr/ports/Tools/scripts/plist -Md -m `make -V MTREE_FILE` /var/tmp/`make -V
PORTNAME` > pkg-plist
```

Die Paketliste muss immer noch von Hand aufgeräumt werden, wie es oben erklärt wurde.

Ein weiteres Werkzeug zur Erzeugung einer ersten pkg-plist-Datei ist ports-mgmt/genplist. Wie bei jedem automatisierten Hilfswerkzeug, sollte die erzeugte pkg-plist-Datei überprüft und bei Bedarf von Hand nachbearbeitet werden.

Es gibt noch einige Tricks mit pkg-*, die wir noch nicht erwähnt haben, die aber oft sehr praktisch sind.

Kapitel 8. Die pkg-* Dateien

8.1. pkg-message

Wenn Sie dem Anwender bei der Installation weitere Informationen anzeigen wollen, so können Sie diese Nachricht in pkg-message speichern. Diese Vorgehensweise ist oft nützlich, um zusätzliche Schritte anzuzeigen, die nach pkg_add(1) durchgeführt werden müssen. Dadurch können Sie auch Lizenzinformationen darstellen.

Wollen Sie nur ein paar Zeilen über die Einstellungen zum Erstellen des Ports oder Warnungen ausgeben, benutzen Sie ECHO_MSG. pkg-message ist nur für Schritte nach der Installation vorgesehen. Sie sollten den Unterschied zwischen ECHO_MSG und ECHO_CMD beachten: Ersteres wird benutzt, um Informationen auf dem Bildschirm auszugeben, während Letzteres für Kommando-Pipelining bestimmt ist.

Ein gutes Beispiel für die Benutzung der beiden Befehle ist in shells/bash2/Makefile zu finden:

```
update-etc-shells:
    @${ECHO_MSG} "updating /etc/shells"
    @${CP} /etc/shells /etc/shells.bak
    @( ${GREP} -v ${PREFIX}/bin/bash /etc/shells.bak; \
    ${ECHO_CMD} ${PREFIX}/bin/bash) >/etc/shells
    @${RM} /etc/shells.bak
```



Stellen Sie sicher, dass sie auf die korrekten Tools zum Verwalten von Diensten verweisen. * Verwenden Sie `service name start`, um einen Dienst zu starten, anstatt `/usr/local/etc/rc.d/name start` zu verwenden. * Verwenden Sie `sysrc name_enable=YES`, um Optionen in rc.conf zu ändern.



Die pkg-message wird nicht zur pkg-plist hinzugefügt. Sie wird auch nicht automatisch angezeigt, falls ein Anwender den Port installiert. Sie müssen also die Ausgabe selbst im `post-install`-Ziel des Make-Vorgangs veranlassen.

8.2. pkg-install

Sollte es nötig sein, dass Ihr Port bei der Installation des Binärpakets mit pkg_add(1) Befehle ausführt, können Sie das Skript pkg-install benutzen. Dieses Skript wird automatisch dem Paket hinzugefügt und zweimal von pkg_add(1) ausgeführt: Zuerst als `${SH} pkg-install ${PKGNAME} PRE-INSTALL` und beim zweiten Mal als `${SH} pkg-install ${PKGNAME} POST-INSTALL`. `$2` kann also getestet werden, um festzustellen, in welchem Modus das Skript ausgeführt wird. Die Umgebungsvariable `PKG_PREFIX` wird auf das Verzeichnis gesetzt, in welches das Paket installiert wird. Siehe pkg_add(1) für weiterführende Informationen.



Das Skript wird nicht automatisch ausgeführt, wenn Sie den Port mit `make install` installieren. Wenn Sie es ausführen lassen wollen, dann müssen Sie es im Makefile

> aufrufen: `PKG_PREFIX=${PREFIX} ${SH} ${PKGINSTALL} ${PKGNAME} PRE-INSTALL.`

8.3. pkg-deinstall

Dieses Skript wird ausgeführt, wenn ein Paket deinstalliert wird.

Es wird zweimal von pkg_delete(1) aufgerufen. Das erste Mal als `${SH} pkg-deinstall ${PKGNAME} DEINSTALL` und dann als `${SH} pkg-deinstall ${PKGNAME} POST-DEINSTALL.`

8.4. pkg-req

Muss Ihr Port entscheiden, ob er installiert werden soll oder nicht, können Sie ein pkg-req -"Bedingungsskript" verwenden. Dieses wird automatisch bei der Installation/ Deinstallation aufgerufen, um zu entscheiden, ob die Installation/ Deinstallation fortgesetzt werden soll.

Das Skript wird während der Installation von pkg_add(1) als `pkg-req ${PKGNAME} INSTALL` aufgerufen. Bei der Deinstallation wird es von pkg_delete(1) als `pkg-req ${PKGNAME} DEINSTALL` ausgeführt.

8.5. Ändern der Namen der pkg-* Dateien

Alle Namen der pkg- Dateien werden durch Variablen festgelegt. Sie können sie bei Bedarf also im Makefile des Ports ändern. Das ist besonders nützlich, wenn Sie die gleichen pkg- Dateien in mehreren Ports nutzen oder in eine der oben genannten Dateien schreiben wollen. Schreiben Sie niemals außerhalb des Unterverzeichnisses `WRKDIR`pkg-*, eine Erklärung hierzu finden Sie in Schreiben ausserhalb von `WRKDIR`.

Hier ist eine Liste von Variablennamen und ihren Standardwerten (`PKGDIR` ist standardmäßig `${MASTERDIR}`).

Variable	Standardwert
`DESCR`	`${PKGDIR}/pkg-descr`
`PLIST`	`${PKGDIR}/pkg-plist`
`PKGINSTALL`	`${PKGDIR}/pkg-install`
`PKGDEINSTALL`	`${PKGDIR}/pkg-deinstall`
`PKGREQ`	`${PKGDIR}/pkg-req`
`PKGMESSAGE`	`${PKGDIR}/pkg-message`

Bitte benutzen Sie diese Variablen anstatt `PKG_ARGS` zu ändern. Wenn Sie `PKG_ARGS` modifizieren, werden diese Dateien bei der Installation des Ports nicht korrekt in /var/db/pkg installiert.

8.6. Nutzung von `SUB_FILES` und `SUB_LIST`

Die Variablen `SUB_FILES` und `SUB_LIST` sind nützlich, um dynamische Werte in Port-Dateien zu verwenden, wie beispielsweise der Installations-`PREFIX` in pkg-message.

Die Variable SUB_FILES enthält eine Liste von Dateien, die automatisch verändert werden. Jede *Datei* in SUB_FILES muss ein entsprechendes Pendant *datei.in* im Verzeichnis FILESDIR haben. Die modifizierte Version wird in WRKDIR angelegt. Dateien, die als Werte von USE_RC_SUBR (oder veraltet in USE_RCORDER) gespeichert werden, werden automatisch zu SUB_FILES hinzugefügt. Für die Dateien pkg-message, pkg-install, pkg-deinstall und pkg-req werden die jeweiligen Makefile-Variablen selbsttätig auf die geänderte Version der Datei gesetzt.

Die Variable SUB_LIST ist eine Liste von VAR=WERT-Paaren. Jedes Paar %%VAR%% in den Dateien von SUB_FILES wird mit WERT ersetzt. Einige gebräuchliche Paare werden automatisch definiert: PREFIX, LOCALBASE, DATADIR, DOCSDIR, EXAMPLESDIR. Jede Zeile, die mit @comment beginnt, wird nach der Variablen-Ersetzung aus der neu erstellten Datei gelöscht.

Im folgenden Beispiel wird %%ARCH%% mit der Systemarchitektur in pkg-message ersetzt:

```
SUB_FILES=     pkg-message
SUB_LIST=      ARCH=${ARCH}
```

Beachten Sie bitte, dass in diesem Beispiel die Datei pkg-message.in im Verzeichnis FILESDIR vorhanden sein muss.

Hier ein Beispiel für eine gute pkg-message.in:

```
Now it is time to configure this package.
Copy %%PREFIX%%/shared/examples/putsy/%%ARCH%%.conf into your home directory
as .putsy.conf and edit it.
```

Kapitel 9. Ihren Port testen

9.1. `make describe` ausführen

Einige der FreeBSD-Werkzeuge zur Pflege von Ports, wie zum Beispiel portupgrade(1), verwenden eine Datenbank names /usr/ports/INDEX, welche Eigenschaften, wie z.B. Port-Abhängigkeiten, verfolgt. INDEX wird vom Makefile der höchsten Ebene, ports/Makefile, mittels `make index` erstellt, welches in das Unterverzeichnis jedes Ports wechselt und dort `make describe` ausführt. Wenn also `make describe` bei einem Port fehlschlägt, kann INDEX nicht generiert werden und schnell werden viele Leute darüber unzufrieden sein.



Es ist wichtig diese Datei erzeugen zu können, unabhängig davon, welche Optionen in make.conf vorhanden sind. Bitte vermeiden Sie es daher beispielsweise `.error`-Anweisungen zu benutzen, wenn zum Beispiel eine Abhängigkeit nicht erfüllt wird (Lesen Sie dazu bitte Vermeiden Sie den Gebrauch des `.error`-Konstruktes).

Wenn `make describe` eine Zeichenkette anstatt einer Fehlermeldung erzeugt, sind Sie wahrscheinlich auf der sicheren Seite. Vergleichen Sie die erzeugte Zeichenkette mit bsd.port.mk, um mehr über deren Bedeutung zu erfahren.

Beachten Sie bitte außerdem, dass die Benutzung einer aktuellen Version von `portlint` (wie im nächsten Abschnitt beschrieben) automatisch `make describe` startet.

9.2. Portlint

Bitte überprüfen Sie Ihre Arbeit stets mit `portlint`, bevor Sie diese einreichen oder committen. `portlint` warnt Sie bei häufigen Fehlern, sowohl funktionaler als auch stilistischer Natur. Für einen neuen (oder repokopierten) Port ist `portlint -A` die gründlichste Variante; für einen bereits existierenden Port ist `portlint -C` ausreichend.

Da `portlint` heuristische Methoden zur Fehlersuche benutzt, kann es vorkommen, dass Warnungen für Fehler erzeugt werden, die keine sind. Gelegentlich kann etwas, das als Problem angezeigt wird, aufgrund von Einschränkungen im Port-System nicht anders gelöst werden. Wenn es Zweifel gibt, fragen Sie am besten auf FreeBSD ports nach.

9.3. Port Tools

Das Programm ports-mgmt/porttools ist Teil der Ports-Sammlung.

`port` ist das Front-End-Skript, das Ihnen dabei behilflich sein kann Ihre Arbeit als Tester zu vereinfachen. Um einen neuen Port zu testen oder einen bereits bestehenden Port zu aktualisieren, können Sie `port test` verwenden, damit die Tests, inklusive der `portlint`-Überprüfung, durchgeführt werden. Dieser Befehl spürt ausserdem alle nicht in pkg-plist enthaltenen Dateien auf und gibt eine Liste dieser aus. Hier ein Beispiel:

```
# port test /usr/ports/net/csup
```

9.4. PREFIX und DESTDIR

PREFIX bestimmt, an welche Stelle der Port installiert werden soll. In der Regel ist dies /usr/local oder /opt, was jedoch anpassbar ist. Ihr Port muss sich an diese Variable halten.

DESTDIR, wenn es vom Benutzer gesetzt wird, bestimmt die alternative Umgebung (in der Regel eine Jail oder ein installiertes System, welches an anderer Stelle als / eingehängt ist). Ein Port wird unter DESTDIR/PREFIX installiert und registriert sich in der Paket-Datenbank unter DESTDIR/var/db/pkg. Da DESTDIR mittels eines chroot(8)-Aufrufs vom Ports-System automatisch gesetzt wird, brauchen Sie keine Änderungen oder besondere Pflege für DESTDIR-konforme Ports.

Der Wert von PREFIX wird auf LOCALBASE gesetzt (Standard ist /usr/local). Falls USE_LINUX_PREFIX gesetzt ist, wird PREFIX LINUXBASE annehmen (Standard ist /compat/linux).

Die Vermeidung der hart kodierten Angaben von /usr/local oder /usr/X11R6 im Quelltext wird den Port viel flexibler machen und erleichtert es die Anforderungen anderer Einsatzorte zu erfüllen. Für X-Ports, die imake benutzen, geschieht dies automatisch; andernfalls kann dies erreicht werden, indem alle Angaben von /usr/local (oder /usr/X11R6 für X-Ports, die nicht imake benutzen) in den verschiedenen Makefiles im Port ersetzt werden, um ${PREFIX} zu lesen, da diese Variable automatisch an jede Stufe des Build- und Install-Prozesses übergeben wird.

Vergewissern Sie sich bitte, dass Ihre Anwendung nichts unter /usr/local an Stelle von PREFIX installiert. Um dies festzustellen, können Sie folgendes machen:

```
# make clean; make package PREFIX=/var/tmp/`make -V PORTNAME`
```

Wenn etwas außerhalb von PREFIX installiert wird, so gibt der Prozess der Paketerstellung eine Meldung aus, dass es die Dateien nicht finden kann.

Dies prüft nicht das Vorhandensein eines internen Verweises oder die richtige Verwendung von LOCALBASE für Verweise auf Dateien anderer Ports. Das Testen der Installation in /var/tmp/make -V PORTNAME würde dies erledigen.

Die Variable PREFIX kann in Ihrem Makefile oder der Umgebung des Benutzers neu gesetzt werden. Allerdings wird für einzelne Ports dringend davon abgeraten diese Variable in den Makefiles direkt zu setzen.

Verweisen Sie bitte außerdem auf Programme/Dateien von anderen Ports durch die oben erwähnten Variablen und nicht mit den eindeutigen Pfadnamen. Wenn Ihr Port zum Beispiel vom Makro PAGER erwartet, dass es den vollständigen Pfadnamen von less enthält, benutzen Sie folgendes Compiler-Flag:

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

anstatt `-DPAGER=\"/usr/local/bin/less\"`. Somit ist die Wahrscheinlichkeit höher, dass es auch funktioniert, wenn der Administrator den ganzen /usr/local-Baum an eine andere Stelle verschoben hat.

9.5. Die Tinderbox

Wenn Sie ein begeisterter Ports-Entwickler sind möchten Sie vielleicht einen Blick auf die Tinderbox werfen. Es ist ein leistungsstarkes System zur Erstellung und zum Testen von Ports, welches auf Skripten basiert, die auf Pointyhat verwendet werden. Sie können Tinderbox installieren, indem Sie den Port ports-mgmt/tinderbox benutzen. Bitte lesen Sie die mitgelieferte Dokumentation gründlich, da die Konfiguration nicht einfach ist.

Um Näheres darüber zu erfahren, besuchen Sie bitte die Tinderbox Homepage.

Kapitel 10. Einen existierenden Port aktualisieren

Wenn Sie feststellen, dass ein Port verglichen mit der neuesten Version des Originalautors nicht mehr auf dem aktuellen Stand ist, sollten Sie als Erstes sicherstellen, dass Sie die aktuellste Version des Ports haben. Diese finden Sie im Verzeichnis ports/ports-current der FreeBSD FTP-Spiegelseiten. Wenn Sie allerdings mit mehr als ein paar Ports arbeiten, werden Sie es wahrscheinlich einfacher finden CVSup zu benutzen, um Ihre gesamte Ports-Sammlung aktuell zu halten, wie es im Handbuch beschrieben wird. Das hat zusätzlich den Vorteil, dass Sie so auch alle Abhängigkeiten des Ports aktuell halten.

Der nächste Schritt besteht darin festzustellen, ob bereits eine Aktualisierung des Ports darauf wartet committet zu werden. Um das sicherzustellen haben Sie folgende Möglichkeiten. Es gibt eine durchsuchbare Schnittstelle zur FreeBSD Problembericht Datenbank (PR - Problem Report) (auch bekannt als `GNATS`). Wählen Sie dazu `Ports` im Drop-Down-Menü und geben Sie den Namen des Ports ein.

Allerdings wird manchmal vergessen den Namen des Ports eindeutig im Feld für die Zusammenfassung anzugeben. In diesem Fall können Sie das FreeBSD Ports Monitoring System (auch bekannt als `portsmon`) nutzen. Dieses versucht PRs von Ports nach Portname zu sortieren. Um PRs nach einem bestimmten Port zu durchsuchen können Sie die Übersicht eines Ports verwenden.

Wenn es keine wartenden PRs gibt, ist der nächste Schritt eine E-Mail an den Maintainer des Ports zu schicken, wie von `make maintainer` gezeigt wird. Diese Person arbeitet vielleicht schon an einer Aktualisierung, oder hat einen guten Grund den Port im Moment nicht zu aktualisieren (z.B. wegen Stabilitätsproblemen der neuen Version). Sie wollen sicher nicht die Arbeit des Maintainers doppelt machen. Beachten Sie bitte, dass für Ports ohne Maintainer `ports@FreeBSD.org` eingetragen ist. Das ist nur die allgemeine FreeBSD ports-Mailingliste, deshalb wird es in diesem Fall wahrscheinlich nicht helfen eine E-Mail dorthin zu schicken.

Wenn Sie der Maintainer bittet die Aktualisierung zu erledigen, oder falls es keinen Maintainer gibt, haben Sie Gelegenheit, FreeBSD zu helfen, indem Sie die Aktualisierung selbst bereitstellen. Dazu verwenden Sie diff(1), das bereits im Basissystem enthalten ist.

Um einen brauchbaren `diff` für einen einzelne Datei zu erstellen, kopieren Sie die zu patchende Datei nach *dateiname.orig* und speichern Ihre Änderungen in die Datei *dateiname*. Danach erzeugen Sie den Patch:

```
% /usr/bin/diff dateiname.orig dateiname > dateiname.diff
```

Soll mehr als eine Datei gepatcht werden, können Sie entweder `cvs diff` verwenden (siehe dazu Patches mit CVS erstellen) oder Sie kopieren den kompletten Port in ein neues Verzeichnis und speichern die Ausgabe des rekursiven diff(1) auf das neue und alte Portverzeichniss (wenn Ihr verändertes Portverzeichnis z.B. superedit und das Original superedit.bak heißt, dann speichern Sie bitte die Ergebnisse von `diff -ruN superedit.bak superedit`). Sowohl vereinheitlichendes als auch kontextabhängiges diff (Auflistung der Unterschiede zweier Dateien) sind akzeptabel, aber im

Allgemeinen bevorzugen Port-Committer vereinheitlichende `diff`s. Bitte beachten Sie die Verwendung der `-N`-Option. Dies ist der gebräuchliche Weg `diff` dazu zu bewegen korrekt damit umzugehen, neue Dateien anzulegen und alte zu löschen. Bevor Sie das diff einsenden überprüfen Sie bitte die Ausgabe, um sicherzugehen, dass die Änderungen sinnvoll sind. Stellen Sie insbesondere sicher, dass Sie das Arbeitsverzeichnis mit `make clean` aufgerät haben).

Um gängige Operationen mit Korrekturdateien zu vereinfachen, können Sie /usr/ports/Tools/scripts/patchtool.py benutzen. Aber lesen Sie bitte vorher /usr/ports/Tools/scripts/README.patchtool.

Falls der Port keinen Maintainer hat und Sie ihn selbst aktiv benutzen, ziehen Sie bitte in Erwägung sich als Maintainer zu melden. FreeBSD hat mehr als 4000 Ports ohne Maintainer und in diesem Bereich werden immer zusätzliche Freiwillige benötigt (Für eine ausführliche Beschreibung der Verantwortlichkeiten eines Maintainers lesen Sie bitte im Developer's Handbook nach).

Der beste Weg uns das diff zu schicken ist mittels send-pr(1) (Kategorie Ports). Wenn Sie der Maintainer des Ports sind, fügen Sie bitte `[maintainer update]` an den Anfang Ihrer Zusammenfassung und setzen Sie die "Klasse" des PR auf `maintainer-update`. Ansonsten sollte die "Klasse" des PR `change-request` sein. Bitte erwähnen Sie alle hinzugefügten oder gelöschten Dateien in der Nachricht, da diese beim Commit ausdrücklich an cvs(1) übergeben werden müssen. Wenn das diff größer ist als 20 Kilobyte komprimieren und uuencoden Sie es bitte. Ansonsten können Sie es in den PR einfügen wie es ist.

Bevor Sie den PR mit send-pr(1) abschicken, sollten Sie den Abschnitt Den Problembericht schreiben im Artikel über Problemberichte lesen. Dieser enthält sehr viel mehr Informationen darüber, wie man nützliche Problemberichte verfasst.



Wenn Sie Ihre Aktualisierung aufgrund von Sicherheitsbedenken oder eines schwerwiegenden Fehlers bereitstellen wollen, informieren Sie bitte das Ports Management Team <portmgr@FreeBSD.org>, um einen sofortigen Rebuild und eine Neuverteilung des Pakets Ihres Ports durchzuführen. Sonst werden ahnungslose Nutzer von pkg_add(1) über mehrere Wochen die alte Version durch `pkg_add -r` installieren.



Noch einmal: Bitte verwenden Sie diff(1) und nicht shar(1), um Aktualisierungen existierender Ports zu senden. Sie erleichtern es damit den Ports-Committern, Ihre Änderungen nachzuvollziehen.

Nun, da Sie all das geschafft haben, können Sie in Auf dem Laufenden bleiben nachlesen, wie Sie den Port aktuell halten.

10.1. Patches mit CVS erstellen

Wenn möglich, sollten Sie stets eine cvs(1)-Differenz einreichen. Diese sind leichter zu bearbeiten als Differenzen zwischen "neuen und alten" Verzeichnissen. Außerdem könenn Sie so einfacher feststellen, welche Änderungen Sie vorgenommen haben oder Ihren Patch modifizieren, falls dies durch Änderungen in einem anderen Bereich der Ports-Sammlung notwendig wird oder Sie vom Committer um eine Korrektur Ihres Patches gebeten werden.

```
% cd ~/my_wrkdir ①
% cvs -d R_CVSROOT co pdnsd ② ③
% cd ~/my_wrkdir/pdnsd
```

① Das Verzeichnis, in dem Sie den Port bauen wollen. Dieses Arbeitsverzeichnis kann sich auch außerhalb von /usr/ports/ befinden.

② R_CVSROOT steht für einen öffentlichen CVS-Server. Eine Liste aller verfügbaren Server finden Sie im FreeBSD Handbuch.

③ Ersetzen Sie "pdnsd" durch den Modulnamen des Ports. Dieser entspricht in der Regel dem Namen des Ports. Allerdings gibt es einige Ausnahmen von dieser Regel, insbesondere bei sprachspezifischen Ports (beispielsweise lautet der Modulname für den Port german/selfhtml de-selfhtml). Um den Namen des Moduls herauszufinden, können Sie entweder die cvsweb-Schnittstelle verwenden oder den kompletten Pfad des Ports angeben (in unserem Beispiel wäre der komplette Pfad also ports/dns/pdnsd).

Danach modifizieren Sie den Port in gewohnter Weise. Falls Sie Dateien hinzufügen oder entfernen, sollten Sie dies mit `cvs` protokollieren:

```
% cvs add new_file
% cvs remove deleted_file
```

Überprüfen Sie die Funktion Ihres Ports anhand der Checklisten in Den Port testen und Ihren Port mit `portlint` überprüfen.

```
% cvs status
% cvs update ①
```

① Dadurch wird versucht, die Differenz zwischen Ihrer geänderten Version und dem aktuellen Stand im CVS zu kombinieren. Achten Sie dabei unbedingt auf die Ausgabe dieses Befehls. Vor jeder Datei wird ein Buchstabe angezeigt, der Ihnen mitteilt, was mit dieser Datei passiert ist. Eine vollständige Liste dieser Präfixe finden Sie in Von cvs update verwendete Präfixe.

Tabelle 47. Von cvs update verwendete Präfixe

U	Die Datei wurde aktualisiert. Es traten dabei keine Probleme auf.
P	Die Datei wurde ohne Probleme aktualisiert (dieses Präfix wird nur verwendet, wenn Sie mit einem entfernten Repository arbeiten).
M	Die Datei wurde modifiziert. Es traten keine Konflikte auf.
C	Die Datei wurde modifiziert, allerdings kam es dabei zu Konflikten zwischen Ihrer geänderten Version und der aktuellen Version im CVS.

Wird das Präfix `C` nach einem `cvs update` angezeigt, bedeutet dies, dass im CVS etwas geändert wurde und cvs(1) daher nicht in der Lage war, Ihre Änderungen und die Änderungen im CVS zu kombinieren. Es ist immer sinnvoll, sich die Änderungen anzusehen, da `cvs` keine Informationen darüber hat, wie ein Port aufgebaut sein soll. Es kann (und wird wahrscheinlich) daher vorkommen, dass sich manchmal Änderungen ergeben, die keinen Sinn machen.

Im letzten Schritt erzeugen Sie einen "unified diff(1)" gegen die derzeit im CVS vorhandenen Dateien:

```
% cvs diff -uN > ../`basename ${PWD}`.diff
```



Verwenden Sie unbedingt die Option `-N`, um sicherzustellen, dass von hinzugefügte oder gelöschte Dateien im Patch erfasst sind. Der Patch enthät auch von Ihnen gelöschte Dateien (allerdings ohne Inhalt). Dies ist wichtig, da nur so der Committer wissen kann, welche Dateien er entfernen muss.

Zuletzt reichen Sie Ihren Patch ein, indem Sie der Anleitung in Einen existierenden Port aktualisieren folgen.

10.2. Die Dateien UPDATING und MOVED

Wenn die Aktualisierung des Ports spezielle Schritte wie die Anpassung von Konfigurationsdateien oder die Ausführung eines speziellen Programms erfordert, sollten Sie diesen Umstand in der Datei /usr/ports/UPDATING dokumentieren. Einträge in dieser Datei haben das folgende Format:

```
YYYYMMDD:
  AFFECTS: users of portcategory/portname
  AUTHOR: Your name <Your email address>

  Special instructions
```

Wenn Sie exakte Portmaster oder Portupgrade-Meldungen einfügen wollen, stellen Sie bitte sicher, dass alle Sonderzeichen korrekt dargestellt werden.

Wurde der Port gelöscht oder umbenannt, sollten Sie dies in der Datei /usr/ports/MOVED vermerken. Einträge in dieser Datei haben das folgende Format:

```
old name|new name (blank for deleted)|date of move|reason
```

Kapitel 11. Sicherheit der Ports

11.1. Warum Sicherheit so wichtig ist

Es finden sich immer wieder Fehler in Software. Die gefährlichsten davon sind wohl jene, die Sicherheitslücken öffnen. Technisch gesehen müssen diese Lücken geschlossen werden, indem die Fehler, die Sie verursacht haben, beseitigt werden. Aber die Vorgehensweisen, wie mit bloßen Fehlern und Sicherheitslücken umgegangen wird, sind sehr unterschiedlich.

Ein typischer kleiner Fehler betrifft nur Nutzer, die eine bestimmte Kombination von Optionen aktiviert haben, die den Fehler auslöst. Der Entwickler wird letztendlich einen Patch herausgeben, gefolgt von einer neuen Version des Programms, die den Fehler nicht mehr enthält - jedoch wird die Mehrheit der Nutzer nicht sofort aktualisieren, da sie von diesem Fehler nicht betroffen sind. Ein kritischer Fehler, der zu Datenverlust führen kann, stellt ein schwerwiegendes Problem dar. Dennoch sind sich umsichtige Nutzer bewusst, dass Datenverlust verschiedene Ursachen - neben Softwarefehlern - haben kann, und machen deshalb Sicherungskopien wichtiger Daten. Zumal ein kritischer Fehler sehr schnell entdeckt wird.

Bei einer Sicherheitslücke ist dies ganz anders. Erstens wird sie vielleicht jahrelang nicht entdeckt, da dies oftmals keine Fehlfunktion im Programm verursacht. Zweitens kann eine böswillige Person unerlaubten Zugriff auf ein unsicheres System erlangen, um empfindliche Daten zu verändern oder zu zerstören; im schlimmsten Fall findet der Nutzer nicht einmal die Ursache des Schadens. Drittens hilft der Zugriff auf ein unsicheres System dem Angreifer oft in ein anderes System einzudringen, welches ansonsten nicht gefährdet wäre. Deshalb reicht es nicht aus eine Sicherheitslücke nur zu schließen: Die Zielgruppe sollte möglichst genau und umfassend darüber informiert werden, damit sie die Gefahr einschätzen und passende Maßnahmen ergreifen können.

11.2. Sicherheitslücken schliessen

Bei Ports und Paketen kann eine Sicherheitslücke im ursprünglichen Programm oder in den Port-Dateien verursacht werden. Im ersten Fall wird der ursprüngliche Entwickler den Fehler wahrscheinlich umgehend korrigieren oder eine neue Version herausgeben und Sie müssen den Port nur aktualisieren und die Korrekturen des Autors beachten. Falls sich die Korrektur aus irgendeinem Grund verzögert, sollten Sie den Port als `FORBIDDEN` markieren oder selbst den Fehler für den Port korrigieren. Falls die Sicherheitslücke im Port verursacht wird, sollten Sie ihn sobald wie möglich berichtigen. In jedem Fall sollte die Standardvorgehensweise zum Einreichen von Änderungen beachtet werden - es sei denn, Sie haben das Recht diese direkt in den Ports-Baum zu committen.



Ports-Committer zu sein ist nicht genug, um Änderungen an einem beliebigen Port zu committen. Bitte denken Sie daran, dass Ports üblicherweise Maintainer haben, die Sie respektieren sollten.

Bitte stellen Sie sicher, dass die Revision des Ports erhöht wird, sobald die Sicherheitslücke geschlossen wurde. Dadurch sehen die Nutzer, die installierte Pakete regelmäßig aktualisieren, dass es an der Zeit ist eine Aktualisierung durchzuführen. Außerdem wird ein neues Paket gebaut, über FTP- und WWW-Spiegel verteilt und die unsichere Version damit verdrängt. `PORTREVISION`

sollte erhöht werden - es sei denn, PORTREVISION hat sich im Laufe der Korrektur des Fehlers geändert. Das heißt, Sie sollten PORTREVISION erhöhen, wenn Sie eine Korrektur hinzugefügt haben. Sie sollten diese aber nicht erhöhen, wenn Sie den Port auf die neueste Version des Programms gebracht haben und PORTREVISION somit schon verändert wurde. Bitte beachten Sie den betreffenden Abschnitt für weitere Informationen.

11.3. Die Community informiert halten

11.3.1. Die VuXML-Datenbank

Ein sehr wichtiger und dringender Schritt, den man unternehmen muss, sobald eine Sicherheitslücke entdeckt wurde, ist die Gemeinschaft der Anwender des Ports über die Gefahr zu informieren. Diese Benachrichtigung hat zwei Gründe. Erstens wird es sinnvoll sein, wenn die Gefahr wirklich so groß ist, sofort Abhilfe zu schaffen, indem man z.B. den betreffenden Netzwerkdienst beendet oder den Port komplett deinstalliert, bis die Lücke geschlossen wurde. Und Zweitens pflegen viele Nutzer installierte Pakete nur gelegentlich zu aktualisieren. Sie werden aus der Mitteilung erfahren, dass Sie das Paket, sobald eine Korrektur verfügbar ist, sofort aktualisieren *müssen*.

Angesichts der riesigen Zahl an Ports kann nicht für jeden Vorfall ein Sicherheitshinweis erstellt werden, ohne durch die Flut an Nachrichten die Aufmerksamkeit der Empfänger zu verlieren, im Laufe der Zeit kommt es so zu ernsten Problemen. Deshalb werden Sicherheitslücken von Ports in der FreeBSD VuXML-Datenbank aufgezeichnet. Das Team der Sicherheitsverantwortlichen beobachtet diese wegen Angelegenheiten, die Ihr Eingreifen erfordern.

Wenn Sie Committerrechte haben, können Sie die VuXML-Datenbank selbst aktualisieren. Auf diese Weise helfen Sie den Sicherheitsverantwortlichen und liefern die kritischen Informationen frühzeitig an die Community. Aber auch wenn Sie kein Committer sind und glauben, Sie haben eine außergewöhnlich schwerwiegende Lücke gefunden - egal welche - zögern Sie bitte nicht die Sicherheitsverantwortlichen zu kontaktieren, wie es in den FreeBSD Sicherheitsinformationen beschrieben wird.

Wie vielleicht aus dem Titel hervorgeht, handelt es sich bei der VuXMl-Datenbank um ein XML-Dokument. Die Quelldatei vuln.xml können Sie im Port security/vuxml finden. Deshalb wird der komplette Pfadname PORTSDIR/security/vuxml/vuln.xml lauten. Jedes Mal, wenn Sie eine Sicherheitslücke in einem Port entdecken, fügen Sie bitte einen Eintrag dafür in diese Datei ein. Solange Sie nicht mit VuXML vertraut sind, ist es das Beste, was Sie machen können, einen vorhandenen Eintrag, der zu Ihrem Fall passt, zu kopieren und als Vorlage zu verwenden.

11.3.2. Eine kurze Einführung in VuXML

Das komplette XML ist komplex und würde den Rahmen dieses Buches sprengen. Allerdings benötigen Sie für einen grundlegenden Einblick in die Struktur eines VuXML-Eintrags nur eine Vorstellung der Tags. XML-Tags bestehen aus Namen, die in spitzen Klammern eingeschlossen sind. Zu jedem öffnenden <Tag> muss ein passendes </Tag> existieren. Tags können geschachtelt werden. Wenn sie geschachtelt werden müssen die inneren Tags vor den Äußeren geschlossen werden. Es gibt eine Hierarchie von Tags - das heißt komplexere Regeln zur Schachtelung. Klingt so ähnlich wie HTML, oder? Der größte Unterschied ist: XML ist erweiterbar (eXtensible) - das heißt es basiert

darauf maßgeschneiderte Tags zu definieren. Aufgrund seiner wesentlichen Struktur bringt XML ansonsten formlose Daten in eine bestimmte Form. VuXML ist speziell darauf zugeschnitten Beschreibungen von Sicherheitslücken zu verwalten.

Lassen Sie uns nun einen realistischen VuXML-Eintrag betrachten:

```
<vuln vid="f4bc80f4-da62-11d8-90ea-0004ac98a7b9"> ①
  <topic>Several vulnerabilities found in Foo</topic> ②
  <affects>
    <package>
      <name>foo</name> ③
      <name>foo-devel</name>
      <name>ja-foo</name>
      <range><ge>1.6</ge><lt>1.9</lt></range> ④
      <range><ge>2.*</ge><lt>2.4_1</lt></range>
      <range><eq>3.0b1</eq></range>
    </package>
    <package>
      <name>openfoo</name> ⑤
      <range><lt>1.10_7</lt></range> ⑥
      <range><ge>1.2,1</ge><lt>1.3_1,1</lt></range>
    </package>
  </affects>
  <description>
    <body xmlns="http://www.w3.org/1999/xhtml">
      <p>J. Random Hacker reports:</p> ⑦
      <blockquote
        cite="http://j.r.hacker.com/advisories/1">
        <p>Several issues in the Foo software may be exploited
          via carefully crafted QUUX requests.  These requests will
          permit the injection of Bar code, mumble theft, and the
          readability of the Foo administrator account.</p>
      </blockquote>
    </body>
  </description>
  <references> ⑧
    <freebsdsa>SA-10:75.foo</freebsdsa> ⑨
    <freebsdpr>ports/987654</freebsdpr> ⑩
    <cvename>CAN-2010-0201</cvename> ⑪
    <cvename>CAN-2010-0466</cvename>
    <bid>96298</bid> ⑫
    <certsa>CA-2010-99</certsa> ⑬
    <certvu>740169</certvu> ⑭
    <uscertsa>SA10-99A</uscertsa> ⑮
    <uscertta>SA10-99A</uscertta> ⑯
    <mlist
msgid="201075606@hacker.com">http://marc.theaimsgroup.com/?l=bugtraq&amp;m=20388660782
5605</mlist> ⑰
    <url>http://j.r.hacker.com/advisories/1</url> ⑱
  </references>
```

```
  <dates>
    <discovery>2010-05-25</discovery> ⑲
    <entry>2010-07-13</entry> ⑳
    <modified>2010-09-17</modified>
  </dates>
</vuln>
```

Die Namen der Tags sollten selbsterklärend sein - also werfen wir einen genaueren Blick auf die Felder, die Sie selbst ausfüllen müssen:

① Dies ist die höchste Tag-Ebene eines VuXML-Eintrags. Es ist ein vorgeschriebenes Attribut `vid`, welches eine allgemein einzigartige Kennung (universally unique identifier, UUID) in Anführungszeichen für diesen Eintrag festlegt. Sie sollten eine UUID für jeden neuen VuXML-Eintrag erzeugen (und vergessen Sie nicht die UUID der Vorlage zu ersetzen, es sei denn, Sie schreiben den Eintrag von Grund auf selbst). Sie können uuidgen(1) verwenden, um eine VuXML UUID zu erzeugen.

② Dies ist eine einzeilige Beschreibung des gefundenen Fehlers.

③ Hier werden die Namen betroffener Pakete aufgeführt. Es können mehrere Namen angegeben werden, da mehrere Pakete von einem einzigen Master-Port oder Software-Produkt abhängen können. Das schließt Stable- und Developement-Zweige, lokalisierte Versionen und Slave-Ports ein, die verschiedene Auswahlmöglichkeiten wichtiger Kompilierungszeit-Optionen bieten.

④ Betroffene Versionen der Pakete werden hier als ein Bereich oder mehrere durch eine Kombination aus `<lt>`, `<le>`, `<eq>`, `<ge>`, und `<gt>`-Elementen ausgegeben. Die angegebenen Bereiche sollten sich nicht überschneiden.In einer Bereichsangabe steht `*` (Asterisk) für die kleinste Versionsnummer. Insbesondere ist `2.*` kleiner als `2.a`. Deshalb kann ein Stern benutzt werden, um auf alle möglichen `Alpha`-, `Beta`- und `RC`-Versionen zuzutreffen. Zum Beispiel passt `<ge>2.</ge><lt>3. </lt>` auf alle Versionen der Form `2.x`, während `<ge> 2.0</ge><lt>3.0</lt>` das nicht erfüllt, da es nicht auf `2.r3` passt, auf `3.b` aber schon.Das obige Beispiel legt fest, dass Versionen von `1.6` bis `1.9` betroffen sind - außerdem Versionen `2.x` vor `2.4_1` und Version `3.0b1`.

⑤ Mehrere zusammenhängende Gruppen von Paketen (im wesentlichen Ports) können im Abschnitt `<affected>` aufgeführt werden. Das kann man benutzen, wenn sich Programme (sagen wir FooBar, FreeBar und OpenBar) denselben Quelltext als Grundlage haben und sich noch dessen Fehler und Sicherheitslücken teilen. Beachten Sie den Unterschied zum Anführen mehrerer Namen innerhalb eines <package> Abschnittes.

⑥ Die Versionsbereiche sollten, wenn möglich, sowohl `PORTEPOCH` als auch `PORTREVISION` erlauben. Bitte denken Sie daran, dass gemäß der Vergleichsregeln eine Version mit einer `PORTEPOCH`, die nicht Null ist, größer ist als jede Version ohne `PORTEPOCH`. Das heißt, `3.0,1` ist größer als `3.1` oder sogar `8.9`.

⑦ Das ist die Zusammenfassung des Problems. In diesem Feld wird XHTML verwendet. Zumindest umschließende `<p>` und `</p>` sollten auftauchen. Komplexere Tags sind zwar möglich, aber sollten nur um der Genauigkeit und Klarheit willen verwendet werden: Bitte verwenden Sie hier kein Eye-Candy.

⑧ Dieser Abschnitt enthält Verweise auf relevante Dokumente. Es wird empfohlen so viele Referenzen wie nötig aufzuführen.

⑨ Das ist ein FreeBSD Sicherheitshinweis.

⑩ Das ist ein FreeBSD Problembericht.

⑪ Das ist eine Mitre CVE Kennung.

⑫ Das ist eine SecurityFocus Fehler-Kennung.

⑬ Das ist ein Sicherheitshinweis von US-CERT.

⑭ Das ist eine Mitteilung über eine Schwachstelle von US-CERT.

⑮ Das ist ein Cyber-Sicherheitsalarm von US-CERT.

⑯ Das ist ein technischer Cyber-Sicherheitsalarm von US-CERT.

⑰ Das ist eine URL zu einem archivierten Posting auf einer Mailingliste. Das Attribut `msgid` ist optional und gibt die Nachrichtenkennung des Postings an.

⑱ Das ist eine gewöhnliche URL. Sie sollte nur verwendet werden, wenn keine der anderen Referenzkategorien verfügbar ist.

⑲ Das ist das Datum, an dem die Sicherheitslücke bekannt wurde (*JJJJ-MM-TT*).

⑳ Das ist das Datum, an dem der Eintrag hinzugefügt wurde (*JJJJ-MM-TT*).

Das ist das Datum, an dem zuletzt irgendeine Information des Eintrags verändert wurde (*JJJJ-MM-TT*). Neue Einträge dürfen dieses Feld nicht enthalten. Es sollte beim Editieren eines existierenden Eintrags eingefügt werden.

11.3.3. Ihre Änderungen an der VuXML-Datenbank testen

Nehmen wir an, Sie haben gerade einen Eintrag für eine Sicherheitslücke in dem Paket `clamav` geschrieben oder ausgefüllt, die in der Version `0.65_7` korrigiert wurde.

Als Voraussetzung müssen Sie die aktuellen Versionen der Ports ports-mgmt/portaudit, ports-mgmt/portaudit-db sowie security/vuxml*installieren*.



Um `packaudit` auszuführen, müssen Sie die Berechtigung haben DATABASEDIR zu schreiben - üblicherweise ist das /var/db/portaudit.

Durch Setzen der Umgebungsvariable DATABASEDIR können Sie hier auch ein anderes Verzeichnis angeben.

Arbeiten Sie nicht aus dem Verzeichnis ${PORTSDIR}/security/vuxml heraus, müssen Sie zusätzlich die Umgebungsvariable VUXMLDIR setzen, um anzugeben, in welchem Verzeichnis sich die Datei vuln.xml befindet.

Zuerst überprüfen Sie bitte, ob bereits ein Eintrag für diese Schwachstelle existiert. Wenn es einen solchen Eintrag gibt, sollte er auf die vorige Version `0.65_6` zutreffen:

```
% packaudit
% portaudit clamav-0.65_6
```

Wenn keine vorhandenen Einträge gefunden werden haben Sie grünes Licht, einen neuen Eintrag

für diese Sicherheitslücke anzulegen. Sie können nun eine neue UUID erzeugen (wir nehmen an, diese lautet `74a9541d-5d6c-11d8-80e3-0020ed76ef5a`) und einen neuen Eintrag in der VuXML-Datenbank anlegen. Bitte überprüfen Sie danach die Syntax mit folgendem Befehl:

```
% cd ${PORTSDIR}/security/vuxml && make validate
```



Sie werden zumindest eines der folgenden Pakete benötigen: textproc/libxml2, textproc/jade.

Jetzt bauen Sie bitte die `portaudit`-Datenbank aus der VuXML-Datei neu:

```
% packaudit
```

Um sicherzustellen, dass der Abschnitt `<affected>` Ihres Eintrags die richtigen Pakete betrifft, verwenden Sie bitte den folgenden Befehl:

```
% portaudit -f /usr/ports/INDEX -r 74a9541d-5d6c-11d8-80e3-0020ed76ef5a
```



Bitte lesen Sie in portaudit(1) nach, um ein besseres Verständnis der Befehlssyntax zu entwickeln.

Bitte stellen Sie sicher, dass Ihr Eintrag keine falschen Treffer in der Ausgabe erzeugt.

Jetzt überprüfen Sie bitte, dass Ihr Eintrag die richtigen Versionen des Pakets angibt:

```
% portaudit clamav-0.65_6 clamav-0.65_7
Affected package: clamav-0.65_6 (matched by clamav<0.65_7)
Type of problem: clamav remote denial-of-service.
Reference: <http://www.freebsd.org/ports/portaudit/74a9541d-5d6c-11d8-80e3-
0020ed76ef5a.html>

1 problem(s) found.
```

Offensichtlich sollte die erste Version ausgegeben werden - die zweite jedoch nicht.

Abschließend überprüfen Sie bitte, ob die Webseite, die aus der VuXML-Datenbank erzeugt wird, wie erwartet aussieht:

```
% mkdir -p ~/public_html/portaudit
% packaudit
% lynx ~/public_html/portaudit/74a9541d-5d6c-11d8-80e3-0020ed76ef5a.html
```

Kapitel 12. Was man machen respektive vermeiden sollte

12.1. Einführung

Hier ist eine Liste von gebräuchlichen Dos and Don'ts (Dinge, die man machen oder vermeiden sollte), welchen Sie während des Portierungsprozesses begegnen werden. Sie sollten Ihren Port anhand dieser Liste überprüfen. Sie können auch Ports in der PR Datenbank, welche andere Menschen eingereicht haben, kontrollieren. Senden Sie bitte Kommentare zu Ports, die Sie verifizieren wie unter Bug Reports and General Commentary beschrieben. Der Abgleich von Ports aus der PR-Datenbank hilft uns diese schneller zu committen, und zeigt auch, dass Sie wissen, worum es geht.

12.2. WRKDIR

Schreiben Sie in keine Dateien außerhalb von WRKDIR. WRKDIR ist der einzige Ort, welcher während des Erstellen des Ports garantiert beschreibbar ist (siehe Ports Installieren von CDROM für ein Beispiel, um Ports in einem schreibgeschützen Zweig zu erstellen). Wenn Sie eine der pkg-* Dateien modifizieren müssen, sollten Sie eine Variable erneut definieren, anstatt die Datei zu überschreiben.

12.3. WRKDIRPREFIX

Vergewissern Sie sich, dass Ihr Port WRKDIRPREFIX beachtet. Die meisten Ports sollten sich darüber keine Sorgen machen. Beachten Sie bitte, falls auf WRKDIR eines anderen Ports verwiesen wird, dass die korrekte Position WRKDIRPREFIXPORTSDIR/subdir/name/work, und nicht etwa PORTSDIR/subdir/name/work, .CURDIR/../../subdir/name/work oder ähnliches ist.

Falls Sie WRKDIR selbst definieren, sollten Sie sicherstellen, dass Sie ${WRKDIRPREFIX}${.CURDIR} am Anfang anfügen.

12.4. Unterschiedliche Betriebssysteme und Betriebssystemversionen

Sie können auf Quelltext treffen, welcher Modifizierungen oder bedingtes Kompilieren, abhängig davon, unter welcher Unix-Version er läuft, benötigt. Falls Sie Änderungen an solch einem Quelltext vornehmen müssen, stellen Sie bitte sicher, dass Sie Ihre Änderungen so allgemein wie möglich halten, damit wir den Quelltext auf ältere FreeBSD-Systeme portieren und zur Quer-Portierung auf andere BSD-Systeme, wie etwa 4.4BSD von CSRG, BSD/386, 386BSD, NetBSD und OpenBSD verwenden können.

Der bevorzugte Weg, um 4.3BSD/Reno (1990) und neuere Versionen des BSD-Quelltextes zu unterscheiden, ist das BSD-Makro zu nutzen, welches in sys/param.h definiert ist. Hoffentlich ist diese Datei schon enthalten - falls nicht, so fügen Sie folgenden Quelltext:

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

an der richtigen Stelle in der .c Datei hinzu. Wir glauben, dass jedes System, welches diese beiden Symbole definiert, die Datei sys/param.h besitzt. Wenn Sie auf Systeme stoßen, wo dies nicht so ist, würden wir gerne davon erfahren. Bitte senden Sie eine E-Mail an FreeBSD ports.

Eine andere Möglichkeit zur Unterscheidung ist der GNU Autoconf-Stil:

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

Vergessen Sie nicht `-DHAVE_SYS_PARAM_H` zu den `CFLAGS` im Makefile hinzuzufügen, falls Sie diese Methode benutzen sollten.

Sobald Sie sys/param.h hinzugefügt haben, können Sie mit Hilfe von

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

unterscheiden, ob der Quelltext auf einer 4.3 Net2 Code-Basis oder neuer (z.B. FreeBSD 1.x, 4.3/Reno, NetBSD 0.9, 386BSD, BSD/386 1.1 und niedriger) kompiliert werden wird.

Benutzen Sie:

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

um zu differenzieren, ob der Quelltext auf der Basis von 4.4 Code oder neuer (z.B. FreeBSD 2.x, 4.4, NetBSD 1.0, BSD/386 2.0 oder höher) kompiliert werden wird.

Der Wert des `BSD`-Makros ist `199506` für die 4.4BSD-Lite2 Codebasis. Beachten Sie bitte, dass dies hier nur der Information wegen angegeben ist. Das Makro sollte nicht dazu benutzt werden, um zwischen Versionen von FreeBSD, welche auf 4.4-Lite basieren, und Versionen, welche Änderungen von 4.4-Lite2 übernommen haben, zu unterscheiden. Das {freebsd} Makro sollte stattdessen verwandt werden.

Sparsam sollte eingesetzt werden:

- {freebsd} ist in allen Versionen von FreeBSD definiert. Benutzen Sie dieses Makro, falls die Änderung(en), die Sie machen, *nur* FreeBSD betrifft. Portierungsfallen, wie der Gebrauch von `sys_errlist[]` gegenüber `strerror()` sind Berkeley-Eigenheiten, keine FreeBSD Änderungen.
- In FreeBSD 2.x, ist {freebsd} auf `2` definiert. In älteren Versionen, ist es `1`. Alle späteren Versionen erhöhen es, damit es mit der Haupt-Versionsnummer übereinstimmt.
- Falls Sie zwischen einem FreeBSD 1.x und einem FreeBSD 2.x (oder höher) System

unterscheiden müssen, ist es normalerweise richtig, die BSD-Makros (wie oben beschrieben) zu benutzen. Gibt es tatsächlich eine FreeBSD-spezifische Änderung (wie z.B. spezielle Optionen von Shared-Libraries für ld), ist es nicht zu beanstanden {freebsd} und #if {freebsd} > 1 zu nutzen, um FreeBSD 2.x und spätere Systeme zu erkennen. Falls Sie eine höhere Genauigkeit benötigen, um FreeBSD Systeme seit 2.0-RELEASE zu erkennen, können Sie folgendes nutzen:

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

In den Tausenden von Ports, die bis jetzt erstellt wurden, gab es nur ein oder zwei Fälle, in denen {freebsd} hätte benutzt werden sollen. Nur weil ein früherer Port es an der falschen Stelle benutzt hatte, bedeutet das nicht, dass Sie dies auch machen sollten.

12.5. __FreeBSD_version Werte

Hier ist eine praktische Liste von __FreeBSD_version-Werten wie in sys/param.h definiert:

Tabelle 48. __FreeBSD_version-Werte

Wert	Datum	Release
119411		2.0-RELEASE
199501, 199503	19. März 1995	2.1-CURRENT
199504	9. April 1995	2.0.5-RELEASE
199508	26. August 1995	2.2-CURRENT vor 2.1
199511	10. November 1995	2.1.0-RELEASE
199512	10. November 1995	2.2-CURRENT vor 2.1.5
199607	10. Juli 1996	2.1.5-RELEASE
199608	12. Juli 1996	2.2-CURRENT vor 2.1.6
199612	15. November 1996	2.1.6-RELEASE
199612		2.1.7-RELEASE
220000	19. Februar 1997	2.2-RELEASE
(nicht geändert)		2.2.1-RELEASE
(nicht geändert)		2.2-STABLE nach 2.2.1-RELEASE
221001	15. April 1997	2.2-STABLE nach texinfo-3.9
221002	30. April 1997	2.2-STABLE nach top
222000	16. Mai 1997	2.2.2-RELEASE
222001	19. Mai 1997	2.2-STABLE nach 2.2.2-RELEASE

Wert	Datum	Release
225000	2. Oktober 1997	2.2.5-RELEASE
225001	20. November 1997	2.2-STABLE nach 2.2.5-RELEASE
225002	27. Dezember 1997	2.2-STABLE nach der Aufnahme von ldconfig -R
226000	24. März 1998	2.2.6-RELEASE
227000	21. Juli 1998	2.2.7-RELEASE
227001	21. Juli 1998	2.2-STABLE nach 2.2.7-RELEASE
227002	19. September 1998	2.2-STABLE nach semctl(2) Änderung
228000	29. November 1998	2.2.8-RELEASE
228001	29. November 1998	2.2-STABLE nach 2.2.8-RELEASE
300000	19. Februar 1996	3.0-CURRENT vor mount(2) Änderung
300001	24. September 1997	3.0-CURRENT nach mount(2) Änderung
300002	2. Juni 1998	3.0-CURRENT nach semctl(2) Änderung
300003	7. Juni 1998	3.0-CURRENT nach ioctl arg Änderungen
300004	3. September 1998	3.0-CURRENT nach ELF-Konvertierung
300005	16. Oktober 1998	3.0-RELEASE
300006	16. Oktober 1998	3.0-CURRENT nach 3.0-RELEASE
300007	22. Januar 1999	3.0-STABLE nach 3/4 Zweig
310000	9. Februar 1999	3.1-RELEASE
310001	27. März 1999	3.1-STABLE nach 3.1-RELEASE
310002	14. April 1999	3.1-STABLE nach Änderung der C++ Konstruktor/Destruktor-Reihenfolge
320000		3.2-RELEASE
320001	8. Mai 1999	3.2-STABLE
320002	29. August 1999	3.2-STABLE nach binär-inkompatibler IPFW und Socket-Änderungen
330000	2. September 1999	3.3-RELEASE
330001	16. September 1999	3.3-STABLE

Wert	Datum	Release
330002	24. November 1999	3.3-STABLE nach Hinzufügen von mkstemp(3) zur libc
340000	5. Dezember 1999	3.4-RELEASE
340001	17. Dezember 1999	3.4-STABLE
350000	20. Juni 2000	3.5-RELEASE
350001	12. Juli 2000	3.5-STABLE
400000	22. Januar 1999	4.0-CURRENT nach 3.4 Zweig
400001	20. Februar 1999	4.0-CURRENT nach der Änderung im Verhalten des dynamischen Linkers.
400002	13. März 1999	4.0-CURRENT nach Änderung der C++ Konstruktor/Destruktor Reihenfolge.
400003	27. März 1999	4.0-CURRENT nach funktionierendem dladdr(3).
400004	5. April 1999	4.0-CURRENT nach der __deregister_frame_info Fehlerbehebung für den dynamischen Linker (auch 4.0-CURRENT nach EGCS 1.1.2 Integration).
400005	27. April 1999	4.0-CURRENT nach suser(9) API Änderung (auch 4.0-CURRENT nach newbus).
400006	31. Mai 1999	4.0-CURRENT nach Änderung der cdevsw-Registrierung.
400007	17. Juni 1999	4.0-CURRENT nach Hinzufügen von so_cred für Zugangsberechtigungen auf Socket-Ebene.
400008	20. Juni 1999	4.0-CURRENT nach Hinzufügen eines poll Syscall-Wrappers zur libc_r.
400009	20. Juli 1999	4.0-CURRENT nach der Änderung des Kernel `dev_t` -Typs zum `struct specinfo` -Zeiger.
400010	25. September 1999	4.0-CURRENT nach dem Beseitigen eines Fehlers in jail(2).

Wert	Datum	Release
400011	29. September 1999	4.0-CURRENT nach der `sigset_t` Datentyp Änderung.
400012	15. November 1999	4.0-CURRENT nach dem Wechsel zum GCC 2.95.2-Compiler.
400013	4. Dezember 1999	4.0-CURRENT nach Hinzufügen der erweiterbaren Linux Mode ioctl-Routinen.
400014	18. Januar 2000	4.0-CURRENT nach dem OpenSSL-Import.
400015	27. Januar 2000	4.0-CURRENT nach der C++ ABI Änderung in GCC 2.95.2 von -fvtable-thunks zu -fno-vtable -thunks als Standard.
400016	27. Februar 2000	4.0-CURRENT nach OpenSSH-Import.
400017	13. März 2000	4.0-RELEASE
400018	17. März 2000	4.0-STABLE nach 4.0-RELEASE
400019	5. Mai 2000	4.0-STABLE nach der Einführung von verzögerten Prüfsummen.
400020	4. Juni 2000	4.0-STABLE nach dem Einpflegen des libxpg4-Quelltextes in die libc.
400021	8. Juli 2000	4.0-STABLE nach der Aktualisierung von Binutils auf 2.10.0, Änderungen der binären ELF-Markierungen, Aufnahme von tcsh ins Basissystem.
410000	14. Juli 2000	4.1-RELEASE
410001	29. Juli 2000	4.1-STABLE nach 4.1-RELEASE
410002	16. September 2000	4.1-STABLE nachdem setproctitle(3) von der libutil in die libc verschoben wurde.
411000	25. September 2000	4.1.1-RELEASE
411001		4.1.1-STABLE nach 4.1.1-RELEASE
420000	31. Oktober 2000	4.2-RELEASE

Wert	Datum	Release
420001	10. Januar 2001	4.2-STABLE nach Kombinaion von libgcc.a und libgcc_r.a und zugehörigen Änderungen der GCC-Bindungen.
430000	6. März 2001	4.3-RELEASE
430001	18. Mai 2001	4.3-STABLE nach der Einführung von wint_t.
430002	22. Juli 2001	4.3-STABLE nach dem Einpflegen der PCI Stromstatus-API.
440000	1. August 2001	4.4-RELEASE
440001	23. Oktober 2001	4.4-STABLE nach der Einführung von d_thread_t.
440002	4. November 2001	4.4-STABLE nach den Änderungen der mount-Struktur (betrifft Dateisystem-Kernelmodule).
440003	18. Dezember 2001	4.4-STABLE nachdem die Userland-Komponenten von smbfs importiert worden sind.
450000	20. Dezember 2001	4.5-RELEASE
450001	24. Februar 2002	4.5-STABLE nach der Umbenennung von Elementen der USB-Struktur.
450004	16. April 2002	4.5-STABLE nachdem die `sendmail_enable` rc.conf(5) Variable geändert worden ist, um den Wert `NONE` zu akzeptieren.
450005	27. April 2002	4.5-STABLE nachdem XFree86 4 als Standard zum Bauen der Pakete benutzt wird.
450006	1. Mai 2002	4.5-STABLE nach dem Reparieren des Empfangsfilters, welcher anfällig für einfache DoS-Attacken war.
460000	21. Juni 2002	4.6-RELEASE

Wert	Datum	Release
460001	21. Juni 2002	4.6-STABLE sendfile(2) repariert, um mit der Dokumentation übereinzustimmen, und nicht mehr die Anzahl der gesendeten Header mit der Anzahl der Daten, welche aus der Datei geschickt werden, gegenzurechnen.
460002	19. Juli 2002	4.6.2-RELEASE
460100	26. Juni 2002	4.6-STABLE
460101	26. Juni 2002	4.6-STABLE nach dem Einfließen von `sed -i` aus CURRENT.
460102	1. September 2002	4.6-STABLE nach dem Einfließen von vielen neuen pkg_install-Funktionen aus HEAD (HEAD = die aktuellste und letzte Version des Quellverzeichnisbaumes).
470000	8. Oktober 2002	4.7-RELEASE
470100	9. Oktober 2002	4.7-STABLE
470101	10. November 2002	Beginn von generierten *std{in,out,err}p Referenzen statt* sF. Dies ändert std{in,out,err} von einem Ausdruck während des Kompilierens zu einem Laufzeitausdruck.
470102	23. Januar 2003	4.7-STABLE nach dem Einfliessen von mbuf-Änderungen, um m_aux mbufs mit denen von m_tag zu ersetzen
470103	14. Februar 2003	4.7-STABLE erhält OpenSSL 0.9.7
480000	30. März 2003	4.8-RELEASE
480100	5. April 2003	4.8-STABLE
480101	22. Mai 2003	4.8-STABLE nachdem realpath(3) Thread-sicher gemacht wurde.

Wert	Datum	Release
480102	10. August 2003	4.8-STABLE Änderung der 3ware-API in twe.
490000	27. Oktober 2003	4.9-RELEASE
490100	27. Oktober 2003	4.9-STABLE
490101	8. Januar 2004	4.9-STABLE nachdem e_sid zu der Struktur kinfo_eproc hinzugefügt wurde.
490102	4. Februar 2004	4.9-STABLE nach dem Einfliessen der libmap-Funktionalität für rtld.
491000	25. Mai 2004	4.10-RELEASE
491100	1. Juni 2004	4.10-STABLE
491101	11. August 2004	4.10-STABLE nach dem Einfliessen von Revision 20040629 der Paket-Werkzeuge aus CURRENT.
491102	16. November 2004	4.10-STABLE nach der Fehlerbehebung in der VM, um das Freigeben von fiktiven Speicherseiten korrekt zu handhaben.
492000	17. Dezember 2004	4.11-RELEASE
492100	17. Dezember 2004	4.11-STABLE
492101	18. April 2006	4.11-STABLE nach dem Hinzufügen von libdata/ldconfig Verzeichnissen zu den mtree-Dateien.
500000	13. März 2000	5.0-CURRENT
500001	18. April 2000	5.0-CURRENT nach Hinzufügen von zusätzlichen Feldern in den ELF-Headern und Ändern der Methode zur ELF-Markierung von Binärdateien.
500002	2. Mai 2000	5.0-CURRENT nach kld-Metadaten Änderungen.
500003	18. Mai 2000	5.0-CURRENT nach buf/bio Änderungen.
500004	26. Mai 2000	5.0-CURRENT nach binutils Aktualisierung.

Wert	Datum	Release
500005	3. Juni 2000	5.0-CURRENT nach dem Einfliessen des libxpg4 Quelltextes in die libc und der Einführung der TASKQ-Schnittstelle.
500006	10. Juni 2000	5.0-CURRENT nach dem Hinzufügen der AGP-Schnittstellen.
500007	29. Juni 2000	5.0-CURRENT nach der Aktualisierung von Perl auf Version 5.6.0.
500008	7. Juli 2000	5.0-CURRENT nach der Aktualisierung des KAME-Quelltextes zu den 2000/07-Quellen.
500009	14. Juli 2000	5.0-CURRENT nach ether_ifattach() und ether_ifdetach() Änderungen.
500010	16. Juli 2000	5.0-CURRENT nachdem die mtree-Standards zurück zur ursprünglichen Variante geändert wurden; -L hinzugefügt, um Symlinks zu folgen.
500011	18. Juli 2000	5.0-CURRENT nachdem die kqueue-API geändert worden ist.
500012	2. September 2000	5.0-CURRENT nachdem setproctitle(3) von libutil nach libc verschoben worden ist.
500013	10. September 2000	5.0-CURRENT nach dem ersten SMPng-Commit.
500014	4. Januar 2001	5.0-CURRENT nachdem <sys/select.h> nach <sys/selinfo.h> verschoben worden ist.
500015	10. Januar 2001	5.0-CURRENT nach dem Kombinieren von libgcc.a und libgcc_r.a und damit verbundene Änderungen an GCC-Bindungen.

Wert	Datum	Release
500016	24. Januar 2001	5.0-CURRENT nach der Änderung das Zusammenbinden von libc und libc_r zu erlauben, womit die -pthread Option veraltet ist.
500017	18. Februar 2001	5.0-CURRENT nach dem Umschalten von struct ucred zu struct xucred, um die vom Kernel exportierte API für mount u.a.zu stabilisieren.
500018	24. Februar 2001	5.0-CURRENT nach dem Hinzufügen der CPUTYPE make Variable zum Kontrollieren von CPU-spezifischen Optimierungen.
500019	9. Juni 2001	5.0-CURRENT nach dem Verschieben von machine/ioctl_fd.h nach sys/fdcio.h
500020	15. Juni 2001	5.0-CURRENT nach der Umbenennung der locale-Namen.
500021	22. Juni 2001	5.0-CURRENT nach dem Bzip2-Import. Kennzeichnet auch, dass S/Key entfernt wurde.
500022	12. Juli 2001	5.0-CURRENT nach SSE Unterstützung.
500023	14. September 2001	5.0-CURRENT nach KSE-Meilenstein 2.
500024	1. Oktober 2001	5.0-CURRENT nach d_thread_t, und nachdem UUCP in die Ports verschoben worden ist.
500025	4. Oktober 2001	5.0-CURRENT nach Änderungen in der ABI bei der Weitergabe von Deskriptoren und Berechtigungen auf 64 Bit Plattformen.
500026	9. Oktober 2001	5.0-CURRENT nachdem XFree86 4 als Standard zum Erstellen der Pakete benutzt wird und die neue libc strnstr()-Funktion hinzugefügt wurde.

Wert	Datum	Release
500027	10. Oktober 2001	5.0-CURRENT nachdem die neue libc strcasestr()-Funktion hinzugefügt wurde.
500028	14. Dezember 2001	5.0-CURRENT nachdem die Userland-Komponenten von smbfs importiert wurden.
(nicht geändert)		5.0-CURRENT nachdem die neuen C99-Ganzzahlen mit spezifischer Breite hinzugefügt wurden.
500029	29. Januar 2002	5.0-CURRENT nachdem eine Änderung im Rückgabewert von sendfile(2) gemacht wurde.
500030	15. Februar 2002	5.0-CURRENT nach der Einführung des Types `fflags_t`, welches die passende Größe für Dateiflags hat.
500031	24. Februar 2002	5.0-CURRENT nach der Umbenennung der USB elements-Struktur.
500032	16. März 2002	5.0-CURRENT nach der Einführung von Perl 5.6.1.
500033	3. April 2002	5.0-CURRENT nachdem die `sendmail_enable` rc.conf(5) Variable geändert worden ist, um den Wert `NONE` zu akzeptieren.
500034	30. April 2002	5.0-CURRENT nachdem mtx_init() einen dritten Parameter entgegen nimmt.
500035	13. Mai 2002	5.0-CURRENT mit GCC 3.1.
500036	17. Mai 2002	5.0-CURRENT ohne Perl in /usr/src
500037	29. Mai 2002	5.0-CURRENT nach dem Hinzufügen von dlfunc(3)
500038	24. Juli 2002	5.0-CURRENT nachdem die Typen von einigen Elementen der sockbuf-Struktur geändert wurden und nachdem die Struktur neu geordnet wurde.

Wert	Datum	Release
500039	1. September 2002	5.0-CURRENT nach dem GCC 3.2.1 Import. Und auch nachdem die Header nicht mehr *BSD_FOO_T* sondern _FOO_T_DECLARED benutzen. Dieser Wert kann auch als konservative Schätzung für den Beginn der Unterstützung des bzip2(1) Pakets verwendet werden.
500040	20. September 2002	5.0-CURRENT nachdem verschiedene Änderungen an Plattenfunktionen gemacht wurden, um die Anhängigkeit von Interna der disklabel-Struktur zu entfernen.
500041	1. Oktober 2002	5.0-CURRENT nach dem Hinzufügen von getopt_long(3) zur libc.
500042	15. Oktober 2002	5.0-CURRENT nach der Aktualisierung von Binutils auf 2.13, bei denen die FreeBSD-Emulation, vec und das Ausgabeformat geändert wurden.
500043	1. November 2002	5.0-CURRENT nach dem Hinzufügen schwacher pthread_XXX Stubs zur libc, womit libXThrStub.so veraltet ist. 5.0-RELEASE.
500100	17. Januar 2003	5.0-CURRENT nach dem Erstellen des RELENG_5_0-Zweiges
500101	19. Februar 2003	<sys/dkstat.h> ist leer und sollte nicht inkludiert werden.
500102	25. Februar 2003	5.0-CURRENT nach der Änderung in der d_mmap_t-Schnittstelle.

Wert	Datum	Release
500103	26. Februar 2003	5.0-CURRENT nachdem taskqueue_swi geädert wurde, um ohne Giant zu arbeiten, und taskqueue_swi_giant hinzugefügt wurde, um Giant zu verwenden.
500104	27. Februar 2003	cdevsw_add() und cdevsw_remove() gibt es nicht länger. Auftauchen der MAJOR_AUTO-Allokationsmöglichkeit.
500105	4. März 2003	5.0-CURRENT nach der neuen cdevsw-Initialisierungsmethode.
500106	8. März 2003	devstat_add_entry() wurde durch devstat_new_entry() ersetzt.
500107	15. März 2003	Devstat Schnittstellenänderung; siehe sys/sys/param.h 1.149.
500108	15. März 2003	Token-Ring Schnittstellenänderungen.
500109	25. März 2003	Hinzufügen von vm_paddr_t.
500110	28. März 2003	5.0-CURRENT nachdem realpath(3) Thread-sicher gemacht wurde.
500111	9. April 2003	5.0-CURRENT nachdem usbhid(3) mit NetBSD synchronisiert wurde.
500112	17. April 2003	5.0-CURRENT nach der neuen NSS Implementierung und Hinzufügen der POSIX.1 getpw*_r, getgr*_r Funktionen.
500113	2. Mai 2003	5.0-CURRENT nach Entfernen des alten rc-Systems.
501000	4. Juni 2003	5.1-RELEASE.
501100	2. Juni 2003	5.1-CURRENT nach dem Erstellen des RELENG_5_1 Zweiges.

Wert	Datum	Release
501101	29. Juni 2003	5.1-CURRENT nachdem die Semantik von sigtimedwait(2) and sigwaitinfo(2) korrigiert wurden.
501102	3. Juli 2003	5.1-CURRENT nach dem Hinzufügen der lockfunc und lockfuncarg-Felder zu bus_dma_tag_create(9).
501103	31. Juli 2003	5.1-CURRENT nach der Integration des GCC 3.3.1-pre 20030711 Snapshots.
501104	5. August 2003	5.1-CURRENT 3ware-API Änderungen in twe.
501105	17. August 2003	5.1-CURRENT Unterstützung von dynamisch gebundenen /bin und /sbin und Verschieben von Bibliotheken nach /lib.
501106	8. September 2003	5.1-CURRENT nachdem im Kernel Unterstützung für Coda 6.x hinzugefügt wurden.
501107	17. September 2003	5.1-CURRENT nachdem die 16550 UART-Konstanten von <dev/sio/sioreg.h> nach <dev/ic/ns16550.h> verschoben wurden. Und nachdem die libmap Funktionalität vorbehaltlos vom rtld unterstützt wurde.
501108	23. September 2003	5.1-CURRENT nach Aktualisierung der PFIL_HOOKS API.
501109	27. September 2003	5.1-CURRENT nachdem kiconv(3) hinzugefügt wurde.
501110	28. September 2003	5.1-CURRENT nachdem der standardmäßige Ablauf von open und close in cdevsw geändert wurde.
501111	16. Oktober 2003	5.1-CURRENT nachdem das Layout von cdevsw geändert wurde.

Wert	Datum	Release
501112	16. Oktober 2003	5.1-CURRENT nach dem Hinzufügen von Mehrfachvererbung in kobj.
501113	31. Oktober 2003	5.1-CURRENT nach der if_xname Änderung in der Struktur ifnet
501114	16. November 2003	5.1-CURRENT nachdem /bin und /sbin geändert wurden, um sie dynamisch zu binden.
502000	7. Dezember 2003	5.2-RELEASE
502010	23. Februar 2004	5.2.1-RELEASE
502100	7. Dezember 2003	5.2-CURRENT nach dem Erstellen des RELENG_5_2-Zweiges.
502101	19. Dezember 2003	5.2-CURRENT nachdem die *cxa_atexit*/cxa_finalize Funktionen zur libc hinzugefügt wurden.
502102	30. Januar 2004	5.2-CURRENT nachdem die Standard-Thread Bibliothek von libc_r zu libpthread geändert wurde.
502103	21. Februar 2004	5.2-CURRENT nach dem Gerätetreiber API Megapatch.
502104	25. Februar 2004	5.2-CURRENT nachdem getopt_long_only() hinzugefügt wurde.
502105	5. März 2004	5.2-CURRENT nachdem NULL für C in ((void *)0) geändert wurde, was mehr Warnungen erzeugt.
502106	8. März 2004	5.2-CURRENT nachdem pf beim Bauen und Installieren mit eingebunden wird.
502107	10. März 2004	5.2-CURRENT nachdem time_t auf der sparc64-Plattform in einen 64-bit Wert geändert wurde.

Wert	Datum	Release
502108	12. März 2004	5.2-CURRENT nachdem sich die Unterstützung für den Intel C/C++-Compiler in einigen Headern und execve(2) geändert hat, um sich strikter an POSIX zu halten.
502109	22. März 2004	5.2-CURRENT nach der Einführung der bus_alloc_resource_any API
502110	27. März 2004	5.2-CURRENT nach dem Hinzufügen von UTF-8 locales
502111	11. April 2004	5.2-CURRENT nach dem Entfernen der getvfsent(3) API
502112	13. April 2004	5.2-CURRENT nach dem Hinzufügen der .warning Directive für make.
502113	4. Juni 2004	5.2-CURRENT nachdem ttyioctl() zwingend erforderlich für serielle Treiber gemacht wurde.
502114	13. Juni 2004	5.2-CURRENT nach dem Import des ALTQ-Frameworks.
502115	14. Juni 2004	5.2-CURRENT nachdem sema_timedwait(9) geändert wurde, 0 bei Erfolg und einen von 0 verschiedenen Fehlercode im Falle eines Fehlers zurückzuliefern.
502116	16. Juni 2004	5.2-CURRENT nach dem Ändern der Kernel Struktur dev_t, in ein Zeiger auf die Struktur cdev *
502117	17. Juni 2004	5.2-CURRENT nach dem Ändern der Kernelstruktur udev_t in dev_t.
502118	17. Juni 2004	5.2-CURRENT nachdem Unterstützung für CLOCK_VIRTUAL und CLOCK_PROF zu clock_gettime(2) und clock_getres(2) hinzugefügt wurde.

Wert	Datum	Release
502119	22. Juni 2004	5.2-CURRENT nachdem die Überprüfung des Klonens von Netzwerk-Schnittstellen geändert wurde.
502120	2. Juli 2004	5.2-CURRENT nach dem Einfliessen von Revision 20040629 der Paket-Werkzeuge.
502121	9. Juli 2004	5.2-CURRENT nachdem Bluetooth-Quelltext als nicht i386-spezifisch markiert wurde.
502122	11. Juli 2004	5.2-CURRENT nach der Einführung des KDB Debugger Frameworks, der Umwandlung des DDB in ein Backend und der Einführung des GDB-Backends.
502123	12. Juli 2004	5.2-CURRENT nachdem VFS_ROOT geändert wurde, eine Struktur thread als Argument zu aktzeptieren, wie vflush. Die Struktur kinfo_proc enthält nun einen Zeiger auf Benutzer Daten. Der Umstieg auf `xorg` als standardmäßige X Implementierung wurde auch zu dieser Zeit durchgeführt.
502124	24. Juli 2004	5.2-CURRENT nachdem die Art und Weise, wie rc.d-Skripte von Ports und Altlasten gestartet werden, getrennt wurde.
502125	28. Juli 2004	5.2-CURRENT nachdem die vorherige Änderung rückgängig gemacht wurde.
502126	31. Juli 2004	5.2-CURRENT nach dem Entfernen von kmem_alloc_pageable() und dem Import von GCC 3.4.2.
502127	2. August 2004	5.2-CURRENT nachdem die UMA Kernel API geändert wurde, um Konstruktoren und Initialisierungsmethoden zu erlauben fehlzuschlagen.

Wert	Datum	Release
502128	8. August 2004	5.2-CURRENT nach der Änderung in der vfs_mount Signatur sowie allgemeines Ersetzen von PRISON_ROOT durch SUSER_ALLOWJAIL in der suser(9) API.
503000	23. August 2004	5.3-BETA/RC vor der Änderung der pfil-API.
503001	22. September 2004	5.3-RELEASE
503100	16. Oktober 2004	5.3-STABLE nach dem Erstellen des RELENG_5_3-Zweiges.
503101	3. Dezember 2004	5.3-STABLE nach dem Hinzufügen von Fülloptionen im Stile der libc zu strftime(3).
503102	13. Februar 2005	5.3-STABLE nachdem OpenBSD's nc(1) von CURRENT importiert wurde.
503103	27. Februar 2005	5.4-PRERELEASE nach dem Einfliessen der Reparaturen aus CURRENT, in <src/include/stdbool.h> und <src/sys/i386/include/_types.h>, um die GCC-Kompatibilität des Intel C/C++-Compilers zu benutzen.
503104	28. Februar 2005	5.4-PRERELEASE nach dem Einfliessen der Änderung aus CURRENT in ifi_epoch statt der lokalen Zeit die Betriebszeit des Systems zu benutzen.
503105	2. März 2005	5.4-PRERELEASE nach dem Einfliessen der Reparaturen von EOVERFLOW in vswprintf(3) aus CURRENT.
504000	3. April 2005	5.4-RELEASE.
504100	3. April 2005	5.4-STABLE nach dem Erstellen des RELENG_5_4-Zweiges.
504101	11. Mai 2005	5.4-STABLE nach dem Vergrößern der standardmäßigen Stackgröße für Threads.

Wert	Datum	Release
504102	24. Juni 2005	5.4-STABLE nach dem Hinzufügen von sha256.
504103	3. Oktober 2005	5.4-STABLE nach dem Einfliessen von if_bridge aus CURRENT.
504104	13. November 2005	5.4-STABLE nach dem Einfliessen von bsdiff und portsnap aus CURRENT.
504105	17. Januar 2006	5.4-STABLE nach dem Einfliessen der Änderung von ldconfig_local_dirs aus CURRENT.
505000	12. Mai 2006	5.5-RELEASE.
505100	12. Mai 2006	5.5-STABLE nach dem Erstellen des RELENG_5_5-Zweiges.
600000	18. August 2004	6.0-CURRENT
600001	27. August 2004	6.0-CURRENT nach der festen Aktivierung von PFIL_HOOKS im Kernel.
600002	30. August 2004	6.0-CURRENT nach der anfänglichen Einführung von ifi_epoch zur Struktur if_data. Wurde nach ein paar Tagen wieder rückgängig gemacht. Benutzen Sie diesen Wert bitte nicht.
600003	8. September 2004	6.0-CURRENT nach dem erneuten Hinzufügen des Elements ifi_epoch zur Struktur if_data.
600004	29. September 2004	6.0-CURRENT nach dem Hinzufügen der Struktur inpcb als Argument in der pfil API.
600005	5. Oktober 2004	6.0-CURRENT nach dem Hinzufügen des "-d DESTDIR" Schalters zu newsyslog.
600006	4. November 2004	6.0-CURRENT nach dem Hinzufügen von Fülloptionen im Style der libc zu strftime(3).

Wert	Datum	Release
600007	12. Dezember 2004	6.0-CURRENT nach dem Hinzufügen von 802.11 Framework Neuerungen.
600008	25. Januar 2005	6.0-CURRENT Änderung an den VOP_*VOBJECT() Funktionen und Einführung des MNTK_MPSAFE Schalters für Dateisysteme, welche ohne Giant arbeiten.
600009	4. Februar 2005	6.0-CURRENT nach dem Hinzufügen von cpufreq Framework und Treibern.
600010	6. Februar 2005	6.0-CURRENT nachdem OpenBSD's nc(1) importiert wurde.
600011	12. Februar 2005	6.0-CURRENT nachdem der Anschein von `matherr()` Unterstützung in SVID2 entfernt wurde.
600012	15. Februar 2005	6.0-CURRENT nach dem Vergrößern der standardmäßigen Stackgröße für Threads.
600013	19. Februar 2005	6.0-CURRENT nach dem Einfliessen der Reparaturen in <src/include/stdbool.h> und <src/sys/i386/include/_types.h>, um die GCC-Kompatibilität des Intel C/C++-Compilers zu benutzen.
600014	21. Februar 2005	6.0-CURRENT nachdem die Überprüfungen auf EOVERFLOW in vswprintf(3) korrigiert wurden.
600015	25. Februar 2005	6.0-CURRENT nach dem Einfliessen der Änderung, in ifi_epoch, statt der lokalen Zeit, die Betriebzeit des Systems zu benutzen.
600016	26. Februar 2005	6.0-CURRENT nachdem das Format von LC_CTYPE auf der Festplatte verändert wurde.

Wert	Datum	Release
600017	27. Februar 2005	6.0-CURRENT nachdem das Format der NLS-Kataloge auf der Festplatte verändert wurde.
600018	27. Februar 2005	6.0-CURRENT nachdem das Format von LC_COLLATE auf der Festplatte verändert wurde.
600019	28. Februar 2005	Installation der acpica Include-Dateien in /usr/include.
600020	9. März 2005	Hinzufügen des MSG_NOSIGNAL Schalters zur send(2) API.
600021	17. März 2005	Hinzufügen von Feldern zu cdevsw
600022	21. März 2005	gtar wurde aus dem Basissystem entfernt.
600023	13. April 2005	Die Optionen LOCAL_CREDS, LOCAL_CONNWAIT für Sockets wurde zu unix(4) hinzugefügt.
600024	19. April 2005	hwpmc(4) und zugehörige Werkzeuge wurden zu 6.0-CURRENT hinzugefügt.
600025	26. April 2005	Die Struktur icmphdr wurden zu 6.0-CURRENT hinzugefügt.
600026	3. Mai 2005	pf Aktualisierung auf 3.7.
600027	6. Mai 2005	Kernel libalias und ng_nat wurden eingeführt.
600028	13. Mai 2005	POSIX ttyname_r(3) wurde über unistd.h und libc zur Verfügung gestellt.
600029	29. Mai 2005	6.0-CURRENT nachdem libpcap zu Version v0.9.1 alpha 096 aktualisiert wurde.
600030	5. Juni 2005	6.0-CURRENT nach dem Import von NetBSDs if_bridge(4).
600031	10. Juni 2005	6.0-CURRENT nachdem die Struktur ifnet aus dem Treiber softcs herausgelöst wurde.
600032	11. Juli 2005	6.0-CURRENT nach dem Import von libpcap v0.9.1.

Wert	Datum	Release
600033	25. Juli 2005	6.0-STABLE nachdem die Versionen aller gemeinsam genutzten Bibliotheken, welche seit RELENG_5 nicht geändert wurden, erhöht wurden.
600034	13. August 2005	6.0-STABLE nachdem das Argument credential zu der dev_clone-Ereignisbehandlung hinzugefügt wurde. 6.0-RELEASE.
600100	1. November 2005	6.0-STABLE nach dem Erstellen des 6.0-RELEASE-Zweiges.
600101	21. Dezember 2005	6.0-STABLE nach dem Aufnehmen von Skripten aus den local_startup-Verzeichnissen in rcorder(8) des Basissystems.
600102	30. Dezember 2005	6.0-STABLE nach dem Aktualisieren der ELF-Typen und Konstanten.
600103	15. Januar 2006	6.0-STABLE nach dem Einfliessen der pidfile(3)-API aus CURRENT.
600104	17. Januar 2006	6.0-STABLE nach dem Einfliessen der Änderung von ldconfig_local_dirs aus CURRENT.
600105	26. Februar 2006	6.0-STABLE nach der NLS-Katalogunterstützung von csh(1).
601000	6. Mai 2006	6.1-RELEASE
601100	6. Mai 2006	6.1-STABLE nach 6.1-RELEASE.
601101	22. Juni 2006	6.1-STABLE nach dem Import von csup.
601102	11. Juli 2006	6.1-STABLE nach der iwi(4)-Aktualisierung.

Wert	Datum	Release
601103	17. Juli 2006	6.1-STABLE nach der Aktualisierung der Namensauflösung zu BIND9 und Aufnahme der ablaufinvarianten Versionen der netdb-Funktionen.
601104	8. August 2006	6.1-STABLE nachdem Unterstützung für DSO (dynamic shared objects - gemeinsam genutzte, dynamische Objekte) in OpenSSL aktiviert wurde.
601105	2. September 2006	6.1-STABLE nachdem 802.11 Reparaturen die API der IEEE80211_IOC_STA_INFO ioctl geändert haben.
602000	15. November 2006	6.2-RELEASE
602100	15. September 2006	6.2-STABLE nach 6.2-RELEASE.
602101	12. Dezember 2006	6.2-STABLE nach dem Hinzufügen der Wi-Spy Eigenart.
602102	28. Dezember 2006	6.2-STABLE nachdem pci_find_extcap() hinzugefügt wurde.
602103	16. Januar 2007	6.2-STABLE nach dem Einpflegen der dlsym Änderung aus CURRENT, ein angefordertes Symbol sowohl in der spezifizierten dso, als auch in den impliziten Abhängigkeiten nachzuschlagen.
602104	28. Januar 2007	6.2-STABLE nach dem Einpflegen von ng_deflate(4) und ng_pred1(4) netgraph Knoten und neuen Kompressions- und -Verschlüsselungmodi für den ng_ppp(4) Knoten aus CURRENT.

Wert	Datum	Release
602105	20. Februar 2007	6.2-STABLE nach dem Einpflegen der BSD lizensierten Version von gzip(1), welche von NetBSD portiert wurde aus CURRENT.
602106	31. März 2007	6.2-STABLE nach dem Einpflegen der PCI MSI und MSI-X Unterstützung aus CURRENT.
602107	6. April 2007	6.2-STABLE nach dem Einpflegen von ncurses 5.6 und Unterstützung für Multibyte-Zeichen aus CURRENT.
602108	11. April 2007	6.2-STABLE nach dem Einpflegen des 'SG' Peripheriegerätes aus CURRENT in CAM, welches einen Teil der SCSI SG passthrough Geräte API von Linux enthält.
602109	17. April 2007	6.2-STABLE nach dem Einpflegen von readline 5.2 Patchset 002 aus CURRENT.
602110	2. Mai 2007	6.2-STABLE nach dem Einpflegen von pmap_invalidate_cache(), pmap_change_attr(), pmap_mapbios(), pmap_mapdev_attr(), und pmap_unmapbios() für amd64 und i386 aus CURRENT.
602111	11. Juni 2007	6.2-STABLE nach dem Einpflegen von BOP_BDFLUSH aus CURRENT und dem daraus resultierendem Bruch mit dem Dateisystemmodul KBI.
602112	21. September 2007	6.2-STABLE nach dem Einpflegen von libutil(3) aus CURRENT.

Wert	Datum	Release
602113	25. Oktober 2007	6.2-STABLE, nach der Trennung in "wide und single byte ctype". Neu kompilierte Binärdateien, die ctype.h referenzieren, erfordern möglicherweise ein neues Symbol, __mb_sb_limit, das auf älteren Systemen nicht verfügbar ist.
602114	30. Oktober 2007	6.2-STABLE, nachdem die ctype ABI-Aufwärtskompatibilität wiederhergestellt wurde.
602115	21. November 2007	FreeBSD 6.2-STABLE nach der Entfernung/Eliminierung der wide und single Byte ctype-Trennung
603000	25. November 2007	6.3-RELEASE
603100	25. November 2007	6.3-STABLE nach 6.3-RELEASE.
603101	7. Dezember 2007	6.3-STABLE, nachdem der Support für den Multibyte-Datentyp im Bit-Makro gefixt wurde.
603102	24. April 2008	6.3-STABLE nach Hinzufügen von l_sysid zu struct flock.
603103	27. Mai 2008	6.3-STABLE nach Einfließen der `memrchr`-Funktion.
603104	15. Juni 2008	6.3-STABLE nach Übernahme der Unterstützung von `:u` als Variablenwandler in make(1).
604000	4. Oktober 2008	6.4-RELEASE
604100	4. Oktober 2008	6.4-STABLE nach 6.4-RELEASE.
700000	11. Juli 2005	7.0-CURRENT.
700001	23. Juli 2005	7.0-CURRENT nachdem die Versionen aller gemeinsam genutzten Bibliotheken, welche seit RELENG_5 nicht geändert wurden, erhöht wurden.
700002	13. August 2005	7.0-CURRENT nachdem ein Berechtigungs-Argument zur dev_clone-Ereignisroutine hinzugefügt wurde.

Wert	Datum	Release
700003	25. August 2005	7.0-CURRENT nachdem memmem(3) zur libc hinzugefügt wurde.
700004	30. Oktober 2005	7.0-CURRENT nachdem die Argumente der Kernelfunktion solisten(9) modifiziert wurden, um einen Backlog-Parameter (Anzahl der maximalen wartenden Verbindungen) zu akzeptieren.
700005	11. November 2005	7.0-CURRENT nachdem IFP2ENADDR() geändert wurde, einen Zeiger auf IF_LLADDR() zurückzugeben.
700006	11. November 2005	7.0-CURRENT nach dem Hinzufügen des `if_addr` -Elements zur Struktur `ifnet` und dem Entfernen von IFP2ENADDR().
700007	2. Dezember 2005	7.0-CURRENT nach dem Aufnehmen von Skripten aus den local_startup Verzeichnissen in rcorder(8) des Basissystems.
700008	5. Dezember 2005	7.0-CURRENT nach dem Entfernen der MNT_NODEV mount-Option.
700009	19. Dezember 2005	7.0-CURRENT nach ELF-64 Typen Änderungen und Symbol Versionierung.
700010	20. Dezember 2005	7.0-CURRENT nach Hinzufügen der hostb und vgapci Treiber, Hinzufügen von pci_find_extcap() und Änderung der AGP Treiber die Apertur nicht länger abzubilden.
700011	31. Dezember 2005	7.0-CURRENT nachdem auf allen Plattformen außer Alpha tv_sec in time_t umgewandelt wurde.
700012	8. Januar 2006	7.0-CURRENT nach Änderung von ldconfig_local_dirs.

Wert	Datum	Release
700013	12. Januar 2006	7.0-CURRENT nach Änderung in /etc/rc.d/abi um /compat/linux/etc/ld.so.cache als Symlink in ein schreibgeschütztes Dateisystem zu unterstützen.
700014	26. Januar 2006	7.0-CURRENT nach pts Import.
700015	26. März 2006	7.0-CURRENT nach Einführung von Version 2 der hwpmc(4)'s ABI.
700016	22. April 2006	7.0-CURRENT nach dem Hinzufügen von fcloseall(3) zur libc.
700017	13. Mai 2006	7.0-CURRENT nach dem Entfernen von ip6fw.
700018	15. Juli 2006	7.0-CURRENT nach dem Import von snd_emu10kx.
700019	29. Juli 2006	7.0-CURRENT nach dem Import von OpenSSL 0.9.8b.
700020	3. September 2006	7.0-CURRENT nach dem Hinzufügen der bus_dma_get_tag-Funktion
700021	4. September 2006	7.0-CURRENT nach dem Import von libpcap 0.9.4 und tcpdump 3.9.4.
700022	9. September 2006	7.0-CURRENT nach der dlsym Änderung, ein angefordertes Symbol sowohl in der spezifizierten dso, als auch in den impliziten Abhängigkeiten nachzuschlagen.
700023	23. September 2006	7.0-CURRENT nach dem Hinzufügen neuer Sound-IOCTLs für die OSSv4-Mixer-API.
700024	28. September 2006	7.0-CURRENT nach dem Import von OpenSSL 0.9.8d.
700025	11. November 2006	7.0-CURRENT nach dem Hinzufügen der libelf.

Wert	Datum	Release
700026	26. November 2006	7.0-CURRENT nach größeren Änderungen an den Sound sysctls.
700027	30. November 2006	7.0-CURRENT nach dem Hinzufügen der Wi-Spy-Eigenart.
700028	15. Dezember 2006	7.0-CURRENT nach dem Hinzufügen von sctp-Aufrufen zur libc.
700029	26. Januar 2007	7.0-CURRENT nach dem Ersetzen von GNU gzip(1) durch eine von NetBSD portierte Version, die unter BSD-Lizenz steht.
700030	7. Februar 2007	7.0-CURRENT nach dem Entfernen der IPIP Tunnelkapselung (VIFF_TUNNEL) aus dem IPv4 Multicast-Forwarding-Quelltext.
700031	23. Februar 2007	7.0-CURRENT nach den Modifizierungen an bus_setup_intr() (newbus).
700032	2. März 2007	7.0-CURRENT nach der Aufnahme der Firmware für ipw(4) und iwi(4).
700033	9. März 2007	7.0-CURRENT nach Unterstützung für Multibyte-Zeichen.
700034	19. März 2007	7.0-CURRENT nach Änderungen, wie insmntque(), getnewvnode() und vfs_hash_insert() arbeiten.
700035	26. März 2007	7.0-CURRENT nach Hinzufügen eines Benachrichtigungsmechanismus für CPU Frequenzänderungen.
700036	6. April 2007	7.0-CURRENT nach dem Import des ZFS Dateisystemes.

Wert	Datum	Release
700037	8. April 2007	7.0-CURRENT nach dem Einpflegen des 'SG' Peripheriegerätes in CAM, welches einen Teil der SCSI SG passthrough Geräte API von Linux enthält.
700038	30. April 2007	7.0-CURRENT nachdem getenv(3), putenv(3), setenv(3) und unsetenv(3) geändert wurden, um POSIX konform zu sein.
700039	1. Mai 2007	7.0-CURRENT nachdem die Änderungen von 700038 rückgängig gemacht wurden.
700040	10. Mai 2007	7.0-CURRENT nach dem Hinzufügen von flopen(3) zur libutil.
700041	13. Mai 2007	7.0-CURRENT nachdem Symbol Versionierung aktiviert und die standardmäßige Thread-Bibliothek zu libthr geändert wurde.
700042	19. Mai 2007	7.0-CURRENT nach dem Import von GCC 4.2.0.
700043	21. Mai 2007	7.0-CURRENT nachdem die Versionen aller Shared-Libraries, welche seit RELENG_6 nicht geändert wurden, erhöht worden sind.
700044	7. Juni 2007	7.0-CURRENT nachdem das Argument für vn_open()/VOP_OPEN() vom Dateideskriptorindex zur Struktur file * geädert wurde.
700045	10. Juni 2007	7.0-CURRENT nachdem pam_nologin(8) geädert wurde, eine Kontoverwaltungs-Funktion statt einer Authentifizierungsfunktion für das PAM-Framework zur Verfügung zu stellen.

Wert	Datum	Release
700046	11. Juni 2007	7.0-CURRENT nach aktualisierter 802.11 wireless Unterstützung.
700047	11. Juni 2007	7.0-CURRENT, nachdem TCP-LRO-Schnittstellen-Ressourcen hinzugefügt wurden.
700048	12. Juni 2007	7.0-CURRENT, nachdem die RFC 3678 API-Unterstützung zum IPv4-Stack hinzugefügt wurde. Veraltetes RFC 1724-Verhalten des IP_MULTICAST_IF ioctl wurde entfernt; 0.0.0.0/8 darf nicht länger als Schnittstellen-Index benutzt werden. Stattdessen sollte die Struktur ipmreqn verwendet werden.
700049	3. Juli 2007	7.0-CURRENT, nachdem pf von OpenBSD 4.1 importiert wurde
(nicht geändert)		7.0-CURRENT, nachdem die IPv6-Unterstützung um FAST_IPSEC erweitert, KAME IPSEC entfernt und FAST_IPSEC in IPSEC umbenannt wurde.
700050	4. Juli 2007	7.0-CURRENT, nachdem Aufrufe von setenv/putenv/usw. von der traditionellen BSD-Art und Weise nach POSIX konvertiert wurden.
700051	4. Juli 2007	7.0-CURRENT, nachdem neue Systemaufrufe (mmap/lseek/usw.) implementiert wurden.
700052	6. Juli 2007	7.0-CURRENT, nachdem die I4B-Header nach include/i4b verschoben wurden.
700053	30. September 2007	7.0-CURRENT, nachdem die Unterstützung für PCI Domänen hinzugefügt wurde.
700054	25. Oktober 2007	7.0-CURRENT, nach der Trennung in "wide und single byte ctype".

Wert	Datum	Release
700055	28. Oktober 2007	7.0-RELEASE sowie 7.0-CURRENT, nachdem die ABI-Abwärtskompatibilität für die FreeBSD 4/5/6-Versionen der PCIOCGETCONF-, PCIOCREAD- sowie PCIOCWRITE IOCTLs hinzugefügt wurde. Damit verbunden war, dass die ABI der PCIOCGETCONF IOCTL erneut deaktiviert werden musste.
700100	22. Dezember 2007	7.0-STABLE nach 7.0-RELEASE.
700101	8. Februar 2008	7.0-STABLE nach Einführung von m_collapse().
700102	30. März 2008	7.0-STABLE nach Einfließen von kdb_enter_why().
700103	10. April 2008	7.0-STABLE nach Hinzufügen von l_sysid zu struct flock.
700104	11. April 2008	7.0-STABLE nach Übernahme von procstat(1).
700105	11. April 2008	7.0-STABLE nach Einführung von umtx-Features.
700106	15. April 2008	7.0-STABLE nach Hinzufügen der Unterstützung von write(2) zu psm(4).
700107	20. April 2008	7.0-STABLE nach Hinzufügen des Befehls F_DUP2FD zu fcntl(2).
700108	5. Mai 2008	7.0-STABLE nach einigen Änderungen an lockmgr(9), welche die Einbindung von sys/lock.h zur Verwendung von lockmgr(9) voraussetzen.
700109	27. Mai 2008	7.0-STABLE nach Einfließen der `memrchr`-Funktion.
700110	5. August 2008	7.0-STABLE nach Einführung eines Clients für den Kernel NFS lockd.

Wert	Datum	Release
700111	20. August 2008	7.0-STABLE nach Hinzufügen einer Unterstützung von physisch fortlaufender Jumbo Frames.
700112	27. August 2008	7.0-STABLE nach Einfließen einer Kernelunterstützung für DTrace.
701000	25. November 2008	7.1-RELEASE
701100	25. November 2008	7.1-STABLE nach 7.1-RELEASE.
701101	10. Januar 2009	7.1-STABLE nach Übernahme von `strndup`.
701102	17. Januar 2009	7.1-STABLE nach Hinzufügen einer Unterstützung von cpuctl(4).
701103	7. Februar 2009	7.1-STABLE nach Einfließen der Unterstützung von Jails mit keinen oder mehreren IPv4- /IPv6-Adressen.
701104	14. Februar 2009	7.1-STABLE, nachdem der Besitzer des Suspend in struct mount gespeichert wird und die Funktion vfs_susp_clean in struct vfsops aufgenommen ist.
701105	12. März 2009	7.1-STABLE nach der inkompatiblen Änderung am sysctl kern.ipc.shmsegs, um die Anforderung größerer Segmente von gemeinsam genutzten SysV-Speicher auf 64bit-Architekturen zu erlauben.
701106	14. März 2009	7.1-STABLE nach der Übernahme einer Fehlerbehebung für Warteoperationen, die POSIX-Semaphore verwenden.
702000	15. April 2009	7.2-RELEASE
702100	15. April 2009	7.2-STABLE nach 7.2-RELEASE.

Wert	Datum	Release
702101	15. Mai 2009	7.2-STABLE, nachdem ichsmb(4) dahingehend geändert wurde, dass es links-ausgerichtete Adressierung von Slaves verwendet, um anderen SMBus-Kontrollertreibern zu entsprechen.
702102	28. Mai 2009	7.2-STABLE nach dem Einfließen der Funktion `fdopendir`.
702103	06. Juni 2009	7.2-STABLE nach dem Einfließen von PmcTools.
702104	14. Juli 2009	7.2-STABLE nach dem Einfließen des Systemaufrufs `closefrom`.
702105	31. Juli 2009	7.2-STABLE nach dem Einfließen der Änderung an der SYSVIPC-ABI.
702106	14. September 2009	7.2-STABLE nach dem Einfließen der PAT-Verbesserungen für x86-Prozessoren sowie dem Hinzufügen von d_mmap_single() und des VM-Objekttyps für scatter/gather-Listen.
703000	9. Februar 2010	7.3-RELEASE
703100	9. Februar 2010	7.3-STABLE nach 7.3-RELEASE.
704000	22. Dezember 2010	7.4-RELEASE
704100	22. Dezember 2010	7.4-STABLE, nachdem 7.4-RELEASE erzeugt wurde.
800000	11. Oktober 2007	8.0-CURRENT. Nach der Trennung in "wide und single byte ctype".
800001	16. Oktober 2007	8.0-CURRENT, nachdem libpcap 0.9.8 und tcpdump 3.9.8 importiert wurden.
800002	21. Oktober 2007	8.0-CURRENT, nachdem kthread_create() und Konsorten in kproc_create() usw. umbenannt wurden.

Wert	Datum	Release
800003	24. Oktober 2007	8.0-CURRENT, nachdem die ABI-Abwärtskompatibilität für die FreeBSD 4/5/6-Versionen der PCIOCGETCONF-, PCIOCREAD- sowie PCIOCWRITE IOCTLs hinzugefügt wurde. Damit verbunden war, dass die ABI der PCIOCGETCONF IOCTL erneut deaktiviert werden musste.
800004	12. November 2007	8.0-CURRENT, nachdem der agp(4) Treiber verschoben wurde von src/sys/pci nach src/sys/dev/agp.
800005	4. Dezember 2007	8.0-CURRENT nach Änderungen am Jumbo Frame Allocator.
800006	7. Dezember 2007	8.0-CURRENT, nach dem Hinzufügen der callgraph capture Funktionalität zu hwpmc(4).
800007	25. Dezember 2007	8.0-CURRENT nach dem Hinzufügen von "why" als Argument in kdb_enter().
800008	28. Dezember 2007	8.0-CURRENT nach Entfernen der Option LK_EXCLUPGRADE.
800009	9. Januar 2008	8.0-CURRENT nach Einführung von lockmgr_disown(9)
800010	10. Januar 2008	8.0-CURRENT nach Änderungen am vn_lock(9)-Prototyp.
800011	13. Januar 2008	8.0-CURRENT nach Änderungen an den Prototypen von VOP_LOCK(9) und VOP_UNLOCK(9).
800012	19. Januar 2008	8.0-CURRENT nach Einführung von lockmgr_recursed(9), BUF_RECURSED(9) und BUF_ISLOCKED(9) sowie Entfernung von `BUF_REFCNT()`.
800013	23. Januar 2008	8.0-CURRENT nach Einführung der "ASCII"-Kodierung.

Wert	Datum	Release
800014	24. Januar 2008	8.0-CURRENT nach Änderungen am lockmgr(9)-Prototyp und Entfernung von `lockcount()` sowie `LOCKMGR_ASSERT()`.
800015	26. Januar 2008	8.0-CURRENT nach Erweiterung der Datentypen der fts(3) -Strukturen.
800016	1. Februar 2008	8.0-CURRENT nach Hinzufügen eines neuen Parameters zu MEXTADD(9).
800017	6. Februar 2008	8.0-CURRENT nach Einführung der Optionen LK_NODUP und LK_NOWITNESS in die lockmgr(9)-Umgebung.
800018	8. Februar 2008	8.0-CURRENT nach Hinzufügen von m_collapse.
800019	9. Februar 2008	8.0-CURRENT nach Hinzufügen einer Arbeits-, Wurzel- und Jailverzeichnisunterstützung zur sysctl-Variable kern.proc.filedesc.
800020	13. Februar 2008	8.0-CURRENT nach Einführung der Funktionen lockmgr_assert(9) und `BUF_ASSERT`.
800021	15. Februar 2008	8.0-CURRENT nach Einführung von lockmgr_args(9) und Entfernung der Option LK_INTERNAL.
800022	(zurückgezogen)	8.0-CURRENT nach Setzen von BSD ar(1) als Systemstandard.
800023	25. Februar 2008	8.0-CURRENT nach Prototypenänderungen an lockstatus(9) und VOP_ISLOCKED(9);, eigens zur Abschaffung des Parameters `struct thread`.

Wert	Datum	Release
800024	1. März 2008	8.0-CURRENT nach Beseitigung der Funktionen `lockwaiters` und `BUF_LOCKWAITERS`, Änderung des Rückgabewerts der Funktion `brelvp` von void nach int sowie Einführung neuer Optionen für lockinit(9).
800025	8. März 2008	8.0-CURRENT nach Hinzufügen des Kommandos F_DUP2FD zu fcntl(2).
800026	12. März 2008	8.0-CURRENT nach Änderung des Parameters für die Priorität an cv_broadcastpri, sodass 0 für keine Priorität steht.
800027	24. März 2008	8.0-CURRENT nach Änderung der Monitoring ABI von BPF, als Zero-Copy Puffer hinzugefügt wurden.
800028	26. März 2008	8.0-CURRENT nach Hinzufügen von l_sysid zu struct flock.
800029	28. März 2008	8.0-CURRENT nach Wiedereingliederung der Funktion `BUF_LOCKWAITERS` und Hinzufügen von lockmgr_waiters(9).
800030	1. April 2008	8.0-CURRENT nach Einführung der Funktionen rw_try_rlock(9) und rw_try_wlock(9).
800031	6. April 2008	8.0-CURRENT nach Einführung der Funktionen `lockmgr_rw` und `lockmgr_args_rw`.
800032	8. April 2008	8.0-CURRENT nach Implementierung des Systemaufrufs openat und seiner Verwandten, Einführung der Option O_EXEC in open(2) und Bereitstellung der entsprechenden Systemaufrufe innerhalb der Linux®-Kompatibilitätsumgebung.

Wert	Datum	Release
800033	8. April 2008	8.0-CURRENT nach Hinzufügen der Unterstützung von write(2) in der nativen Operationsebene von psm(4). Es können nun beliebig Kommandos nach /dev/psm%d geschrieben und der Status dann von dort gelesen werden.
800034	10. April 2008	8.0-CURRENT nach Einführung der Funktion `memrchr`.
800035	16. April 2008	8.0-CURRENT nach Einführung der Funktion `fdopendir`.
800036	20. April 2008	8.0-CURRENT nach Umstellung des Standards 802.11 auf Unterstützung von Multi-BSS (auch vaps).
800037	9. Mai 2008	8.0-CURRENT nach Hinzufügen einer Unterstützung für Multi Routing-Tabellen (siehe setfib(1), setfib(2)).
800038	26. Mai 2008	8.0-CURRENT nach Entfernen von netatm und ISDN4BSD sowie dem Hinzufügen der Compact C Type (CTF)-Tools.
800039	14. Juni 2008	8.0-CURRENT nach Entfernen von sgtty.
800040	26. Juni 2008	8.0-CURRENT nach Einführung eines Clients für den Kernel NFS lockd.
800041	22. Juli 2008	8.0-CURRENT nach Hinzufügen von arc4random_buf(3) und arc4random_uniform(3).
800042	8. August 2008	8.0-CURRENT nach Hinzufügen von cpuctl(4).
800043	13. August 2008	8.0-CURRENT nach Änderung von bpf(4) zur Verwendung einer einzelnen Gerätedatei anstatt von Klonierung.

Wert	Datum	Release
800044	17. August 2008	8.0-CURRENT nach Übernahme des ersten Teils aus dem vimage-Projekt durch Erweitern globaler Variablen um den Präfix V_. Zukünftig werden die virtualisierten Variablen dann mit Hilfe von Makros in ihre globalen Namen aufgelöst.
800045	20. August 2008	8.0-CURRENT nach Eingliederung des MPSAFE TTY-Layers, einschließlich Änderungen an diversen Treibern und Werkzeugen, die mit ihm kommunizieren.
800046	8. September 2008	8.0-CURRENT nach Abschottung der GDT pro CPU auf der AMD64-Architektur.
800047	10. September 2008	8.0-CURRENT nach Entfernen von VSVTX, VSGID und VSUID.
800048	16. September 2008	8.0-CURRENT nach Anpassung des Codes für Kernel NFS mount, sodass einzelne Mountoptionen im Parameter struct iovec an nmount() akzeptiert werden und nicht nur ein großes struct nfs_args.
800049	17. September 2008	8.0-CURRENT nach Entfernen von suser(9) und suser_cred(9).
800050	20. Oktober 2008	8.0-CURRENT nach API-Änderungen im Umgang mit dem Buffer Cache.
800051	23. Oktober 2008	8.0-CURRENT nach Entfernen der Makros MALLOC(9) und FREE(9).
800052	28. Oktober 2008	8.0-CURRENT nach Einführung von accmode_t und Umbennung des Parameters a_mode an VOP_ACCESS nach a_accmode.

Wert	Datum	Release
800053	2. November 2008	8.0-CURRENT nach Änderung des Prototyps von vfs_busy(9) und Einführung der Optionen MBF_NOWAIT sowie MBF_MNTLSTLOCK.
800054	22. November 2008	8.0-CURRENT nach Hinzufügen von Funktionen im Bereich buf_ring, Memory Barriers und ifnet, um mehrere Sendeschlangen auf Hardwareebene für Karten zu ermöglichen, die dies unterstützen, sowie einer Ring Buffer-Implementierung ohne Lock, um Treibern zu ermöglichen, Paketschlangen effizienter zu verwalten.
800055	27. November 2008	8.0-CURRENT nach Hinzufügen einer Unterstützung für Intel® Core, Core2 und Atom zu hwpmc(4).
800056	29. November 2008	8.0-CURRENT nach Einführung von Jails mit mehreren oder gar keinen IPv4-/IPv6-Adressen.
800057	1. Dezember 2008	8.0-CURRENT nach Wechsel zum ath_hal Quellcode.
800058	12. Dezember 2008	8.0-CURRENT nach Einführung der Funktion VOP_VPTOCNP.
800059	15. Dezember 2008	8.0-CURRENT gliedert das neue ARPv2 ein.
800060	19. Dezember 2008	8.0-CURRENT nach Hinzufügen von makefs.
800061	15. Januar 2009	8.0-CURRENT nach Umsetzung von TCP Appropriate Byte Counting.
800062	28. Januar 2009	8.0-CURRENT nach Entfernen von minor(), minor2unit(), unit2minor() usw.

Wert	Datum	Release
800063	18. Februar 2009	8.0-CURRENT nach Änderung der GENERIC-Konfiguration zur Verwendung des USB2-Stack und Hinzufügen von fdevname(3).
800064	23. Februar 2009	8.0-CURRENT, nachdem der USB2-Stack nach dev/usb verschoben wurde, um es zu ersetzen.
800065	26. Februar 2009	8.0-CURRENT nach Umbenennen aller Funktionen in libmp(3).
800066	27. Februar 2009	8.0-CURRENT nach Anpassung des devfs-Verhaltens im Zusammenhang mit USB.
800067	28. Februar 2009	8.0-CURRENT nach Hinzufügen von getdelim(), getline(), stpncpy(), strnlen(), wcsnlen(), wcscasecmp() und wcsncasecmp().
800068	2. März 2009	8.0-CURRENT nach Umbenennen der Geräteklasse ushub in uhub.
800069	9. März 2009	8.0-CURRENT nach Umbenennen von libusb20.so.1 in libusb.so.1.
800070	9. März 2009	8.0-CURRENT nach der Einführung von IGMPv3 und Source-Specific-Multicast (SSM) in den IPv4-Stack.
800071	14. März 2009	8.0-CURRENT nach der Anpassung von gcc zur Verwendung der C99-Inline-Semantik in den Modi c99 und gnu99.
800072	15. März 2009	8.0-CURRENT, nachdem die Option IFF_NEEDSGIANT entfernt wurde; Netzwerktreiber, die nicht MPSAFE sind, werden nicht mehr unterstützt.

Wert	Datum	Release
800073	18. März 2009	8.0-CURRENT, nachdem die dynamische Ersetzung von Zeichenkettenkürzeln für rpath und benötigte Pfade implementiert wurde.
800074	24. März 2009	8.0-CURRENT nach dem Einfließen von tcpdump 4.0.0 und libpcap 1.0.0.
800075	6. April 2009	8.0-CURRENT, nachdem die Deklarationen von struct vnet_net, struct vnet_inet und struct vnet_ipfw geändert wurden.
800076	9. April 2009	8.0-CURRENT nach dem Hinzufügen von Laufzeitprofilen in dummynet.
800077	14. April 2009	8.0-CURRENT nach dem Entfernen von VOP_LEASE() und vop_vector.vop_lease.
800078	15. April 2009	8.0-CURRENT, nachdem die Felder aus struct rt_weight zu struct rt_metrics und struct rt_metrics_lite hinzugefügt wurden, wobei die Deklaration von struct rt_metrics_lite geändert wurde. RTM_VERSION wurde hochgezählt (zurückgezogen).
800079	15. April 2009	8.0-CURRENT, nachdem Pointer auf struct llentry zu struct route und struct route_in6 hinzugefügt wurden.
800080	15. April 2009	8.0-CURRENT nach Änderung der Deklaration von struct inpcb.
800081	19. April 2009	8.0-CURRENT nach Änderung der Deklaration von struct malloc_type.

Wert	Datum	Release
800082	21. April 2009	8.0-CURRENT nach Änderung der Deklaration von struct ifnet und Hinzufügen von if_ref() und if_rele() zur Verwaltung von Referenzen auf ifnet.
800083	22. April 2009	8.0-CURRENT nach der Implementierung einer systemnahen Bluetooth-HCI-API.
800084	29. April 2009	8.0-CURRENT nach Änderungen an IPv6-SSM und MLDv2.
800085	30. April 2009	8.0-CURRENT, nachdem der Bau von VIMAGE-Kernel mit einem aktiven Image unterstützt wird.
800086	8. Mai 2009	8.0-CURRENT nach Hinzufügen der Unterstützung für Eingabezeilen mit beliebiger Länge durch patch(1).
800087	11. Mai 2009	8.0-CURRENT nach einigen Änderungen im Zusammenhang mit dem VFS-KPI. Der Thread-Parameter wurde von den FSD-Teilen des VFS entfernt. `VFS_*`-Funktionen benötigen den Kontext nicht mehr, da er sich immer auf `curthread` bezieht. In wenigen Sonderfällen ist das bisherige Verhalten nicht geändert worden.
800088	20. Mai 2009	8.0-CURRENT nach Änderungen am net80211-Monitormodus.
800089	23. Mai 2009	8.0-CURRENT nach dem Hinzufügen der Unterstützung von UDP-Kontrollblocks.
800090	23. Mai 2009	8.0-CURRENT nach der Virtualisierung der Schnittstellenklonierung.
800091	27. Mai 2009	8.0-CURRENT nach dem Hinzufügen von hierarchischen Jails und dem Entfernen des globalen securelevel.

Wert	Datum	Release
800092	29. Mai 2009	8.0-CURRENT nach der Änderung des `sx_init_flags()` -KPI. SX_ADAPTIVESPIN wurde zurückgezogen und eine neue Option SX_NOADAPTIVE wurde eingeführt, um die umgekehrte Logik zu behandeln.
800093	29. Mai 2009	8.0-CURRENT nach dem Hinzufügen von mnt_xflag zu struct mount.
800094	30. Mai 2009	8.0-CURRENT nach dem Hinzufügen von VOP_ACCESSX(9).
800095	30. Mai 2009	8.0-CURRENT nach der Änderung des Polling-KPI. Die Polling-Handler liefern nun die Zahl der verarbeiteten Pakete zurück. Die neue Option IFCAP_POLLING_NOCOUNT wurde weiter eingeführt, um anzugeben, dass der Rückgabewert nicht von Bedeutung ist und das Zählen der Pakete ausgelassen werden soll.
800096	1. Juni 2009	8.0-CURRENT nach der Aktualisierung der netisr-Implementierung und nachdem die Weise, wie FIBs gespeichert werden und wie auf sie zugegriffen wird, geändert wurde.
800097	8. Juni 2009	8.0-CURRENT nach Einführung der Destruktor-Infrastruktur für vnet einschließlich Hooks.
800097	11. Juni 2009	8.0-CURRENT nach Einführung eines Erkennungssystems für ausgehende Pakete, die direkt wieder in netgraph gelangen und deswegen eingereiht werden. Dabei wurde auch die Definition von struct thread geändert.

Wert	Datum	Release
800098	14. Juni 2009	8.0-CURRENT nach dem Einfließen von OpenSSL 0.9.8k.
800099	22. Juni 2009	8.0-CURRENT nach der Aktualisierung von NGROUPS und dem Verschieben der Routing-Virtualisierung in ein eigenes VImage-Modul.
800100	24. Juni 2009	8.0-CURRENT nach Änderung der SYSVIPC-ABI.
800101	29. Juni 2009	8.0-CURRENT nach dem Entfernen der zeichenorientierten Geräte aus /dev/net, von denen für jede Schnittstelle eines existiert.
800102	12. Juli 2009	8.0-CURRENT, nachdem struct sackhint, struct tcpcb und struct tcpstat mit Padding-Bytes aufgefüllt wurden.
800103	13. Juli 2009	8.0-CURRENT, nachdem struct tcpopt durch struct toeopt in der Schnittstelle zwischen dem TOE-Treiber und dem TCP-SYN-Cache ersetzt wurde.
800104	19. Juli 2009	8.0-CURRENT nach dem Hinzufügen einer vnet-spezifischen Speicherzuweisung, die auf dem Linker-Set-Verfahren basiert.
800105	19. Juli 2009	8.0-CURRENT nach der Inkrementierung der Versionsnummer aller Shared-Libraries, die Symbol-Versioning nicht aktiviert haben.
800106	24. Juli 2009	8.0-CURRENT nach Einführung des VM-Objekttyps OBJT_SG.
800107	2. August 2009	8.0-CURRENT nach Befreiung des Newbus-Subsystems von Giant durch Hinzufügen von sxlock und 8.0-RELEASE.

Wert	Datum	Release
800108	21. November 2009	8.0-CURRENT nach Implementierung des kevent-Filters EVFILT_USER.
800500	7. Januar 2010	8.0-STABLE nach Erhöhung von `__FreeBSD_version`, damit `pkg_add -r` packages-8-stable verwendet.
800501	24. Januar 2010	8.0-STABLE, nachdem die Prototypen von `scandir(3)` und `alphasort(3)` geändert wurden, um der SUSv4 zu entsprechen.
800502	31. Januar 2010	8.0-STABLE nach Hinzufügen von `sigpause(3)`.
800503	25. Februar 2010	8.0-STABLE nach dem Hinzufügen der ioctls SIOCGIFDESCR und SIOCSIFDESCR für Netzwerk-Schnittstellen. Diese ioctls können, nach dem Vorbild von OpenBSD, dazu verwendet werden, Schnittstellenbeschreibungen zu bearbeiten und auszulesen.
800504	1. März 2010	8.0-STABLE, nachdem x86emu, ein Software-Emulator von OpenBSD für x86-Prozessoren im Real-Mode, von CURRENT übernommen wurde.
800505	18. Mai 2010	8.0-STABLE nach dem Einfließen von liblzma, xz, xzdec und lzmainfo.
801000	14. Juni 2010	8.1-RELEASE
801500	14. Juni 2010	8.1-STABLE nach 8.1-RELEASE.
801501	November 3, 2010	8.1-STABLE nach der KBI-Änderung in struct sysentve und der Implementierung von PL_FLAG_SCE/SCX/EXEC/SI und pl_siginfo für ptrace(PT_LWPINFO) .
802000	22. Dezember 2010	8.2-RELEASE
802500	22. Dezember 2010	8.2-STABLE, nachdem 8.2-RELEASE erzeugt wurde.

Wert	Datum	Release
802501	28. Februar 2011	8.2-STABLE, nachdem DTrace aktualisiert wurde (so wird nun auch Userland-Tracing unterstützt).
802502	6. März 2011	8.2-STABLE, nachdem log2 und log2f in libm aufgenommen wurden.
802503	1. Mai 2011	8.2-STABLE, nachdem gcc auf die letzte unter der GPLv2 stehenden Version (aus dem FSF gcc-4_2-Zweig) aktualisiert wurde.
802504	28. Mai 2011	8.2-STABLE, nachdem KPI sowie die Infrastruktur zur Unterstützung von "modular congestion control" implementiert wurden.
802505	28. Mai 2011	8.2-STABLE, nachdem die KPIs Hhook und Khelp implementiert wurden.
802506	M28. Mai 2011	8.2-STABLE, nachdem OSD in die Struktur tcpcb eingebaut wurde.
802507	6. Juni 2011	8.2-STABLE nach dem Import von ZFS v28.
802508	8. Juni 2011	8.2-STABLE, nach dem Entfernen der Methode sv_schedtail struct sysvec.
802509	14. Juli 2011	8.2-STABLE, nachdem die binutils um die SSSE3-Unterstützung erweitert wurden.
802510	19. Juli 2011	8.2-STABLE, nach dem Hinzufügen des Flags RFTSIGZMB zu `rfork(2)`.
900000	22. August 2009	9.0-CURRENT.
900001	8. September 2009	9.0-CURRENT nach dem Import von x86emu, einem Software-Emulator von OpenBSD für x86-Prozessoren im Real-Mode.

Wert	Datum	Release
900002	23. September 2009	9.0-CURRENT nach Implementierung des kevent-Filters EVFILT_USER.
900003	2. Dezember 2009	9.0-CURRENT nach Hinzufügen von `sigpause(3)` und der PIE-Unterstützung zu csu.
900004	6. Dezember 2009	9.0-CURRENT nach Hinzufügen von libulog und dessen libutempter-Kompatibilitätsschnittstelle.
900005	12. Dezember 2009	9.0-CURRENT nach Hinzufügen von `sleepq_sleepcnt()`, das dazu verwendet werden kann, die Anzahl der in einer bestimmten Warteschlange eingereihten Threads abzufragen.
900006	4. Januar 2010	9.0-CURRENT, nachdem die Prototypen von `scandir(3)` und `alphasort(3)` geändert wurden, um der SUSv4 zu entsprechen.
900007	13. Januar 2010	9.0-CURRENT nach dem Entfernen von utmp(5) und dem Hinzufügen von utmpx (siehe `getutxent(3)`) zur besseren Erfassung von Benutzeranmeldungen und Systemereignissen.
900008	20. Januar 2010	9.0-CURRENT nach der Einführung von BSDL bc/dc zur Ersetzung von GNU bc/dc.
900009	26. Januar 2010	9.0-CURRENT nach dem Hinzufügen der ioctls SIOCGIFDESCR und SIOCSIFDESCR für Netzwerk-Schnittstellen. Diese ioctls können, nach dem Vorbild von OpenBSD, dazu verwendet werden, Schnittstellenbeschreibungen zu bearbeiten und auszulesen.
900010	22. März 2010	9.0-CURRENT nach dem Import von zlib 1.2.4.

Wert	Datum	Release
900011	24. April 2010	9.0-CURRENT nach Hinzufügen von Soft Updates Journaling.
900012	10. Mai 2010	9.0-CURRENT nach Hinzufügen von liblzma, xz, xzdec und lzmainfo.
900013	24. Mai 2010	9.0-CURRENT nach Einbringen von USB-Fehlerbehebungen in linux(4).
900014	10. Juni 2010	9.0-CURRENT nach Hinzufügen von Clang.
900015	22. Juli 2010	9.0-CURRENT nach dem Import von BSD grep.
900016	28. Juli 2010	9.0-CURRENT, nachdem mti_zone zu struct malloc_type_internal hinzugefügt wurde.
900017	23. August 2010	9.0-CURRENT nach dem Zurückkehren zu GNU grep als Standard und Hinzufügen der Option WITH_BSD_GREP.
900018	24. August 2010	9.0-CURRENT, nachdem das von `pthread_kill(3)` generierte Signal in si_code als SI_LWP bezeichnet wird. Zuvor war si_code SI_USER.
900019	28. August 2010	9.0-CURRENT nach Hinzufügen des Schalters MAP_PREFAULT_READ zu `mmap(2)`.
900020	9. September 2010	9.0-CURRENT, nachdem "drain"-Funktionalität in sbufs integriert wurde (wodurch sich auch das Layout von struct sbuf geändert hat).
900021	13. September 2010	9.0-CURRENT, nachdem "Userland tracing" in DTrace eingeführt wurde.
900022	2. Oktober 2010	9.0-CURRENT nach Hinzufügen der BSDL man-Utilities (und gleichzeitigem Entfernen der GNU/GPL man-Utilities).

Wert	Datum	Release
900023	11. Oktober 2010	9.0-CURRENT nach der Aktualisierung von xz auf den git-Snapshot 20101010.
900024	11. November 2010	9.0-CURRENT, nachdem libgcc.a durch libcompiler_rt.a.
900025	12. November 2010	9.0-CURRENT nach der Einführung von "modularised congestion control".
900026	30. November 2010	9.0-CURRENT nach der Einführung von "Serial Management Protocol (SMP) passthrough" sowie den XPT_SMP_IO und XPT_GDEV_ADVINFO CAM CCBs.
900027	5. Dezember 2010	9.0-CURRENT, nachdem log2 zu libm hinzugefügt wurde.
900028	21. Dezember 2010	9.0-CURRENT, nach dem HInzufügen von Hhook (Helper Hook), Khelp (Kernel Helpers) und Object Specific Data (OSD) KPIs.
900029	28. Dezember 2010	9.0-CURRENT, nach der TCP-Stack modifiziert wurde, um es den Khelp-Modulen zu erlauben, mit ihm über Helper Hook Points zu kommunizieren und Verbindungsdaten im TCP-Kontrollblock zu speichern.
900030	12. Januar 2011	9.0-CURRENT, nachdem libdialog auf die Version 20100428 aktualisiert wurde.
900031	7. Februar 2011	9.0-CURRENT, nach dem Hinzufügen von `pthread_getthreadid_np(3)`.
900032	8. Februar 2011	9.0-CURRENT, nachdem Prototyp und Symbol für uio_yield entfernt wurden.
900033	18. Februar 2011	9.0-CURRENT, nachdem die binutils auf Version 2.17.50 aktualisiert wurden.

Wert	Datum	Release
900034	8. März 2011	9.0-CURRENT, nachdem die Struktur sysvec (sv_schedtail) modifiziert wurde.
900035	29. März 20111	9.0-CURRENT, nach dem Update des im Basissystem enthaltenen gcc sowie von libstdc++ auf die letzten unter GPLv2 lizenzierten Versionen.
900036	18. April 2011	9.0-CURRENT, nachdem libobjc und die Unterstützung für Objective-C aus dem Basissystem entfernt wurden.
900037	13. Mai 2011	9.0-CURRENT, nach dem Import der libprocstat(3)-Bibliothek sowie von fuser(1) in das Basissystem.
900038	22. Mai 2011	9.0-CURRENT, nachdem ein Lock-Flag zu VFS_FHTOVP(9) hinzugefügt wurde.
900039	28. Juni 2011	9.0-CURRENT, nachdem pf von OpenBSD 4.5 importiert wurde.
900040	19. Juli 2011	Standardmäßige Erhöhung von MAXCPU für FreeBSD auf 64 für amd64 und ia64 und auf 128 für XLP (mips).
900041	13. August 2011	9.0-CURRENT, nachdem Capsicum-Funktionalitäten implementiert wurden. Zusätzlich wurde fget(9) um ein Rechte-Argument erweitert.
900042	28. August 2011	Versionssprünge für Shared-Libraries deren ABI sich geändert hat, in Vorbereitung für 9.0.
900043	2. September 2011	Automatische Erkennung von USB-Massenspeicher Geräten, die das no synchronize cache SCSI Kommando nicht unterstützen.
900044	10. September 2011	Re-factor auto-quirk.

Wert	Datum	Release
900045	13. Oktober 2011	Allen nicht-kompatiblen Systemaufruf-Einstiegspunkten wurde ein sys_ vorangestellt.



Beachten Sie, dass 2.2-STABLE sich nach dem 2.2.5-RELEASE manchmal als "2.2.5-STABLE" identifiziert. Das Muster war früher das Jahr gefolgt von dem Monat, aber wir haben uns entschieden, ab 2.2. einen geradlinigeren Ansatz mit major/minor-Nummern zu benutzen. Dies liegt daran, dass gleichzeitiges Entwickeln an mehreren Zweigen es unmöglich macht, die Versionen nur mit Hilfe des Datums des Releases zu unterteilen. Wenn Sie jetzt einen Port erstellen brauchen Sie sich nicht um alte -CURRENTs zu kümmern; diese sind hier nur als Referenz augeführt.

12.6. Etwas hinter die bsd.port.mk-Anweisung schreiben

Schreiben Sie bitte nichts hinter die `.include <bsd.port.mk>`-Zeile. Normalerweise kann dies vermieden werden, indem Sie die Datei bsd.port.pre.mk irgendwo in der Mitte Ihres Makefiles und bsd.port.post.mk am Ende einfügen.



Sie dürfen entweder nur das bsd.port.pre.mk/bsd.port.post.mk-Paar oder bsd.port.mk alleine hinzufügen; vermischen Sie diese Verwendungen nicht!

bsd.port.pre.mk definiert nur einige Variablen, welche in Tests im Makefile benutzt werden können, bsd.port.post.mk definiert den Rest.

Hier sind einige wichtige Variablen, welche in bsd.port.pre.mk definiert sind (dies ist keine vollständige Liste, lesen Sie bitte bsd.port.mk für eine vollständige Auflistung).

Variable	Beschreibung
`ARCH`	Die Architektur, wie von `uname -m` zurückgegeben (z.B. `i386`)
`OPSYS`	Der Typ des Betriebsystems, wie von `uname -s` zurückgegeben (z.B. `FreeBSD`)
`OSREL`	Die Release Version des Betriebssystems (z.B., `2.1.5` oder `2.2.7`)
`OSVERSION`	Die numerische Version des Betriebssystems; gleichbedeutend mit `__FreeBSD_version`.
`PORTOBJFORMAT`	Das Objektformat des Systems (`elf` oder `aout`; beachten Sie, dass für "moderne" Versionen von FreeBSD `aout` veraltet ist).
`LOCALBASE`	Die Basis des "local" Verzeichnisbaumes (z.B. `/usr/local/`)

Variable	Beschreibung
PREFIX	Wo der Port sich selbst installiert (siehe Mehr Informationen über PREFIX).



Falls Sie die Variablen USE_IMAKE, USE_X_PREFIX, oder MASTERDIR definieren müssen, sollten Sie dies vor dem Einfügen von bsd.port.pre.mk machen.

Hier sind ein paar Beispiele von Dingen, die Sie hinter die Anweisung bsd.port.pre.mk schreiben können:

```
# lang/perl5 muss nicht kompliliert werden, falls perl5 schon auf dem System ist
.if ${OSVERSION} > 300003
BROKEN= perl ist im System
.endif

# nur eine Versionsnummer für die ELF Version der shlib
.if ${PORTOBJFORMAT} == "elf"
TCL_LIB_FILE=  ${TCL_LIB}.${SHLIB_MAJOR}
.else
TCL_LIB_FILE=  ${TCL_LIB}.${SHLIB_MAJOR}.${SHLIB_MINOR}
.endif

# die Software erstellt schon eine Verknüpfung fü ELF, aber nicht fü a.out
post-install:
.if ${PORTOBJFORMAT} == "aout"
       ${LN} -sf liblinpack.so.1.0 ${PREFIX}/lib/liblinpack.so
.endif
```

Sie haben sich daran erinnert Tabulator statt Leerzeichen nach BROKEN= und TCL_LIB_FILE= zu benutzen, oder? :-).

12.7. Benutzen Sie die exec-Anweisung in Wrapper-Skripten

Falls der Port ein Shellskript installiert, dessen Zweck es ist ein anderes Programm zu starten, und falls das Starten des Programmes die letzte Aktion des Skripts ist, sollten Sie sicherstellen, dass Sie die Funktion exec dafür benutzen; zum Beispiel:

```
#!/bin/sh
exec %%LOCALBASE%%/bin/java -jar %%DATADIR%%/foo.jar "$@"
```

Die Funktion exec ersetzt den Shell-Prozess mit dem angegebenen Programm. Falls exec ausgelassen wird, verbleibt der Shell-Prozess im Speicher während das Programm ausgefährt wird und verbraucht unnötig Systemressourcen.

12.8. Aufgaben vernünftig lösen

Das Makefile sollte die nötigen Schritte einfach und vernünftig durchführen. Wenn Sie ein einige Zeilen einsparen oder die Lesbarkeit verbessern können, dann machen Sie dies bitte. Beispiele sind: Ein make-Konstrukt `.if` anstatt eines Shellkonstrukt `if` zu verwenden, anstatt `do-extract` neu zu definieren, dies mit `EXTRACT*` machen, oder `GNU_CONFIGURE` anstelle von `CONFIGURE_ARGS += --prefix=${PREFIX}` zu verwenden.

Falls Sie sich in einer Situation wiederfinden, in der Sie viel Code neu schreiben müssen, um etwas zu testen, sollten Sie zuerst bsd.port.mk erneut konsultieren und nachprüfen ob es nicht bereits eine Lösung für Ihr Problem enthält. Es ist zwar schwer zu lesen, beinhaltet jedoch eine Menge kurzer Lösungen für viele scheinbar schwierige Probleme.

12.9. Berücksichtigen Sie sowohl `CC` als auch `CXX`

Der Port sollte sowohl die `CC`- wie auch die `CXX`-Variable berücksichtigen. Damit ist gemeint, dass der Port diese Variablen nicht ohne Rücksicht auf eventuell schon gesetzte Werte einfach überschreiben sollte; stattdessen sollten neue Werte an schon existierende angehängt werden. Dadurch können Build-Optionen, die alle Ports betreffen, global definiert werden.

Falls der Port diese Variablen nicht berücksichtigt, sollte `NO_PACKAGE=ignores either cc or cxx` ins Makefile eingefügt werden.

Im Folgenden wird ein Beispiel eines Makefiles gezeigt, welches die beiden Variablen `CC` und `CXX` berücksichtigt. Beachten Sie das `?=`:

```
CC?= gcc
```

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

Nachfolgend ein Beispiel, welches weder `CC` noch `CXX` berücksichtigt:

```
CC= gcc
```

```
CXX= g++
```

Die Variablen `CC` und `CXX` können auf FreeBSD-Systemen in /etc/make.conf definiert werden. Im ersten Beispiel wird ein Wert nur dann gesetzt, falls dieser vorher noch nicht gesetzt war, um so systemweite Definitionen zu berücksichtigen. Im zweiten Beispiel werden die Variablen ohne Rücksicht überschrieben.

12.10. Berücksichtigen Sie CFLAGS

Der Port sollte die Variable CFLAGS berücksichtigen. Damit ist gemeint, dass der Port den Wert dieser Variablen nicht absolut setzen und damit existierende Werte überschreiben sollte; stattdessen sollte er weitere Werte der Variablen durch Anhängen hinzufügen. Dadurch können Build-Optionen, die alle Ports betreffen, global definiert werden.

Falls der Port diese Variablen nicht berücksichtigt, sollte `NO_PACKAGE=ignores cflags` ins Makefile eingefügt werden.

Im Folgenden wird ein Beispiel eines Makefiles gezeigt, welches die Variable CFLAGS berücksichtigt. Beachten Sie das `+=`:

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

Nachfolgend finden Sie ein Beispiel, welches die CFLAGS-Variable nicht berücksichtigt:

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

Die Variable CFLAGS wird auf FreeBSD-Systemen in /etc/make.conf definiert. Im ersten Beispiel werden weitere Flags an die Variable CFLAGS angehängt und somit der bestehende Wert nicht gelöscht. Im zweiten Beispiel wird die Variable ohne Rücksicht überschrieben.

Sie sollten Optimierungsflags aus Makefiles Dritter entfernen. Die CFLAGS des Systems beinhalten systemweite Optimierungsflags. Ein Beispiel eines unveränderten Makefiles:

```
CFLAGS= -O3 -funroll-loops -DHAVE_SOUND
```

Werden nun systemweite Optimierungsflags verwendet so würde das Makefile in etwa folgendermaßen aussehen:

```
CFLAGS+= -DHAVE_SOUND
```

12.11. Threading-Bibliotheken

Die Threading-Bibliothek muss mit Hilfe eines speziellen Linker-Flags `-pthread` in die Binärdateien unter FreeBSD gebunden werden. Falls ein Port auf ein direktes Verlinken gegen `-lpthread` oder `-lc_r` besteht, passen Sie den Port bitte so an, dass er die durch das Port-Framework bereitgestellte Variable PTHREAD_LIBS verwendet. Diese Variable hat üblicherweise den Wert `-pthread`, kann aber auf einigen Architekturen und FreeBSD-Versionen abweichende Werte haben und daher sollte nie `-pthread` direkt in Patches geschrieben werden, sondern immer PTHREAD_LIBS.



Falls durch das Setzen von PTHREAD_LIBS der Bau des Ports mit der Fehlermeldung `unrecognized option '-pthread'` abbricht, kann die Verwendung des `gcc` als Linker

> durch setzen von CONFIGURE_ENV auf LD=$Cheng Cui <cc@FreeBSD.org> helfen. Die Option -pthread wird nicht direkt von ld unterstützt.

12.12. Rückmeldungen

Brauchbare Änderungen/Patches sollten an den ursprünglichen Autor/Maintainer der Software geschickt werden, damit diese in der nächsten Version der Software mit aufgenommen werden können. Dadurch wird Ihre Aufgabe für die nächste Version der Software deutlich einfacher.

12.13. README.html

Nehmen Sie bitte keine README.html in den Port auf. Diese Datei ist kein Bestandteil der CVS-Sammlung sondern wird durch make readme erzeugt.

12.14. Einen Port durch BROKEN, FORBIDDEN oder IGNORE als nicht installierbar markieren

In manchen Fällen sollten Benutzer davon abgehalten werden einen Port zu installieren. Um einem Benutzer mitzuteilen, dass ein Port nicht installiert werden sollte, gibt es mehrere Variablen für make, die im Makefile des Ports genutzt werden können. Der Wert der folgenden make-Variablen wird dem Benutzer als Grund für die Ablehnung der Installation des Ports zurückgegeben. Bitte benutzen Sie die richtige make-Variable, denn jede enthält eine völlig andere Bedeutung für den Benutzer und das automatische System, das von dem Makefile abhängt, wie der Ports-Build-Custer, FreshPorts und portsmon.

12.14.1. Variablen

- BROKEN ist reserviert für Ports, welche momentan nicht korrekt kompiliert, installiert oder deinstalliert werden. Es sollte für Ports benutzt werden, von denen man annimmt, dass dies ein temporäres Problem ist.

  Falls angegeben, wird der Build-Cluster dennoch versuchen den Port zu bauen, um zu sehen, ob das zugrunde liegende Problem behoben wurde (das ist jedoch im Allgemeinen nicht der Fall).

  Benutzen Sie BROKEN zum Beispiel, wenn ein Port:

  - nicht kompiliert
  - beim Konfiguration- oder Installation-Prozess scheitert
  - Dateien außerhalb von ${LOCALBASE} installiert
  - beim Deinstallieren nicht alle seine Dateien sauber entfernt (jedoch kann es akzeptable und wünschenswert sein, Dateien, die vom Nutzer verändert wurden, nicht zu entfernen)

- FORBIDDEN wird für Ports verwendet, die Sicherheitslücken enthalten oder die ernste Sicherheitsbedenken für das FreeBSD-System aufwerfen, wenn sie installiert sind (z.B. ein als unsicher bekanntes Programm, oder ein Programm, das einen Dienst zur Verfügung stellt, der leicht kompromittiert werden kann). Ports sollten als FORBIDDEN gekennzeichnet werden, sobald

ein Programm eine Schwachstelle hat und kein Update veröffentlicht wurde. Idealerweise sollten Ports so bald wie möglich aktualisiert werden wenn eine Sicherheitslücke entdeckt wurde, um die Zahl verwundbarer FreeBSD-Hosts zu verringern (wir schätzen es für unsere Sicherheit bekannt zu sein), obwohl es manchmal einen beachtlichen Zeitabstand zwischen der Bekanntmachung einer Schwachstelle und dem entsprechenden Update gibt. Bitte kennzeichnen Sie einen Port nicht aus irgendeinem Grund außer Sicherheit als FORBIDDEN.

- IGNORE ist für Ports reserviert, die aus anderen Gründen nicht gebaut werden sollten. Es sollte für Ports verwendet werden, in denen ein strukturelles Problem vermutet wird. Der Build-Cluster wird unter keinen Umständen Ports, die mit IGNORE markiert sind, erstellen. Verwenden Sie IGNORE zum Beispiel, wenn ein Port:
  - kompiliert, aber nicht richtig läuft
  - nicht auf der installierten Version von FreeBSD läuft
  - FreeBSD Kernelquelltext zum Bauen benötigt, aber der Benutzer diese nicht installiert hat
  - ein Distfile benötigt, welches aufgrund von Lizenzbeschränkungen nicht automatisch abgerufen werden kann
  - nicht korrekt mit einem momentan installiertem Port arbeitet (der Port hängt zum Beispiel von www/apache21 ab, aber www/apache13 ist installiert)



Wenn ein Port mit einem momentan installiertem Port kollidiert (zum Beispiel, wenn beide eine Datei an die selbe Stelle installieren, diese aber eine andere Funktion hat), benutzen Sie stattdessen CONFLICTS. CONFLICTS setzt IGNORE dann selbstständig.

- Um einen Port nur auf bestimmte Systemarchitekturen mit IGNORE zu markieren, gibt es zwei Variablen, die automatisch IGNORE für Sie setzen: ONLY_FOR_ARCHS und NOT_FOR_ARCHS. Beispiele:

```
ONLY_FOR_ARCHS= i386 amd64
```

```
NOT_FOR_ARCHS= alpha ia64 sparc64
```

  Eine eigene IGNORE-Ausgabe kann mit ONLY_FOR_ARCHS_REASON und NOT_FOR_ARCHS_REASON festgelegt werden. Für eine bestimmte Architektur sind Angaben durch ONLY_FOR_ARCHS_REASONARCH_ und NOT_FOR_ARCHS_REASONARCH_ möglich.

- Wenn ein Port i386-Binärdateien herunterlädt und installiert, sollte IA32_BINARY_PORT gesetzt werden. Wenn die Variable gesetzt ist, wird überprüft, ob das Verzeichnis /usr/lib32 für IA32-Versionen der Bibliotheken vorhanden ist, und ob der Kernel mit IA32-Kompatibilität gebaut wurde. Wenn eine dieser zwei Voraussetzungen nicht erfüllt ist, wird IGNORE automatisch gesetzt.

12.14.2. Anmerkungen zur Implementierung

Zeichenketten sollten nicht in Anführungszeichen gesetzt werden. Auch die Wortwahl der Zeichenketten sollte die Art und Weise beachten, wie die Informationen dem Nutzer angezeigt

werden. Beispiele:

```
BROKEN= this port is unsupported on FreeBSD 5.x
```

```
IGNORE= is unsupported on FreeBSD 5.x
```

resultieren in den folgenden Ausgaben von `make describe`:

```
===>  foobar-0.1 is marked as broken: this port is unsupported on FreeBSD 5.x.
```

```
===>  foobar-0.1 is unsupported on FreeBSD 5.x.
```

12.15. Kennzeichnen eines Ports zur Entfernung durch `DEPRECATED` oder `EXPIRATION_DATE`

Denken Sie bitte daran, dass `BROKEN` und `FORBIDDEN` nur als temporärer Ausweg verwendet werden sollten, wenn ein Port nicht funktioniert. Dauerhaft defekte Ports sollten komplett aus der Ports-Sammlung entfernt werden.

Wenn es sinnvoll ist, können Benutzer vor der anstehenden Entfernung eines Ports mit `DEPRECATED` und `EXPIRATION_DATE` gewarnt werden. Ersteres ist einfach eine Zeichenkette, die angibt, warum der Port entfernt werden soll. Letzteres ist eine Zeichenkette im ISO 8601-Format (JJJJ-MM-TT). Beides wird dem Benutzer gezeigt.

Es ist möglich `DEPRECATED` ohne `EXPIRATION_DATE` zu setzen (zum Beispiel, um eine neuere Version des Ports zu empfehlen), aber das Gegenteil ist sinnlos.

Es gibt keine Vorschrift wie lange die Vorwarnzeit sein muss. Gegenwärtig ist es üblich einen Monat für sicherheitsrelevante Probleme und zwei Monate für Build-Probleme anzusetzen. Dies gibt allen interessierten Committern ein wenig Zeit die Probleme zu beheben.

12.16. Vermeiden Sie den Gebrauch des `.error` -Konstruktes

Der korrekte Weg eines Makefile anzuzeigen, dass der Port aufgrund eines externen Grundes nicht installiert werden kann (zum Beispiel, weil der Benutzer eine ungültige Kombination von Build-Optionen angegeben hat), ist `IGNORE` auf einen nicht leeren Wert zu setzen. Dieser wird dann formatiert und dem Benutzer von `make install` ausgegeben.

Es ist ein verbreiteter Fehler `.error` für diesem Zweck zu verwenden. Das Problem dabei ist, dass viele automatisierte Werkzeuge, die mit dem Ports-Baum arbeiten, in dieser Situation fehlschlagen. Am Häufigsten tritt das Problem beim Versuch /usr/ports/INDEX zu bauen auf (siehe `make describe` ausführen). Jedoch schlagen auch trivialere Befehle wie `make maintainer` in diesem Fall fehl. Dies ist

nicht akzeptabel!

Beispiel 25. Wie vermeidet man die Verwendung von `.error`

Nehmen Sie an, dass die Zeile

```
USE_POINTYHAT=yes
```

in make.conf enthalten ist. Der erste der folgenden zwei Makefile-Schnipsel lässt `make index` fehlschlagen, während der zweite dies nicht tut.

```
.if USE_POINTYHAT
.error "POINTYHAT is not supported"
.endif
```

```
.if USE_POINTYHAT
IGNORE=POINTYHAT is not supported
.endif
```

12.17. Verwendung von sysctl

Vom Gebrauch von sysctl wird, außer in Targets, abgeraten. Das liegt daran, dass die Auswertung aller `makevar`s, wie sie während `make index` verwendet werden, dann den Befehl ausführen muss, welches den Prozess weiter verlangsamt.

Die Verwendung von sysctl(8) sollte immer durch die Variable `SYSCTL` erfolgen, da diese den vollständigen Pfad enthält und überschrieben werden kann, so dies als notwendig erachtet wird.

12.18. Erneutes Ausliefern von Distfiles

Manchmal ändern die Autoren der Software den Inhalt veröffentlichter Distfiles, ohne den Dateinamen zu ändern. Sie müssen überprüfen, ob die Änderungen offizell sind und vom Autor durchgeführt wurden. Es ist in der Vergangenheit vorgekommen, dass Distfiles still und heimlich auf dem Download-Server geändert wurden, um Schaden zu verursachen oder die Sicherheit der Nutzer zu kompromittieren.

Verschieben Sie das alte Distfile und laden Sie das neue herunter. Entpacken Sie es und vergleichen Sie den Inhalt mittels diff(1). Wenn Sie nichts Verdächtiges sehen können Sie distinfo aktualisieren. Stellen Sie sicher, dass die Änderungen in Ihrem PR oder Commit-Protokoll zusammengefasst sind, um zu Gewährleisten, dass nichts Negatives passiert ist.

Sie können auch mit den Autoren der Software in Verbindung treten und sich die Änderungen bestätigen lassen.

12.19. Verschiedenes

Die Dateien pkg-descr und pkg-plist sollten beide doppelt kontrolliert werden. Wenn Sie einen Port nachprüfen und glauben, dass man es besser machen kann, dann verbessern Sie ihn bitte.

Bitte kopieren Sie nicht noch mehr Exemplare der GNU General Public License in unser System.

Bitte überprüfen Sie alle gesetzlichen Punkte gründlich! Lassen Sie uns bitte keine illegale Software verbreiten!

Kapitel 13. Beispiel eines Makefile

Hier ein Beispiel für ein Makefile, welches als Vorlage für einen neuen Port dienen kann. Alle zusätzlichen Kommentare in eckigen Klammern müssen entfernt werden!

Es wird empfohlen, die hier gezeigte Formatierung zu übernehmen (Reihenfolge der Variablen, Leerzeichen zwischen einzelnen Abschnitten, usw.). Dadurch werden die wichtigen Informationen sofort ersichtlich. Zur Überprüfung Ihres Makefiles sollten Sie portlint verwenden.

```
[the header...just to make it easier for us to identify the ports.]
# New ports collection makefile for:   xdvi
[the "version required" line is only needed when the PORTVERSION
 variable is not specific enough to describe the port.]
# Date created:               26 May 1995
[this is the person who did the original port to FreeBSD, in particular, the
person who wrote the first version of this Makefile.  Remember, this should
not be changed when upgrading the port later.]
# Whom:                       Satoshi Asami <asami@FreeBSD.org>
#
# $FreeBSD$
[ ^^^^^^^^^ This will be automatically replaced with RCS ID string by CVS
when it is committed to our repository.  If upgrading a port, do not alter
this line back to "$FreeBSD$".  CVS deals with it automatically.]
#

[section to describe the port itself and the master site - PORTNAME
 and PORTVERSION are always first, followed by CATEGORIES,
 and then MASTER_SITES, which can be followed by MASTER_SITE_SUBDIR.
 PKGNAMEPREFIX and PKGNAMESUFFIX, if needed, will be after that.
 Then comes DISTNAME, EXTRACT_SUFX and/or DISTFILES, and then
 EXTRACT_ONLY, as necessary.]
PORTNAME=      xdvi
PORTVERSION=   18.2
CATEGORIES=    print
[do not forget the trailing slash ("/")!
 if you are not using MASTER_SITE_* macros]
MASTER_SITES=  ${MASTER_SITE_XCONTRIB}
MASTER_SITE_SUBDIR= applications
PKGNAMEPREFIX= ja-
DISTNAME=      xdvi-pl18
[set this if the source is not in the standard ".tar.gz" form]
EXTRACT_SUFX=  .tar.Z

[section for distributed patches -- can be empty]
PATCH_SITES=   ftp://ftp.sra.co.jp/pub/X11/japanese/
PATCHFILES=    xdvi-18.patch1.gz xdvi-18.patch2.gz

[maintainer; *mandatory*!  This is the person who is volunteering to
 handle port updates, build breakages, and to whom a users can direct
 questions and bug reports.  To keep the quality of the Ports Collection
```

```
 as high as possible, we no longer accept new ports that are assigned to
 "ports@FreeBSD.org".]
MAINTAINER=    asami@FreeBSD.org
COMMENT=       A DVI Previewer for the X Window System

[dependencies -- can be empty]
RUN_DEPENDS=   gs:${PORTSDIR}/print/ghostscript
LIB_DEPENDS=   Xpm.5:${PORTSDIR}/graphics/xpm

[this section is for other standard bsd.port.mk variables that do not
 belong to any of the above]
[If it asks questions during configure, build, install...]
IS_INTERACTIVE=        yes
[If it extracts to a directory other than ${DISTNAME}...]
WRKSRC=                ${WRKDIR}/xdvi-new
[If the distributed patches were not made relative to ${WRKSRC}, you
 may need to tweak this]
PATCH_DIST_STRIP=      -p1
[If it requires a "configure" script generated by GNU autoconf to be run]
GNU_CONFIGURE= yes
[If it requires GNU make, not /usr/bin/make, to build...]
USE_GMAKE=     yes
[If it is an X application and requires "xmkmf -a" to be run...]
USE_IMAKE=     yes
[et cetera.]

[non-standard variables to be used in the rules below]
MY_FAVORITE_RESPONSE=  "yeah, right"

[then the special rules, in the order they are called]
pre-fetch:
    i go fetch something, yeah

post-patch:
    i need to do something after patch, great

pre-install:
    and then some more stuff before installing, wow

[and then the epilogue]
.include <bsd.port.mk>
```

Kapitel 14. Auf dem Laufenden bleiben

Die FreeBSD Ports-Sammlung verändert sich ständig. Hier finden Sie einige Informationen, wie Sie auf dem Laufenden bleiben.

14.1. FreshPorts

Einer der einfachsten Wege, um sich über Aktualisierungen, die bereits durchgeführt wurden, zu informieren, ist sich bei FreshPorts anzumelden. Sie können dort beliebige Ports auswählen, die Sie beobachten möchten. Maintainern wird ausdrücklich empfohlen sich anzumelden, da Sie nicht nur über Ihre eigenen Änderungen informiert werden, sondern auch über die aller anderen Committer (Diese sind oft nötig, um über Änderungen des zugrunde liegenden Frameworks informiert zu bleiben. Obwohl es höflich wäre, vorher über solche Änderungen benachrichtigt zu werden, wird es manchmal vergessen oder ist einfach nicht möglich. Außerdem sind die Änderungen manchmal nur sehr klein. Wir erwarten von jedem in solchen Fällen nach bestem Gewissen zu urteilen).

Wenn Sie Fresh-Ports benutzen möchten, benötigen Sie nur einen Account. Falls Sie sich mit einer `@FreeBSD.org` E-Mailadresse registriert haben, werden Sie den Anmeldelink am rechten Rand der Seite finden. Diejenigen, die bereits einen FeshPorts-Account haben, aber nicht Ihre `@FreeBSD.org` E-Mailadresse benutzen, können einfach Ihre E-Mailadresse auf `@FreeBSD.org` ändern, sich anmelden, und dann die Änderung rückgängig machen.

FreshPorts bietet auch eine Überprüfungsfunktion, die automatisch alle Committs zum FreeBSD Ports-Baum testet. Wenn Sie sich für diesen Dienst anmelden, werden Sie über alle Fehler, die bei der Überprüfung Ihres Committs auftreten, informiert.

14.2. Die Webschnittstelle zum Quelltext-Repository

Es ist möglich die Dateien des Quellen-Repositories mit Hilfe einer Webschnittstelle durchzusehen. Änderungen, die das gesamte Ports-System betreffen, werden jetzt in der Datei CHANGES dokumentiert. Solche, die nur bestimmte Ports betreffen, in der Datei UPDATING. Aber die maßgebliche Antwort auf alle Fragen liegt zweifellos darin, den Quelltext von bsd.port.mk und dazugehörige Dateien zu lesen.

14.3. Die FreeBSD Ports-Mailingliste

Wenn Sie Maintainer sind, sollten Sie in Erwägung ziehen die FreeBSD ports-Mailingliste zu verfolgen. Wichtige Änderungen an der grundlegenden Funktionsweise von Ports werden dort angekündigt und dann in CHANGES committet.

14.4. Der Cluster zum Bauen von FreeBSD-Ports auf `pointyhat.FreeBSD.org`

Eine der weniger bekannten Stärken von FreeBSD ist es, dass ein ganzer Cluster von Maschinen nur dafür reserviert ist, andauernd die Ports-Sammlung zu bauen, und zwar für jedes große FreeBSD Release und jede Tier-1-Architektur. Die Ergebnisse können Sie unter package building

logs and errors finden.

Alle Ports ausser denjenigen, die als `IGNORE` markiert sind, werden gebaut. Ports, die als `BROKEN` markiert sind, werden dennoch ausprobiert, um zu sehen, ob das zugrunde liegende Problem gelöst wurde (Dies wird erreicht, indem `TRYBROKEN` an das Makefile des Ports übergeben wird).

14.5. Der FreeBSD Ports-Distfile-Scanner

Der Build-Cluster ist dazu bestimmt, das neueste Release jedes Ports aus bereits heruntergeladenden Distfiles zu bauen. Da sich das Internet aber ständig verändert, können Distfiles schnell verloren gehen. Der FreeBSD Ports-Distfile-Scanner versucht jeden Download-Standort für jeden Port anzufragen, um herauszufinden, ob jedes Distfile noch verfügbar ist. Maintainer werden gebeten diesen Bericht regelmäßig durchzusehen, nicht nur, um den Build-Prozess für die Nutzer zu beschleunigen, sondern auch um zu vermeiden, dass auf den Maschinen, die freiwillig zur Verfügung gestellt werden, um all diese Dateien anzubieten, Ressourcen verschwendet werden.

14.6. Das FreeBSD Ports-Monitoring-System

Eine weitere praktische Ressource ist das FreeBSD Ports-Monitoring-System (auch bekannt als `portsmon`). Dieses System besteht aus einer Datenbank, die Informationen von mehreren Quellen bezieht und es erlaubt diese über ein Webinterface abzufragen. Momentan werden die Ports-Problemberichte (PRs), die Fehlerprotokolle des Build-Clusters und die einzelnen Dateien der Ports-Sammlung verwendet. In Zukunft soll das auf die Distfile-Prüfung und weitere Informationsquellen ausgedehnt werden.

Als Ausgangspunkt können Sie alle Informationen eines Ports mit Hilfe der Übersicht eines Ports betrachten.

Zum Zeitpunkt des Schreibens ist dies die einzige Quelle, die GNATS PR-Einträge auf Portnamen abbildet (PR-Einreicher geben den Portnamen nicht immer in der Zusammenfassung an, obwohl wir uns das wünschen würden). Also ist `portsmon` ein guter Anlaufpunkt, wenn Sie herausfinden wollen, ob zu einem existierenden Port PRs oder Buildfehler eingetragen sind. Oder um herauszufinden, ob ein neuer Port, den Sie erstellen wollen, bereits eingereicht wurde.