Sprache: de
[Teams]
en es fr id ja nl pl pt zh-cn zh-tw

Weitere Dokumente
Upgrade Guide
-current folgen
Ports und Packages
Anleitung zum Testen von Ports
AnonCVS anwenden
Stable
CVSup anwenden
Manualseiten
Fehler melden
Mailinglisten
PF-Benutzerhandbuch
OpenSSH-FAQ

PDF-Dateien
OpenBSD-FAQ
PF-Benutzerhandbuch

Textdateien
OpenBSD-FAQ
PF-Benutzerhandbuch

Zurück zu OpenBSD

Dokumentation und häufig gestellte Fragen (FAQs)

Häufige Probleme Neue Updates

Diese FAQ ist eine erweiterte Dokumentation zu den Manualseiten, die sowohl auf einem installierten System als auch online zu finden sind. Die FAQ deckt die jeweils aktuelle Version von OpenBSD ab, zur Zeit also Version 3.6. Es befinden sich sehr wahrscheinlich Funktionen und Änderungen an Funktionen in der Entwicklerversion (-current) von OpenBSD, die in dieser FAQ *nicht* behandelt werden.

Die FAQ als PDF- oder als reine Text-Version ist im Verzeichnis **pub/OpenBSD/doc** der FTP-,mirrors' zusammen mit anderen Dokumenten zu finden.

Anmerkung: Die FAQs sind teilweise **noch nicht in deutsch übersetzt oder veraltet** und verweisen in diesem Fall auf die englische Version. Betroffen sind davon zur Zeit:

- 8 - Allgemeine Fragen
- 10 - Systemverwaltung

Jeder dieser Links wurde verändert, damit du zu der aktuellen englischen Version geleitet wirst.

Wenn du bei unserer Übersetzungsarbeit mithelfen willst, sieh dir bitte die Übersetzungsseite an.

1 - Einführung in OpenBSD

7 - Tastatur- und Bildschirm-Kontrollen

8 - Allgemeine Fragen

9 - Zu OpenBSD migrieren

10 - Systemverwaltung

- 14.8 - Installieren von Bootblocks - i386 spezifisch
- 14.9 - Sich auf das Schlimmste vorbereiten: Backups und Wiederherstellen von Band.
- 14.10 - Diskimages in OpenBSD mounten
- 14.11 - Hilfe! Ich erhalte Fehler mit IDE DMA!
- 14.13 - RAID-Optionen in OpenBSD
- 14.14 - Warum sagt mir `df(1)`, dass ich mehr als 100% von meiner Platte belegt habe?

PF-Benutzerhandbuch

- Grundkonfiguration
  - Erste Schritte
  - Listen und Makros
  - Tabellen
  - Pakete filtern
  - Network Address Translation
  - Verkehr-Umleitung (Port-Weiterleitung)
  - Abkürzungen zum Erzeugen von Regelsätzen
- Fortgeschrittene Konfiguration
  - Laufzeit-Optionen
  - Scrub (Paket-Normalisierung)
  - Anker
  - Paket-Queueing und Priorisierung
  - Adress-Pools und Load Balancing
  - Pakete markieren
- Zusätzliche Themen
  - Aufzeichnen
  - Leistung
  - Probleme mit FTP
  - Authpf: Benutzer-Shell für authentifizierende Gateways
- Beispiel für Regelsätze
  - Beispiel: Firewall für zuhause oder ein kleines Büro

Häufige Probleme

- Häufige Installationsprobleme
- Wie führe ich ein Upgrade meines Systems aus?
- Packet Filter
- Soll ich nun Ports oder Packages benutzen?
- Wie konfiguriere ich ein Multi-Boot-System?
- Festplatten DMA Fehler
- Wireless-Netzwerk-Optionen

Neueste Aktualisierungen

- Neue und verbesserte Anweisungen zum Erzeugen vom Source aus
- Wireless Netzwerk Optionen - verschoben
- FAQ für OpenBSD 3.6 überarbeitet
- Der Upgrade Guide - neu
- FAQ 6, OpenNTPD - neu
- FAQ 6, CARP - neu
- FAQ 1, Wieso ist ProduktX (nicht) enthalten? - neu
- FAQ 14, Warum sagt mir `df(1)`, dass ich mehr als 100% von meiner Platte belegt habe? - aktualisiert

---

Die FAQ-Maintainer sind Nick Holland und Joel Knight. Weitere Mitarbeiter an der FAQ sind unter anderem Eric Jackson, Wim Vandeputte und Chris Cappuccio.

Informationen über die Übersetzung dieser FAQ und den Rest der OpenBSD-Webseite finden sich auf der Übersetzungsseite.

Fragen und Kommentare bezüglich der FAQ können an faq@openbsd.org gerichtet werden. Allgemeine Fragen über OpenBSD gehören auf die passende Mailingliste.

Zurück zu OpenBSD

$OpenBSD: index.html,v 1.100 2005/02/27 12:04:57 jufi Exp $
"If you don't find it in the index, look very carefully through the entire

catalogue."
Sears, Roebuck, and Co., Consumer's Guide, 1897

OpenBSD

[FAQ Index] [Zum Kapitel 2 - Andere Informationsquellen zu OpenBSD]

1 - Einführung in OpenBSD

Inhaltsverzeichnis

1.1 - Was ist OpenBSD?

Das OpenBSD Projekt produziert ein frei erhältliches, plattformübergreifendes, auf 4.4BSD-basierendes UNIX-ähnliches Betriebssystem. Unsere Ziele legen den Schwerpunkt auf Korrektheit, Sicherheit, Standardisierung und Portabilität. OpenBSD unterstützt Binäremulation der meisten Programme von SVR4 (Solaris), FreeBSD, Linux, BSD/OS, SunOS und HP-UX.

Diese FAQ beschränkt sich ausschließlich auf das aktuellste Release von OpenBSD, Version 3.6.

1.2 - Auf welchen Systemen läuft OpenBSD?

OpenBSD 3.6 läuft auf den folgenden Plattformen:

- alpha - nur FTP
- amd64 - von CD bootfähig

- cats - nur FTP
- hp300 - nur FTP
- hppa - nur FTP
- i386 - von CD bootfähig
- mac68k - nur FTP
- macppc - von CD bootfähig
- mvme68k - nur FTP
- mvme88k - nur FTP
- sparc - von CD bootfähig
- sparc64 - von CD bootfähig
- vax

bootfähig bedeutet, dass OpenBSD direkt von der CD starten wird. Die CDs werden auf einigen Plattformen starten. Details, wie man OpenBSD auf CD beziehen kann, gibt es im Kapitel 3 dieser FAQ.

Vorherige Versionen von OpenBSD hatten auch einen Port für:

- amiga - Entfernt nach der Version 3.2
- sun3 - Entfernt nach der Version 2.9
- arc - Entfernt nach der Version 2.3
- pmax - Entfernt nach der Version 2.7

1.3 - Ist OpenBSD wirklich frei?

OpenBSD ist vollständig frei. Die Binärdateien sind frei. Die Quelltexte sind frei. Alle Teile von OpenBSD unterliegen entsprechenden Urheberrechtsbestimmungen, die die freie Weiterverbreitung erlauben. Dies beinhaltet auch die Möglichkeit, die meisten Teile von den OpenBSD Quelltexten WIEDERZUVERWENDEN, sei es für private oder kommerzielle Zwecke. OpenBSD unterliegt KEINEN weiteren Beschränkungen als durch die originale BSD Lizenz. Software, die unter strikterer Lizenz verfasst wurde, kann nicht in der regulären OpenBSD Distribution eingebunden werden. Dies dient zur Sicherstellung der weiteren freien Nutzbarkeit von OpenBSD. Zum Beispiel kann OpenBSD frei für den privaten und akademischen Gebrauch, von Regierungsinstitutionen, von non-profit Organisationen und von Unternehmen eingesetzt werden. OpenBSD, oder Teile davon, können auch problemlos in kommerzielle Produkte integriert werden.

Für weitere Informationen über populäre Lizenzen siehe: OpenBSD Copyright Policy.

Die Leiter von OpenBSD unterstützen das Projekt hauptsächlich aus ihren eigenen Taschen. Dies beinhaltet die Zeit für Programmieren, die eingesetzte Hardware für die verschiedenen Plattformen, die Netzwerkressourcen, um OpenBSD zu dir zu bringen, und die Zeit, um Fragen zu beantworten und die Fehlerberichte der Benutzer zu untersuchen.

Die OpenBSD Entwickler sind nicht gerade reich und auch ein kleiner Beitrag an Zeit, Hardware oder Ressourcen macht einen großen Unterschied.

1.4 - Warum sollte ich OpenBSD benutzen?

Neue Benutzer wollen häufig wissen, ob OpenBSD anderen UNIX-ähnlichen Betriebssytemen überlegen ist. Diese Frage ist eigentlich unbeantwortbar und das Thema von zahllosen (und nutzlosen) Debatten. Stelle diese Frage bitte unter keinen Umständen auf einer OpenBSD Mailingliste.

Anbei stehen ein paar Gründe, warum wir glauben, dass OpenBSD ein nützliches Betriebssystem ist. Ob OpenBSD das richtige System für dich ist, kannst nur du entscheiden.

- OpenBSD läuft auf vielen verschiedenen Hardware Plattformen.
- OpenBSD wird von vielen Sicherheitsexperten als das sicherste UNIX-ähnliche Betriebssystem angesehen, was das Resultat eines immer weiter andauernden ausführlichen Quelltextsicherheitsaudits ist.
- OpenBSD ist ein voll ausgestattetes UNIX-ähnliches Betriebssystem, das im Quelltext ohne Kosten verfügbar ist.
- OpenBSD integriert neueste Sicherheitstechnologien, die geeignet sind, Firewalls und private Netzwerkdienste in verteilten Umgebungen zu errichten.
- OpenBSD profitiert von der raschen Entwicklung in vielen Bereichen, was die Möglichkeit zur Zusammenarbeit von neu entwickelten Technologien mit der internationalen Gemeinschaft der Programmierer und Endbenutzer betrifft.

1.5 - Wie kann ich OpenBSD unterstützen?

Wir sind allen Menschen und Organisationen zu Dank verpflichtet, die zum OpenBSD Projekt beigetragen haben. Sie werden namentlich auf der Spendenseite aufgeführt.

OpenBSD benötigt andauernd bestimmte Arten von Unterstützung von der Benutzergemeinschaft. Wenn du OpenBSD nützlich findest, dann solltest du einen Weg finden, um etwas beizutragen. Solltest du keinen der unten angeführten Vorschläge als für dich angemessen empfinden, dann schlag etwas Besseres vor, indem du eine E-Mail an donations@openbsd.org sendest.

- Kaufe ein OpenBSD CD Set. Es beinhaltet die aktuelle Vollversion von OpenBSD und ist auf vielen Plattformen bootfähig. Der Kauf trägt Geld zur Unterstützung des OpenBSD Projekts bei und reduziert die Belastung der Netzwerkressourcen, um die Distribution via Internet zu verbreiten. Dieses kostengünstige drei-CD Set beinhaltet den vollen Quelltext. Denke daran, auch deine Freunde benötigen ihre eigenen CDs!
- Spende Geld. Das Projekt benötigt andauernd Geld, um Hardware, Netzwerkanbindungen und CD Produktionskosten zu bestreiten. Die Produktion der CDs setzt die Vorfinanzierung durch die OpenBSD Entwickler ohne garantierte

Einnahmen voraus. Schick eine E-Mail an donations@openbsd.org, um herauszufinden, wie du etwas beisteuern kannst. Sogar kleine Spenden machen einen großen Unterschied.

- Spende Ausrüstung und Hardware. Das Projekt sucht immer normale und spezielle Hardware. Dinge wie IDE und SCSI Festplatten oder verschiedene Arten von RAM werden immer gerne genommen. Für andere Hardwaretypen wie Computersysteme und Motherboards solltest du den aktuellen Bedarf durch eine E-Mail an donations@openbsd.org erfragen.
- Steuere deine Zeit und Fähigkeiten bei. Programmierer, die gerne an Betriebssystemen schreiben, sind natürlich immer willkommen, aber es gibt buchstäblich Dutzende von anderen Möglichkeiten, um nützlich zu sein. Verfolge die Mailinglisten und beantworte Fragen von neuen Benutzern.
- Hilf uns, die Dokumentation auf dem Laufenden zu halten, indem du neues FAQ-Material einbringst (an faq@openbsd.org). Richte eine lokale Benutzergruppe ein und überzeuge deine Freunde von OpenBSD. Zeige deinem Arbeitgeber die Fähigkeiten von BSD für die Arbeit. Wenn du ein Student bist, sprich mit deinen Professoren über OpenBSD als Lehrmittel für EDV. Es ist auch noch erwähnenswert, einen der wichtigsten Wege aufzuzeigen, um dem OpenBSD Projekt "nicht" zu helfen: Beteilige dich nicht an Beschimpfungsorgien (Flamewars) in Usenet Newsgroups über andere Betriebssysteme. Dies bringt dem Projekt keine neuen Benutzer und schadet den wichtigen Beziehungen der Entwickler zu den anderen Entwicklern.

1.6 - Wer betreut das OpenBSD Projekt?

OpenBSD wird von einem Entwicklerteam betreut, das über viele verschiedene Länder verteilt ist. Das Projekt wird von Theo de Raadt koordiniert, der in Kanada wohnhaft ist.

1.7 - Wann erscheint die nächste Version von OpenBSD?

Das OpenBSD Team veröffentlicht alle 6 Monate eine neue Version, mit angestrebten Daten im Mai und November. Mehr Informationen zum Entwicklungsprozess gibt es hier.

1.8 - Was beinhaltet OpenBSD?

OpenBSD wird zusammen mit einer ganzen Anzahl von Software dritter Parteien veröffentlicht, einschließlich:

- XFree86 4.4.0, nicht betroffen von einer neuerlichen Änderung der Lizenz mit lokalen Patches. Für i386 werden auch v3.3 X Server mitgeliefert, um mehr Grafikkarten-Chips zu unterstützen. Wird durch die `x*.tgz` Installations Datei Sets installiert.

- GCC Versionen 2.95.3 und 3.3.2. GNU C Compiler. Das OpenBSD Team hat zusätzlich die Propolice Stack Protection Technologie eingebaut, die standardmäßig eingeschaltet ist und sowohl im OpenBSD Userland als auch bei allen Anwendungen genutzt wird, die unter OpenBSD übersetzt werden. Wird als Teil des `comp36.tgz` Datei Sets installiert.
- Perl 5.8.5 mit Patches und Verbesserungen des OpenBSD Teams.
- Apache 1.3.29 Webserver. Das OpenBSD Team hat standardmäßiges chrooting, ‚privilege revocation' und andere sicherheitsrelevante Verbesserungen eingefügt. Er beinhaltet auch mod_ssl 2.8.16 und Unterstützung für DSO.
- OpenSSL 0.9.7d mit Patches und Verbesserungen vom OpenBSD Team.
- Groff 1.15 Textprozessor.
- Sendmail 8.13.0 Mailserver.
- BIND 9.2.3 DNS Server. OpenBSD hat viele Verbesserungen im chroot Betrieb und andere sicherheitsrelevante Dinge eingefügt.
- Lynx 2.8.5rel.2 Text Webbrowser. Mit HTTPS Unterstützung plus Patches des OpenBSD Teams.
- Sudo v1.6.7p5, das den Benutzern erlaubt, einzelne Kommandos als root auszuführen.
- Ncurses 5.2.
- KAME IPv6.
- Heimdal 0.6rc1 mit Patches
- Arla 0.35.7
- OpenSSH 3.9
- gdb 6.1

Wie man sehen kann, patcht das OpenBSD Team oftmals Produkte dritter Parteien, um die Sicherheit und Qualität des Codes zu erhöhen. In einigen Fällen wird der Anwender keinerlei Änderungen im Betrieb bemerken, in einigen anderen Fällen GIBT es Unterschiede, die meist nur einzelne Anwender betreffen. Denke an diese Änderungen, bevor du einfach verschiedene Versionen derselben Software installierst. Du wirst vielleicht eine größere Versionsnummer haben, aber ein unsichereres System.

Natürlich können weitere Anwendungen weiterhin mittels des OpenBSD Packages und Ports Systems eingespielt werden.

1.9 - Was gibt es Neues in OpenBSD 3.6?

Die vollständige Liste der Änderungen an OpenBSD 3.5, um OpenBSD 3.6 zu erstellen, kann man hier einsehen, trotzdem gibt es hier eine Liste einiger Änderungen, die nach Ansicht des OpenBSD-Teams einer speziellen Erwähnung bedürfen, insbesondere für diejenigen, die von älteren Versionen auf OpenBSD 3.6 aktualisieren:

- **Symmetrischer MultiProzessor (SMP) Unterstützung ist für i386/amd64 da!** Ein separater Kernel (`bsd.mp`) wird für SMP Verwendungen zur Verfügung gestellt.
- **Eingliederung eines Network Time Protokoll Daemons (ntpd) in das Basis System.** Wie man es erwarten kann, wurde dieser Daemon entworfen, um solide, sicher und frei zu sein. Er ist ebenfalls recht einfach zu gebrauchen und es wird von ihm erwartet, den Großteil der Benutzer, die NTP Dienste benötigen, zufriedenzustellen. Wenn du bisher den ntpd aus den Ports verwendest, möchtest du wahrscheinlich einen Blick auf dieses neue Programm als eine mögliche Alternative werfen. Aktualisierer müssen bedenken, dass sie am Ende mit zwei ntpd Programmen da stehen und sicherstellen müssen, dass sie dasjenige Programm ausführen, das sie haben möchten.
- **Neue User und Gruppen.** Mehrere neue User und Gruppen wurden OpenBSD zum Zwecke der ‚privilege separation' hinzugefügt. Leute, die ihr System upgraden, müssen ihr `/etc` Verzeichnis sorgfältig, wie in der upgrade36.html beschrieben, aktualisieren.
- **Neues CD Booting System.** Bisher bootete OpenBSD von CD unter Verwendung der "Floppy Emulation". Dies begrenzte den Installationskernel auf eine Größe von 2,88M und nicht alle Computer können eine 2,88M Emulations-CDROM booten. Es wird gehofft, dass das neue "nicht-emulierte" Booten der CD ermöglichen wird, mit vielen neuen Computern gebootet werden zu können, die die bisherigen CDROMs nicht booten konnten.

1.10 - Kann ich OpenBSD als Desktop System benutzen?

Diese Frage wird oftmals genau in diesem Wortlaut gestellt -- ohne jegliche Erklärung, was der Fragesteller genau mit "Desktop" meint. Die einzige Person, die diese Frage beantworten kann, bist du selbst, nur du weißt, was deine Erwartungen und Anforderungen sind.

Während OpenBSD einen ziemlich guten Ruf als "Server" Betriebssystem hat, kann es auch auf dem Desktop benutzt werden und wird es auch bereits. Viele "Desktop" Anwendungen sind durch Ports und Packages verfügbar. Wie bei allen Entscheidungen über Betriebssysteme ist die Frage: kann es die Aufgabe in der Art und Weise bewältigen, die du möchtest? Diese Frage musst du dir selbst beantworten.

1.11 - Wieso ist *ProduktX* (nicht) enthalten?

Leute fragen oft, warum ein bestimmtes Produkt in OpenBSD enthalten ist, oder warum es nicht enthalten ist. Die Antwort basiert auf zwei Dingen: Die Wünsche der Entwickler und die Kompatibilität mit den Zielen des Projekts. Ein Produkt wird nicht einfach beigefügt, nur weil es "ordentlich" ist -- es muss auch als "frei" für die Verwendung, Weiterverbreitung und Modifikation von unseren Standards her gelten. Ein Produkt muss

auch stabil und sicher sein -- eine höhere Versionsnummer bedeutet nicht immer, dass es ein besseres Produkt ist.

Die Lizenz ist meistens das größte Problem: Wir möchten, dass OpenBSD von jeglicher Person auf der Welt für jeglichen Zweck verwendet werden kann.

Einige Fragen zu Produkten dritter Parteien, die häufig gestellt wurden:

- **Wieso wurde Sendmail eingefügt, es ist "als unsicher bekannt"?!** Sendmail hat eine mangelhafte Sicherheitsgeschichte, jedoch waren die Sendmail Autoren und Maintainer bereit, ihren Code zu bearbeiten, um ihn sicherer zu machen (und das ist leider eine ungewöhnliche Reaktion). Die aktuelle Sicherheitsgeschichte von Sendmail unterscheidet sich nicht sonderlich von denen der "sichereren" Alternativen.
- *Warum wurde Postfix nicht eingefügt?* Die Lizenz ist nicht frei und kann daher nicht in Betracht gezogen werden.
- *Warum wurde qmail oder djbdns nicht eingefügt?* Lizenz, oder das fehlen davon: Die Einschränkung, eine modifizierte Version dieser Software weiterzuverbreiten hält es davon ab, in Betracht gezogen zu werden.
- *Warum wurde Apache eingefügt? Es wird nicht von vielen Leuten gebraucht!* Weil die Entwickler es wollen.
- *Warum ist keine neuere Version von Apache dabei?* Die Lizenz der neueren Versionen ist nicht hinnehmbar.
- *Warum wurde XFree86 4.4 nicht eingefügt?* Die Änderungen an der Lizenz in v4.4 sind nicht hinnehmbar und daher hält OpenBSD, so wie viele andere Open Source Software Projekte, an der letzen freien Version fest, die "fast" v4.4 ist.

In den meisten Fällen wurden diese Themen bis ins kleinste Detail in den Mailinglisten diskutiert, siehe bitte in den Archiven nach, wenn du mehr Informationen benötigst.

Wenn du eines dieser Pakete einsetzen möchtest und deine Verwendung mit der Lizenz vereinbar ist, wird dich natürlich niemand davon abhalten (das wäre nicht sonderlich frei, wenn wir es tun würden, nicht wahr?). Bedenke jedoch, dass deine Wünsche sich ändern können -- du wirst bestimmt keine "Killer Anwendung" entwickeln wollen, die du dann nicht verkaufen, weitergeben oder reich davon werden kannst, weil du unfreie Software darin verarbeitet hast.

[FAQ Index] [Zum Kapitel 2 - Andere Informationsquellen zu OpenBSD]

 www@openbsd.org

$OpenBSD: faq1.html,v 1.55 2005/02/10 19:06:22 jufi Exp $

[FAQ Index] [Zum Kapitel 1 - Einführung in OpenBSD] [Zum Kapitel 3 - Wo man OpenBSD herbekommt]

2 - Andere Informationsquellen zu OpenBSD

Inhaltsverzeichnis

2.1 - Interessante Webseiten

Die offizielle Webseite des OpenBSD Projektes findest du unter: http://www.OpenBSD.org.

Hier kannst du viele wertvolle Informationen über alle Aspekte des OpenBSD Projektes erhalten.

Das OpenBSD Journal ist ein Neuigkeiten- und Meinungenportal, das auf OpenBSD fokusiert ist.

Viele Benutzer haben Webseiten und Homepages mit OpenBSD spezifischen Informationen erstellt. Wie mit allem im Internet, wird eine gute Suchmaschine dein Leben einfacher machen, wie es auch mit einer gesunden Dosis an Skepsis der Fall ist. Wie immer, gebe keine Befehle blindlings in deinen Computer ein, die du nicht verstehst.

2.2 - Mailinglisten

Das OpenBSD Projekt betreibt mehrere populäre Mailinglisten, die die Benutzer abonnieren und lesen sollten. Um eine Mailingliste zu abonnieren, schicke eine E-Mail an majordomo@openbsd.org. Diese Adresse ist ein automatischer Abonnementservice. Im Textkörper der Nachricht sollte in einer einzigen Zeile der Befehl stehen, um sich in die gewünschte Liste einzutragen. Zum Beispiel:

```
subscribe announce
```

Der Mailinglistendienst wird dir antworten und dich um Bestätigung bitten, so dass andere nicht in der Lage sind, dich mit einer Flut an ungewünschten E-Mails zu überschwemmen. Die Nachricht beinhaltet mehrere verschiedene Optionen zum Bestätigen, dazu gehören ein list server Webseiten Link und die Möglichkeit, der Bestätigungs E-Mail oder `majordomo@openbsd.org` zu antworten. Verwende die Option, die für dich am einfachsten ist. Du wirst feststellen, dass alle drei Varianten eine einzigarte und zeitbegrenzte Identitätsnummer beinhalten, wie zum Beispiel `A56D-70D4-52C3`, ebenfalls, um zu gewährleisten, dass *du* tatsächlich die Person bist, die den Mailinglisten-Eintrag angefordert hat (das ist *wirklich* ‚opt-in').

Nach deiner Bestätigung wirst du sofort in die Liste eingetragen und der Mailinglistendienst schickt dir eine Erfolgsbestätigung.

Um dich wieder von einer Liste abzumelden, schickst du eine Nachricht an majordomo@openbsd.org. Sie könnte etwa so aussehen:

```
unsubscribe announce
```

Solltest du Schwierigkeiten mit dem Mailinglistensystem haben, dann lies zunächst die Anweisungen und Hinweise, die du durch Schicken einer E-Mail an majordomo@openbsd.org mit dem Textkörper "help" erhalten kannst.

Dein Abonnement der OpenBSD Mailinglisten kann ebenfalls durch das Webinterface auf http://lists.openbsd.org verwaltet werden.

Einige der beliebtesten OpenBSD Mailinglisten sind:

- **announce** - Wichtige Ankündigungen. Sehr wenige E-Mails.
- **security-announce** - Bekanntmachungen von Sicherheitsempfehlungen. Sehr wenige E-Mails.
- **misc** - Benutzerfragen und Antworten. Dies ist die aktivste Liste und sollte daher für fast alle Fragen genutzt werden.
- **bugs** - Via sendbug(1) eingereichte Fehler und Diskussionen darüber.
- **source-changes** - Automatisierter Verteiler von Änderungen des CVS der Quelltexte. Jedes Mal, wenn ein Entwickler eine Änderung am Source Tree durchführt, wird CVS eine Kopie der (recht kurzen) Mitteilung der Änderung über diese Liste versenden.
- **ports** - Diskussionen über das OpenBSD Ports System.
- **ports-changes** - Automatische E-Mails der Ports-spezifischen CVS Source Tree Änderungen.
- **advocacy** - Diskussionen, um die Nutzung von OpenBSD populärer zu machen, und Themen, die einfach zu ‚off-topic' für **misc** sind.

Bevor du eine Frage auf **misc** oder einer anderen Mailingliste postest, überprüfe bitte zunächst, ob deine Frage nicht schon in den Archiven auftaucht, da die meisten Fragen schon einmal gestellt wurden. Auch wenn du das Problem zum ersten Mal siehst, ist es anderen doch oftmals schon begegnet, wurden die Mailinglisten vielleicht schon letzte Woche damit überschwemmt und freuen sich die Leser sicher nicht auf eine Wiederholung. Falls du eine Frage stellst, die möglicherweise mit Hardware zu tun hat, *füge immer eine dmesg(8) mit ein!*

Du kannst einige Archive, andere Mailinglisten Richtlinien und weitere Informationen auf der Mailinglisten Seite finden.

Eine unoffizielle Mailingliste, die insbesondere für neue Anwender von OpenBSD interessant sein könnte, ist die OpenBSD Newbies Liste.

2.3 - Manual Seiten

OpenBSD wird von einer ausführlichen Dokumentation in Form von Manual Seiten (manual pages) und weiteren, längeren Artikeln, die sich auf spezielle Anwendungen beziehen, begleitet. Besonders viel Wert wird darauf gelegt, sicherzustellen, dass die Manual Seiten aktuell und genau sind. In allen Fällen werden die Manual Seiten als maßgebliche Informationsquelle für OpenBSD angesehen.

Um die Manual Seiten lesen zu können, müssen die `man36.tgz` und `misc36.tgz` Datei Sets installiert sein.

Hier eine Liste der nützlichsten Manual Seiten für Anfänger:

Einführung

- **afterboot(8)** - Dinge, die man nach dem ersten Start beachten sollte.
- **help(1)** - Hilfe für neue Benutzer und Administratoren.
- **hier(7)** - Layout der Dateisysteme.
- **man(1)** - Anzeigen der on-line Manual Seiten.
- **intro(1)** - Einführung für generelle Befehle, siehe auch die Einführungen der anderen Kapitel des Manuals: intro(2), intro(3), intro(4) (beachte: intro(4) ist Plattform abhängig), intro(5), intro(6), intro(7), intro(8) und intro(9).
- **adduser(8)** - Befehl, um neue Benutzer anzulegen.
- **vipw(8)** - Editieren der Passwortdatei.
- **disklabel(8)** - Auslesen und Schreiben des Plattenlayouts.
- **reboot, halt(8)** - Neustarten und Stoppen des Systems.
- **shutdown(8)** - Runterfahren des Systems zu einer bestimmten Zeit.
- **dmesg(8)** - Zeige die Kernel Boot Meldungen erneut.
- **sudo(8)** - Logge dich nicht als root ein, aber führe Befehle als root aus.
- **mg(1)** - emacs-ähnlicher Text Editor.

Für fortgeschrittenere Anwender

- **boot(8)** - Systemstartprozeduren.
- **login.conf(5)** - Format der Passwortkonfigurationsdatei.
- **ifconfig(8)** - Konfiguration der Netzwerkadapter Parameter.
- **netstat(1)** - Anzeigen des Netzwerkstatus.

- **boot_config(8)** - Änderung der Kernelkonfiguration beim Starten.
- **release(8)** - Erzeuge ein OpenBSD Release.
- **sendbug(1)** - **Schicken eines Fehlerberichtes über OpenBSD an die Supportzentrale.**
- **sysctl(8)** - **Lese oder setze Kernel Status.**
- **style(9)** - **OpenBSD Kernel Source Code Style Guide.**

Die OpenBSD Manual Seiten sind im Web über http://www.openbsd.org/cgi-bin/man.cgi zu finden oder auch auf deinem eigenen Computer, wenn du das `man36.tgz` Datei Set installierst.

Im Allgemeinen kannst du die Manual Seite eines dir namentlich bekannten Befehls erreichen, indem du "man befehl" eingibst. Zum Beispiel: "man vi" bringt dir die Manual Seite des vi Editors. Solltest du den genauen Namen des Befehls nicht wissen oder "man befehl" nicht die gewünschte Manual Seite bringt, kannst du mittels "apropos irgendwas" oder "man -k irgendwas" nach dem Namen des Befehls suchen, wobei "irgendwas" möglicherweise in der von dir gesuchten Manual Seite enthalten sein sollte. Zum Beispiel:

```
# apropos "time zone"
tzfile (5) - time zone information
zdump (8) - time zone dumper
zic (8) - time zone compiler
```

Die Zahlen in Klammern deuten auf das Kapitel, in dem sich die Manual Seite befindet. In einigen Fällen enthalten die Manual Seiten den gleichen Befehl in verschieden Kapiteln. Zum Beispiel: Du möchtest das Format der Konfigurationsdatei des cron Daemons wissen. Wenn du einmal weißt, in welchem Kapitel sich die gewünschte Manual Seite befindet, kannst du mittels "man n befehl", wobei n die Kapitelnummer ist, die gewünschte Seite direkt aufrufen.

```
# man -k cron
cron (8) - clock daemon
crontab (1) - maintain crontab files for individual users
crontab (5) - tables for driving cron
# man 5 crontab
```

Zusätzlich zu den UNIX Manual Seiten gibt es noch einen weiteren Dokumentensatz (enthalten im `misc36.tgz` Datei Set). Dieser befindet sich im `/usr/share/doc` Verzeichnis. Wenn du auch die text Distribution installiert hast, dann kannst du jedes Dokument mittels "make" im entsprechenden Verzeichnis formatieren. Das `psd` Unterverzeichnis beinhaltet die "Programmer's Supplementary Documents" Distribution. Im smm Unterverzeichnis liegt das "System Manager's Manual". Im `usd` Unterverzeichnis findest du die "UNIX User's Supplementary Documents" Distribution. Du kannst "make" in den drei Distributionsunterverzeichnissen starten oder ein spezielles Kapitel einer Distribution auswählen und dort ein "make" in dessen Unterverzeichnis ausführen.

Einige der Unterverzeichnisse sind leer. Standardmäßig werden die Dokumente im Postscriptformat ausgegeben, das sich zum Ausdrucken eignet. Die Postscriptausgabe kann sehr groß werden -- etwa 250-300% Zuwachs an Größe. Ohne Postscriptdrucker oder -anzeige kannst du die Dokumente auch für das Lesen auf einer Terminalanzeige formatieren. Jedes Dokumentenunterverzeichnis hat ein Ziel zum Erzeugen von ASCII Kopien von diesen Dokumenten (genannt ‚paper.txt'), welches mit make(1) generiert werden kann. Zum Beispiel:

```
# cd /usr/share/doc/usd/04.csh
# make paper.txt
# more paper.txt
```

Bedenke, dass superuser Privilegien benötigt sein können, um die Dokumente in diesen Verzeichnissen zu erstellen, und der Aufruf von make clean alle Dokumente löscht, die von vorherigen make Aufrufen erstellt worden sind. Siehe /usr/share/doc/README für weiter Details über die Dokumente in /usr/share/doc/.

Die UNIX Manual Seiten sind im Allgemeinen aktueller und vertrauenswürdiger als die formatierten Dokumente, die aber manchmal kompliziertere Anwendungen in größerem Detailumfang als die Manual Seiten erklären.

Für viele ist eine ausgedruckte Manual Seite nützlich. Hier folgen die Richtlinien, um eine Manual Seite auszudrucken.

Wie kann ich einen Manual Seiten Quelltext anzeigen (d.h. eine Datei, deren Namen in einer Zahl endet, wie tcpdump.8)?

Dies gilt für den gesamten Source Tree. Die Manual Seiten befinden sich unformatiert im Verzeichnisbaum und werden automatisch durch CVS aktualisiert. Um sich diese Seiten anzusehen:

```
# nroff -Tascii -mandoc <Datei> | more
```

Wie bekomme ich eine reine Manual Seite ohne Formatierung oder Escapesequenzen?

Dies ist nützlich, um die Manual Seite ohne nicht druckbare Zeichen zu erhalten.
Beispiel:

```
# man <Befehl> | col -b
```

Wie erhalte ich eine postscriptformatierte, druckreife Ausgabe einer Manual Seite?

Beachte, dass `<Datei>` die Manual Seiten Quelltextdatei ist (wahrscheinlich eine Datei, die in einer Zahl endet; z.B. `tcpdump.8`). Die Postscriptversionen der Manual Seiten sehen sehr gut aus. Sie können ausgedruckt oder am Bildschirm mit einem Programm wie gv (GhostView) betrachtet werden. GhostView kannst du in unserem Ports Tree finden. Benutze die folgenden nroff(1) Befehlsoptionen, um eine PostScript Version von einer OpenBSD System Manual Seite zu bekommen:

```
# nroff -Tps -mandoc <Datei> > Ausgabedatei.ps
```

Wie erstelle ich komprimierte Ausgaben der Manual Seiten?

Für Leute, die ihr System vom Sourcecode aus erzeugen, gibt es einige Optionen, um Manual Seiten zu entwickeln. Diese Optionen können in /etc/mk.conf (vielleicht muss diese Datei erst angelegt werden) geschrieben und somit in die Systemerzeugung mit integriert werden. Eine besonders gebräuchliche Option ist, komprimierte Manual Seiten zu erzeugen, um Plattenplatz sparen zu können. Diese Dateien können auf den üblichen Weg angezeigt werden, unter der Verwendung des man Befehls. Um diese Option zu setzen, füge das folgende in `/etc/mk.conf` ein:

```
MANZ=yes
```

Eine andere nützliche Option ist, während der Systemkompilierung die Manual Seiten im Postscript Format sowie als ASCII Text entwickeln zu lassen. Dies wird durch das Setzen der Option `MANPS=yes` in /etc/mk.conf gemacht. Siehe mk.conf(5) für weitere Einzelheiten.

Was sind info Dateien?

Einige Teile der Dokumentation für OpenBSD kommt in info Dateien, typischerweise befinden sie sich im `/usr/share/info` Verzeichnis. Dies ist eine alternative Form von Dokumentation, die von GNU angeboten wird. Viele dieser Dateien sind aktueller als die Manual Seiten von GNU und können mit dem info(1) Befehl betrachtet werden. Um zum Beispiel die Informationen über den GNU Compiler gcc(1) anzusehen, gib ein:

```
# info gcc
```

Nach der Verwendung von info wirst du unsere Manual Seiten zu schätzen wissen!

Wie bekomme ich farbige Manual Seiten im XTerm?

Die standardmäßige Konfigurationsdatei für xterm(1) zeigt keine farbigen Manual Seiten an. Um farbige Ausgabe zu erhalten, kopiere die Datei `/etc/X11/app-defaults/XTerm-color` in dein Heimatverzeichnis und benenne sie in "`.Xdefaults`" um. Sei vorsichtig, damit du nicht irgendwelche aktuellen Einstellungen in "`.Xdefaults`" überschreibst. Diese Datei beinhaltet alle Einstellungen, um Farbe im XTerm zu aktivieren. Allerdings müssen diese drei Zeilen auskommentiert werden, bevor sie funktionieren können:

```
!*VT100*colorULMode: on
```

```
!*VT100*underLine: off
!*VT100*colorBDMode: on
```

Der Rest der Datei erlaubt dir, Farben für verschiedene Einstellungen zu wählen. Relevant für die Manual Seiten sind:

```
*VT100*colorUL: yellow
*VT100*colorBD: white
```

Dies produziert recht helle Manual Seiten, daher ändere es nach deinen Bedürfnissen: Dürfen wir rot für "colorUL" und magenta für "colorBD" empfehlen? Es ist ebenfalls ein Manual Seiten Betrachter für X11 verfügbar, xman, welcher eine alternative (grafische) Oberfläche für die Manual Seiten bietet.

Wie schreibe ich meine eigene Manual Seite?

Falls du eine eigene Manual Seite für dein selbstentwickeltes Programm schreiben möchtest, kannst du die Anleitung, die in mdoc.samples(7) zur Verfügung gestellt wird, verwenden. Ein weiterer handlicher Referenz Guide wird in mdoc(7) bereitgestellt.

2.4 - Melden von Fehlern (bug reports)

Bevor du "Fehler!" schreist, stelle sicher, dass du es tatsächlich mit einem zu tun hast. Falls du stattdessen nicht verstehst, wie etwas in OpenBSD funktioniert und auch das Problem nicht mit Hilfe der Manual Seiten oder der OpenBSD Website lösen kannst, verwende die Mailinglisten (für gewöhnlich misc@openbsd.org), um nach Hilfe zu fragen. Falls dies deine erste OpenBSD Erfahrung ist, sei realistisch: du wirst vermutlich keinen unbekannten Fehler gefunden haben. Bedenke auch, dass fehlerhafte Hardware einen Softwarefehler vortäuschen kann, bitte überprüfe den aktuellen Zustand deiner Hardware, bevor du entscheidest, dass du einen "Fehler" gefunden hast.

Schließlich, bevor du einen Fehler meldest, solltest du http://www.openbsd.org/report.html lesen.

Richtige Fehlerberichte sind eine der wichtigsten Aufgaben von Endbenutzern. Sehr detaillierte Informationen werden benötigt, um die schwersten Fehler zu diagnostizieren. Entwickler erhalten häufig Fehlerberichte via E-Mail wie diese:

```
From: joeuser@example.com
To: bugs@openbsd.org
Subject: HELP!!!

I have a PC and it won't boot!!!!! It's a 486!!!!!
```

Hoffentlich verstehen die meisten Menschen, warum solche Fehlerberichte gelöscht werden. Jeder Fehlerbericht sollte detaillierte Informationen enthalten. Wenn ein normaler Benutzer wirklich die Untersuchung seines Fehlers wünscht, dann sollte sein Fehlerbericht in etwa so aussehen:

```
From: smartuser@example.com
To: bugs@openbsd.org
Subject: 3.3-beta panics on a SPARCStation2

OpenBSD 3.2 installed from an official CD-ROM installed and ran fine
on this machine.

After doing a clean install of 3.3-beta from an FTP mirror, I find the
system randomly panics after a period of use, and predictably and
quickly when starting X.

This is the dmesg output:

OpenBSD 3.3-beta (GENERIC) #9: Mon Mar 17 12:37:18 MST 2003
    deraadt@sparc.openbsd.org:/usr/src/sys/arch/sparc/compile/GENERIC
real mem = 67002368
avail mem = 59125760
```

```
using 200 buffers containing 3346432 bytes of memory
bootpath: /sbus@1,f8000000/esp@0,800000/sd@1,0
mainbus0 (root): SUNW,Sun 4/75
cpu0 at mainbus0: CY7C601 @ 40 MHz, TMS390C602A FPU; cache chip bug
- trap page uncached
cpu0: 64K byte write-through, 32 bytes/line, hw flush cache enabled
memreg0 at mainbus0 ioaddr 0xf4000000
clock0 at mainbus0 ioaddr 0xf2000000: mk48t02 (eeprom)
timer0 at mainbus0 ioaddr 0xf3000000 delay constant 17
auxreg0 at mainbus0 ioaddr 0xf7400003
zs0 at mainbus0 ioaddr 0xf1000000 pri 12, softpri 6
zstty0 at zs0 channel 0 (console i/o)
zstty1 at zs0 channel 1
zs1 at mainbus0 ioaddr 0xf0000000 pri 12, softpri 6
zskbd0 at zs1 channel 0: reset timeout
zskbd0: no keyboard
zstty2 at zs1 channel 1: mouse
audioamd0 at mainbus0 ioaddr 0xf7201000 pri 13, softpri 4
audio0 at audioamd0
sbus0 at mainbus0 ioaddr 0xf8000000: clock = 20 MHz
dma0 at sbus0 slot 0 offset 0x400000: rev 1+
esp0 at sbus0 slot 0 offset 0x800000 pri 3: ESP100A, 25MHz, SCSI ID 7
scsibus0 at esp0: 8 targets
sd0 at scsibus0 targ 1 lun 0: <SEAGATE, ST1480 SUN0424, 8628> SCSI2 0/direct fixed
sd0: 411MB, 1476 cyl, 9 head, 63 sec, 512 bytes/sec, 843284 sec total
sd1 at scsibus0 targ 3 lun 0: <COMPAQPC, DCAS-32160, S65A> SCSI2 0/direct fixed
sd1: 2006MB, 8188 cyl, 3 head, 167 sec, 512 bytes/sec, 4110000 sec total
le0 at sbus0 slot 0 offset 0xc00000 pri 5: address 08:00:20:13:10:b9
le0: 16 receive buffers, 4 transmit buffers
cgsix0 at sbus0 slot 1 offset 0x0: SUNW,501-2325, 1152x900, rev 11
wsdisplay0 at cgsix0
wsdisplay0: screen 0 added (std, sun emulation)
fdc0 at mainbus0 ioaddr 0xf7200000 pri 11, softpri 4: chip 82072
fd0 at fdc0 drive 0: 1.44MB 80 cyl, 2 head, 18 sec
root on sd0a
rootdev=0x700 rrootdev=0x1100 rawdev=0x1102
```

This is the panic I got when attempting to start X:

```
panic: pool_get(mclpl): free list modified: magic=78746572; page 0xfaa93000;
 item addr 0xfaa93000
Stopped at      Debugger+0x4:   jmpl            [%o7 + 0x8], %g0
RUN AT LEAST 'trace' AND 'ps' AND INCLUDE OUTPUT WHEN REPORTING THIS PANIC!
DO NOT EVEN BOTHER REPORTING THIS WITHOUT INCLUDING THAT INFORMATION!
ddb> trace
pool_get(0xfaa93000, 0x22, 0x0, 0x1000, 0x102, 0x0) at pool_get+0x2c0
sosend(0x16, 0xf828d800, 0x0, 0xf83b0900, 0x0, 0x0) at sosend+0x608
soo_write(0xfac0bf50, 0xfac0bf70, 0xfac9be28, 0xfab93190, 0xf8078f24, 0x0)
at soo_write+0x18
dofilewritev(0x0, 0xc, 0xfac0bf50, 0xf7fff198, 0x1, 0xfac0bf70) at
dofilewritev+0x12c
sys_writev(0xfac87508, 0xfac9bf28, 0xfac9bf20, 0xf80765c8, 0x1000, 0xfac0bf70)
at sys_writev+0x50
syscall(0x79, 0xfac9bfb0, 0x0, 0x154, 0xfcffffff, 0xf829dea0) at syscall+0x220
slowtrap(0xc, 0xf7fff198, 0x1, 0x154, 0x1, 0xfac87508) at slowtrap+0x1d8
ddb> ps

   PID   PPID   PGRP    UID  S       FLAGS  WAIT       COMMAND
 27765   8819  29550      0  3        0x86  netio      xconsole
  1668  29550  29550      0  3      0x4086  poll       fvwm
 15447  29550  29550      0  3     0x44186  poll       xterm
  8819  29550  29550     35  3      0x4186  poll       xconsole
```

```
  1238  29550  29550      0  3      0x4086  poll       xclock
 29550  25616  29550      0  3      0x4086  pause      sh
  1024  25523  25523      0  3     0x40184  netio      XFree86
*25523  25616  25523     35  2     0x44104             XFree86
 25616  30876  30876      0  3      0x4086  wait       xinit
 30876  16977  30876      0  3      0x4086  pause      sh
 16977      1  16977      0  3      0x4086  ttyin      csh
  5360      1   5360      0  3        0x84  select     cron
 14701      1  14701      0  3     0x40184  select     sendmail
 12617      1  12617      0  3        0x84  select     sshd
 27515      1  27515      0  3       0x184  select     inetd
  1904      1   1904      0  2        0x84             syslogd
  9125      1   9125      0  3        0x84  poll       dhclient
     7      0      0      0  3    0x100204  crypto_wa  crypto
     6      0      0      0  3    0x100204  aiodoned   aiodoned
     5      0      0      0  3    0x100204  syncer     update
     4      0      0      0  3    0x100204  cleaner    cleaner
     3      0      0      0  3    0x100204  reaper     reaper
     2      0      0      0  3    0x100204  pgdaemon   pagedaemon
     1      0      1      0  3      0x4084  wait       init
     0     -1      0      0  3     0x80204  scheduler  swapper
```

```
Thank you!
```

In report.html finden sich weitere Informationen über das Erzeugen und Einsenden von Fehlerberichten (bug reports). Detaillierte Informationen über deine Hardware sind absolut notwendig, wenn du glaubst, der Fehler könnte *irgendwie* mit deiner Hardware oder deiner Hardware-Konfiguration zusammenhängen. Normalerweise ist die Ausgabe von dmesg(8) in dieser Hinsicht ausreichend. Eine detaillierte Beschreibung deines Problems ist notwendig. dmesg beschreibt deine Hardware, der Text erklärt warum Smart User denkt, das System sei defekt (3.2 lief noch bestens), wie dieser Crash erzeugt wurde (indem er X gestartet hat) und die Ausgabe der Debugger-Befehle "`ps`" und "`trace`". In diesem Fall hat Smart User die Informationen bereitgestellt, die er via serieller Konsole bekommen hat, wenn du das nicht tun kannst, bist du gezwungen, mittels Stift und Papier die Informationen vom Crash aufzuzeichnen. (Das Beispiel oben war im Übrigen ein echtes Problem und die oben bereitgestellte Information führte dazu, dass dieses Problem, das Sun4c Systeme betraf, behoben werden konnte).

Wenn Smart User ein funktionierendes OpenBSD System gehabt hätte, von dem aus er einen Fehlerbericht abschicken wolle, dann hätte er sendbug(1) verwendet, um den Fehlerbericht zu verfassen und zum GNATS Problemerfassungssystem zu senden. Wenn dein System nicht startet, dann kannst du sendbug(1) selbstverständlich nicht verwenden, aber du solltest es nutzen, wann immer du kannst. Du wirst noch zusätzliche Informationen, wie z.B. was geschah und deine genaue Konfiguration einfügen müssen und wie man das Problem reproduzieren kann. Der sendbug(1) Befehl benötigt eine Verbindung zum Internet und einen funktionsfähigen MTA, um den Fehlerbericht per E-Mail versenden zu können. Beachte, dass der Mail Server spamd(8) basierendes ‚greylisting' verwendet, so dass es eine halbe Stunde oder so dauern kann, bevor der Mail Server deine Fehlermeldung akzeptiert, habe also bitte Geduld.

Nach dem Einsenden einer Fehlermeldung via sendbug(1) wirst du per E-Mail über den Status informiert. Du kannst auch von Entwicklern kontaktiert werden, die dich nach weiteren Informationen fragen oder dich bitten, Patches zu testen. Desweiteren kannst du dir auch die Archive der `bugs@openbsd.org` Mailingliste ansehen, Details dazu gibt es auf der Mailinglisten-Seite oder du kannst den Status der Fehlermeldung auch online abfragen, und zwar im Bug Tracking System.

Weitere Informationen, um nützliche Informationen für Entwickler zu bekommen

Hier sind ein paar zusätzliche Tipps:

Die "Panic message" verloren?
Unter bestimmten Umständen kannst du die ersten Zeilen vom ‚panic', die den Grund für den ‚panic' angeben, verlieren. Dies ist eine sehr wichtige Nachricht, daher möchtest du sie ebenfalls melden. Du kannst diese wiederbekommen, indem du den "x/s *panicstr" (eXamine String *panicstr) Befehl in ddb> wie folgt verwendest:

```
ddb> x/s *panicstr
0:      kernel: page fault trap, code=0
ddb>
```

In diesem Fall war die panic Zeichenkette "Kernel: page fault trap, code=0"

Besonderer Hinweis für SMP Systeme:
Du solltest einen "trace" von jedem Prozessor als Teil deiner Meldung holen:

```
ddb{0}> trace
pool_get(d05e7c20,0,dab19ef8,d0169414,80) at pool_get+0x226
fxp_add_rfabuf(d0a62000,d3c12b00,dab19f10,dab19f10) at fxp_add_rfabuf+0xa5
fxp_intr(d0a62000) at fxp_intr+0x1e7
Xintr_ioapic0() at Xintr_ioapic0+0x6d
--- interrupt ---
idle_loop+0x21:
ddb{0}> machine ddb 1
Stopped at      Debugger+0x4:   leave
ddb{1}> trace
Debugger(d0319e28,d05ff5a0,dab1bee8,d031cc6e,d0a61800) at Debugger+0x4
i386_ipi_db(d0a61800,d05ff5a0,dab1bef8,d01eb997) at i386_ipi_db+0xb
i386_ipi_handler(b0,d05f0058,dab10010,d01d0010,dab10010) at i386_ipi_handler+0x
4a
Xintripi() at Xintripi+0x47
--- interrupt ---
i386_softintlock(0,58,dab10010,dab10010,d01e0010) at i386_softintlock+0x37
Xintrltimer() at Xintrltimer+0x47
--- interrupt ---
idle_loop+0x21:
ddb{1}>
```

Wiederhole den "`machine ddb x`" gefolgt von "`trace`" für jeden einzelnen Prozessor in deiner Maschine.

[FAQ Index] [Zum Kapitel 1 - Einführung in OpenBSD] [Zum Kapitel 3 - Wo man OpenBSD herbekommt]


www@openbsd.org
$OpenBSD: faq2.html,v 1.46 2005/03/12 17:55:51 jufi Exp $

OpenBSD

[FAQ Index] [Zum Kapitel 2 - Andere Informationsquellen zu OpenBSD] [Zum Kapitel 4 - Installationsanleitung]

3 - Wo man OpenBSD herbekommt

Inhaltsverzeichnis

3.1 - Eine OpenBSD CD kaufen

Eine OpenBSD CD zu kaufen ist der allgemein beste Weg, zu beginnen. Besuche die Bestellseite um deine Kopie zu erwerben: OpenBSD Bestellseite.

Es gibt mehrere gute Gründe, eine OpenBSD CD zu besitzen:

- CD Verkäufe unterstützen die Weiterentwicklung von OpenBSD.
- Die Entwicklung eines plattformübergreifenden Betriebssystems benötigt andauernd Investitionen in Hardware.
- Deine Unterstützung in Form eines CD Kaufes hat einen realen Einfluss auf zukünftige Entwicklungen.
- Die CD beinhaltet Binärdateien (und Quelltexte) für alle unterstützten Plattformen.
- Die CD ist bootfähig auf mehreren Plattformen und man kann mittels CD das Betriebssystem direkt installieren.
- Mit der CD kann man auch die Installation starten, um einen Snapshot zu installieren.
- Die Installation von CD ist schneller! Und Netzwerkressourcen werden eingespart.
- Die OpenBSD CDs beinhalten immer sehr hübsche Aufkleber. Dein System ist nicht komplett, solange du sie nicht hast. Diese Aufkleber kannst du nur durch den Kauf einer CD oder Spenden von Hardware erwerben.

Wenn du eine offizielle Version von OpenBSD installierst, dann solltest du eine CD benutzen.

3.2 - OpenBSD T-Shirts kaufen

Ja, OpenBSD hat T-Shirts für dein Tragevergnügen. Hier kannst du sie sehen: OpenBSD T-Shirts Seite. Nett, oder? :)

3.3 - Gibt es ein OpenBSD ISO Image?

Einige andere Open Source Betriebssysteme werden üblicherweise als CD-ROM Images verbreitet. Das ist bei OpenBSD *nicht* der Fall.

Das OpenBSD Projekt stellt die offiziellen ISO Images nicht zur Verfügung, die als Master für die offiziellen CDs benutzt werden. Der Grund dafür ist einfach, dass wir gerne die CDs verkaufen möchten, und zwar, um die weitere Entwicklung von OpenBSD sicherzustellen. Das offizielle OpenBSD CD-ROM Layout unterliegt dem Copyright von Theo de Raadt. Theo erlaubt es niemandem, die Images der offiziellen CD weiter zu verbreiten. Als weiteren Anreiz zum Kauf der CDs gibts es zusätzlich noch ‚artwork' und Aufkleber.

Denk aber daran, dass nur das CD Layout unter Copyright liegt, OpenBSD selbst ist frei. Nichts hindert irgendjemanden daran, sich OpenBSD zu besorgen und seine eigene CD zu machen. Wenn du aus irgendeinem Grund ein CD Image herunterladen willst, sieh im Archiv der Mailinglisten nach, dort gibt es Quellen. Natürlich gibt es dabei nur zwei Möglichkeiten: Entweder verletzen die im Internet erhältlichen Images das Copyright von Theo de Raadt oder sind einfach keine offiziellen Images. Die Quelle von inoffiziellen Images sind entweder vertrauenswürdig oder eben nicht, die Entscheidung liegt bei dir.

Wir schlagen einfach allen Leuten vor, die OpenBSD umsonst haben wollen, die FTP Installationsoption zu wählen. Diejenigen, die eine bootbare CD für ihr System benötigen, können Bootdisk ISO Images (namens `cd36.iso`) für eine Reihe von Plattformen herunterladen, die es dann erlauben, den Rest des Systems via FTP zu installieren. Diese ISO Images haben nur eine Größe von wenigen MB und enthalten nichts weiter als die Installationstools, aber nicht die wirklichen Dateien.

3.4 - Downloaden via FTP, HTTP oder AFS

Es gibt zahlreiche internationale FTP Server, die OpenBSD Versionen und Snapshots führen. AFS Zugang ist auch möglich. Du solltest immer den dir nächsten Server wählen. Bevor du mit dem Runterladen einer Version oder eines Snapshots beginnst, solltest du mittels ping(8) und traceroute(8) feststellen, welcher Server für dich am schnellsten ist. Klarerweise ist das offizielle OpenBSD CD Set immer näher als jeder Server. Zugangsinformationen findest du hier:

OpenBSD FTP Seite.

3.5 - Wo man den aktuellsten Sourcecode (current) herbekommt

Die Quelltexte von OpenBSD dürfen frei verbreitet werden und sind frei erhältlich. Im Allgemeinen ist der beste Weg, den Verzeichnisbaum mit den aktuellen Quelltexten aufzubauen, die Quelltexte von der letzten CD zu installieren und sie dann mittels AnonCVS regelmäßig zu erneuern. Informationen über AnonCVS findest du hier:

OpenBSD AnonCVS Seite.

Sollte deine Netzwerkanbindung für AnonCVS unzureichend sein oder dein Internetzugang via UUCP besteht, kannst du die Quelltexte mittels CTM auf dem Laufenden halten. Solltest du dich in dieser Situation befinden, dann ist das Beginnen mit der CD umso wichtiger. Informationen über CTM findest du hier:

OpenBSD CTM Seite.

Eine weitere Möglichkeit, an den Quelltext zu kommen bietet dir das Webinterface "cvsweb" an: http://www.openbsd.org/cgi-bin/cvsweb/.

[FAQ Index] [Zum Kapitel 2 - Andere Informationsquellen] [Zum Kapitel 4 - Installationsanleitung]


www@openbsd.org

$OpenBSD: faq3.html,v 1.42 2005/02/10 19:06:22 jufi Exp $

OpenBSD

[FAQ Index] [Zum Kapitel 3 - Wo man OpenBSD herbekommt] [Zum Kapitel 5 - Das System aus dem Source-Code erzeugen]

4 - OpenBSD 3.6 Installationsanleitung

Inhaltsverzeichnis

4.1 - Übersicht der OpenBSD Installationsprozedur

OpenBSD hat eine robuste und anpassungsfähige text-basierte Installationsroutine und kann von einer einzelnen Diskette aus installiert werden. Die meisten Architekturen folgen einer ähnlichen Installationsroutine; allerdings gibt es Unterschiede

in den Details. In jedem Fall wird dringend dazu geraten, das plattform-spezifische INSTALL Dokument im *Plattform* Verzeichnis auf der CD-ROM oder der FTP Seiten (z.B. `i386/INSTALL.i386`, `mac68k/INSTALL.mac68k` oder `sparc/INSTALL.sparc`) durchzulesen.

Auf den meisten Plattformen benutzt die OpenBSD Installation einen speziellen Kernel mit einigen Utilities und Installationsskripten, die sich auf einer vorgeladenen RAM Disk befinden. Nachdem dieser Kernel gebootet wurde, wird das Betriebssystem aus einigen komprimierten tar(1) (`.tgz`) Dateien extrahiert. Es existieren mehrere Wege, diesen Installationskernel zu booten:

- **Floppy:** Floppy Images werden zur Verfügung gestellt, die benutzt werden können, um Installationsdisketten auf einem anderen Unix-ähnlichen System oder DOS/Windows System zu erstellen. Typische Dateinamen sind `floppy36.fs`, obwohl für einige Plattformen mehrere Floppy Images vorhanden sind.
- **CD-ROM:** Für einige Plattformen wird ein CD-ROM Image (`cd36.iso`) bereitgestellt, das die Erstellung von einer bootfähigen CD-ROM ermöglicht. Dies beinhaltet nur den Installationskernel - Installationsdateien müssen weiterhin via FTP oder einer anderen Quelle beschaffen werden. Du kannst natürlich deine eigene CD-ROM erstellen, die mit allen Dateien und Programmen ausgestattet ist, die du haben möchtest.
- **bsd.rd:** Der RAM Disk Kernel, der dafür ausgelegt ist, entweder von einer bereits existierenden OpenBSD Partition oder über das Netzwerk gebootet zu werden.
- **Netzwerk:** Manche Systeme unterstützen das Booten über das Netzwerk.
- **Ein Dateisystem Image auf Platte schreiben:** ein Dateisystem Image, das auf eine existierende Partition geschrieben und daraufhin gebootet werden kann.
- **Bootfähiges Band:** Einige Systeme unterstützen das Booten von Band. Diese Bänder können anhand der INSTALL.*Plattform* Anleitung erstellt werden.

Nicht jede Plattform unterstützt alle Boot Optionen:

- **alpha:** Floppy, CD-ROM, das Schreiben eines Floppy Images auf Festplatte.
- **amd64:** Floppy, CD-ROM, Netzwerk.
- **cats:** CD-ROM.
- **hp300:** CD-ROM, Netzwerk.
- **hppa:** Netzwerk.
- **i386:** Floppy, CD-ROM, Netzwerk.
- **mac68k:** Installiert (und gebootet) unter der Verwendung von Anwendungen unter Mac OS. Siehe INSTALL.mac68k für Details.
- **macppc:** CD-ROM, Netzwerk.
- **mvme68k:** Netzwerk, bootfähiges Band.
- **mvme88k:** Netzwerk, bootfähiges Band.
- **sparc:** Floppy, CD-ROM, Netzwerk, Schreiben eines Images auf eine existierende Partition.
- **vax:** Floppy, Netzwerk.

Alle Plattformen mit Ausnahme von mac68k können ebenfalls bsd.rd zum Neuinstallieren oder Aktualisieren verwenden.

Wenn der Installationskernel erst einmal gebootet ist, hast du mehrere Möglichkeiten, von wo aus du die Installations-Datei Sets herbekommen kannst. Auch hier bietet nicht jede Plattform alle Möglichkeiten.

- **CD-ROM:** Natürlich bevorzugen wir, dass du das offizielle CD-ROM Set verwendest, aber für spezielle Notwendigkeiten kannst du auch dein eigenes erstellen.
- **FTP:** Entweder eine der OpenBSD FTP mirror Seiten oder deinen eigenen FTP Server, der die Datei Sets hält.
- **HTML:** Entweder eine der OpenBSD HTTP mirror Seiten oder deinen eigenen Webserver, der die Datei Sets hält.
- **Lokale Festplatten-Partition:** In vielen Fällen kannst du die Datei Sets von einer anderen Partition einer lokalen Festplatte aus installieren. Zum Beispiel auf i386 kannst du von einer FAT Partition oder einer CD-ROM mit ISO9660, Rock Ridge oder Joliet Format aus installieren. In einigen Fällen musst du das Dateisystem manuell einbinden, bevor du es verwenden kannst.
- **NFS:** Manche Plattformen unterstützen NFS mounts für die Datei Sets.
- **Band:** Datei Sets können ebenfalls von einem unterstützten Band gelesen werden.

4.2 - Checkliste für die Installation

Bevor du mit der eigentlichen Installation beginnst, solltest du dir im Klaren sein, was du eigentlich am Ende haben willst. Mindestens die folgenden Punkte sollten daher vorher geklärt sein:

- Name der Maschine
- Die installierte Hardware
  - Prüfe die Kompatibilität anhand der passenden Seite für deine Hardware.
  - Wenn du ISA-Komponenten verwendest, musst du die Einstellungen kennen und prüfen, ob sie den Anforderungen von OpenBSD entsprechen.
- Die gewünschte Installationsmethode. (CDROM, FTP, etc.)
- Wie soll das System aktualisiert und gepatcht werden?
  - Wenn es lokal ausgeführt werden soll, musst du genügend freien Speicher für den Source Tree und dessen Erzeugung haben.
  - Ansonsten musst du Zugriff auf eine andere Maschine haben, das ein gepatchtes release zur Verfügung stellt.
- Das erwünschte Plattenlayout
  - Müssen vorhandene Daten irgendwo gesichert werden?
  - Soll OpenBSD neben einem anderen Betriebssystem auf dem Rechner existieren? Wenn ja, auf welche Weise sollen die beiden Systeme gebootet werden? Brauchst du vielleicht einen "Boot Manager"?
  - Wird OpenBSD die ganze Festplatte belegen oder möchtest du eine existierende Partition/ein existierendes OS behalten (oder Platz für zukünftige Dinge)?
  - Wie willst du die OpenBSD Partition selbst aufteilen?
- Netzwerk-Einstellungen, falls du nicht DHCP verwendest:
  - Domain Name
  - Domain Name Server(s) (DNS) Adresse
  - IP Adressen und Subnetz-Masken für jede Netzwerk-Karte.
  - Gateway Adressen
- Wirst du auf diesem System X benutzen?

4.3 - Bootfähige OpenBSD Installationsmedien erzeugen

Als Beispiel werden wir die Installationsmedien betrachten, die für die i386 und sparc Plattformen bereitstehen.

Die i386 Plattform besitzt sechs separate Installations Disk Images, die ausgewählt werden können:

- **`floppy36.fs`** (Desktop PC) unterstützt viele PCI und ISA Netzwerkkarten (NICs), IDE und einfache SCSI Adapter und bietet wenig Unterstützung für PCMCIA. Die meisten Benutzer werden dieses Image verwenden, wenn sie von einer Floppy aus booten.
- **`floppyB36.fs`** (Server) unterstützt viele RAID Controller und einige der selteneren SCSI Adapter. Viele der Standard SCSI Adapter und viele EISA und ISA NICs werden nicht unterstützt.
- **`floppyC36.fs`** (Laptops) unterstützt viele der Cardbus und PCMCIA Geräte, die man in Laptops findet.
- **`cdrom36.fs`** ist eine Kombination aller drei anderen Boot-Disketten. Kann dazu verwendet werden, eine bootfähige 2.88M Floppy zu erzeugen.
- **`cd36.iso`** ist ein ISO9660 Image, das zum Erstellen einer bootfähigen CD mit den bekanntesten CD-ROM Brennsoftware Produkten auf den meisten Plattformen verwendet werden kann. Dieses Image hat die größte Auswahl an Treibern und ist normalerweise die empfohlene Wahl, wenn deine Hardware von CDROM booten kann.
- **`cdemu36.iso`** ist ein ISO9660 Image, das zum Booten eine "floppy emulation" benutzt und zwar das 2,88M Image, `cdrom36.fs`. Hoffentlich brauchen nur wenige Leute dieses Image -- die meisten Leute werden `cd36.iso` benutzen, verwende `cdemu36.iso` nur, wenn `cd36.iso` bei dir nicht funktioniert.

Ja, es kann Situationen geben, in denen du eine Floppy für deinen SCSI-Controller brauchst und eine andere für deine Netzwerk-Karte. Zum Glück geschieht das selten und kann normalerweise vermieden werden.

Die sparc Plattform besitzt drei separate Installations Disk Images, die ausgewählt werden können.

- **`floppy36.fs:`** Unterstützt Systeme mit Diskettenlaufwerk.
- **`cd36.iso`** Ein ISO Image, das zum Erstellen einer CD verwendet werden kann, um SPARC Systeme von einer CD-ROM aus booten zu können.
- **`miniroot36.fs`** Kann auf eine Swap Partition geschrieben und gebootet werden.

4.3.1 - Floppies unter Unix erzeugen

Um eine formatierte Floppy zu erzeugen, benutze einfach das fdformat(1) Kommando, damit kannst du deine Floppy sowohl formatieren als auch auf defekte Sektoren prüfen.

```
# fdformat /dev/rfd0c
Format 1440K floppy `/dev/rfd0c'? (y/n): y
Processing VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV done.
```

Ist deine Ausgabe wie im obigen Beispiel, dann ist deine Floppy OK. Wenn du, warum auch immmer, nicht ALLE "V"s siehst, dann ist deine Floppy höchstwahrscheinlich fehlerhaft und du solltest eine neue verwenden.

Bedenke, dass einige Unix-ähnliche Systeme verschiedene Befehle zum Formatieren von Disketten haben. Konsultiere das Handbuch deines Systems für den exakten Ablauf.

Hast du dann eine korrekt formatierte Floppy, ist es an der Zeit, das Installationsimage auf die Floppy zu schreiben. Wenn du sie auf einer OpenBSD Maschine oder auf einem anderen UNIX-ähnlichen OS erstellst, kannst du dd(1) verwenden. Ein Beispiel für `dd(1)` findest du hier:

```
# dd if=floppy36.fs of=/dev/rfd0c bs=32k
```

Ist das Image geschrieben, überprüfe es mit dem cmp(1) Befehl, um sicherzugehen, dass die Kopie dem Original entspricht. Wenn Diskette und Image identisch sind, wirst du einfach einen weiteren Prompt sehen.

```
# cmp /dev/rfd0c floppy36.fs
```

4.3.2 - Floppies unter Windows oder DOS erzeugen

Dieses Kapitel beschreibt, wie man Installationsimages unter Windows oder DOS auf eine Diskette schreibt. Du kannst die Anwendungen, die unten erwähnt werden, aus dem tools Verzeichnis von einem der FTP mirror oder aus dem `3.6/tools` Verzeichnis auf CD1 des OpenBSD CD Sets bekommen.

Zur Vorbereitung der Floppy unter MS-DOS oder Windows benutzt man einfach die bordeigenen Mittel zum Formatieren.

Das Installations-Image wird mittels *rawrite*, *fdimage* oder *ntrw* auf die formatierte Floppy geschrieben. *rawrite* funktioniert nicht unter Windows NT, 2000 oder XP.

Bedenke, dass `FDIMAGE.EXE` und `RAWRITE.EXE` beides MS-DOS Applikationen sind und daher auch den "8.3" Datei-Namen-Beschränkungen von MS-DOS unterliegen. Da sowohl `floppy32B.fs` als auch `floppy32C.fs` längere Dateinamen haben, wirst du überprüfen müssen, wie dein System die Dateien an sein "8.3"-Format angepasst hat, bevor du *FDIMAGE.EXE* oder *RAWRITE.EXE* benutzen kannst, um deine Boot-Floppy zu schreiben.

Ein Beispiel für *rawrite*:

```
C:\> rawrite
RaWrite 1.2 - Write disk file to raw floppy diskette

Enter source file name: floppy36.fs
Enter destination drive: a
Please insert a formatted diskette into drive A: and press -ENTER- : Enter
```

Eine Beispielanwendung von *fdimage*:

```
C:\> fdimage -q floppy36.fs a:
```

Eine Beispielanwendung von *ntrw*:

```
C:\> ntrw floppy36.fs a:
3.5", 1.44MB, 512 bytes/sector
bufsize is 9216
1474560 bytes written
```

4.3.3 - Eine Boot-CD erzeugen

Um eine CD-ROM zu erstellen, kannst du die `cd36.iso` Datei verwenden, oder, im Falle der i386 Plattform, kannst du auch `cdrom36.fs` als bootfähiges Floppy Image nutzen, das zum Booten des i386 Systems von CD-ROM aus dient. Das Herausfinden der exakten Details mit den zur Verfügung stehenden Anwendungen sei an dieser Stelle dem Leser überlassen.

Einige der Anwendungen unter OpenBSD sind:

- mkhybrid(8)
- cdrecord, Teil der cdrtools Kollektion im OpenBSD Ports System.

4.4 - Das Booten der OpenBSD Installations-Images

i386 booten

Das Installationsimage auf der i386 Architektur zu booten ist nichts Neues für die meisten Leute. Wenn du die Floppy verwendest, dann lege sie einfach in dein Diskettenlaufwerk und starte dein System. Das Installationsimage wird automatisch laden, wenn in deinem BIOS das Booten von Diskette aktiviert ist. Wenn du von CD starten willst, dann musst du im System BIOS das Booten von CD-ROM erlauben. Einige ältere BIOSe haben diese Option nicht und du musst eine Floppydisk zum Starten des Installationsimages verwenden. Keine Sorge, du kannst auch dann von CD installieren, wenn von Diskette gebootet wurde.

Du kannst ebenfalls installieren, indem du bsd.rd von einer existierenden OpenBSD Partition oder vom Netzwerk unter Verwendung vom PXE Boot Prozess aus bootest.

sparc/sparc64 booten

HINWEIS: Auf der sparc64 Plattform können nur die SBus Maschinen (Ultra 1, Ultra 2) von einer Floppy aus gebootet werden.

Um von Diskette zu starten, lege die OpenBSD Installationsdiskette in das Diskettenlaufwerk ein. Dann verwende folgendes Kommando, um von der Floppy zu booten:

```
ok boot floppy
```

Um von CD-ROM zu starten, lege die OpenBSD CD-ROM in dein Laufwerk. Wenn deine Sun nur ein CD-ROM Laufwerk hat, dann gehe an den Boot Prompt, an dem du ‚`boot cdrom`' ausführen kannst:

```
ok boot cdrom
```

Natürlich wird dies nur im neuen Befehlsmodus funktionieren. Wenn du am alten Befehlsmodus bist (ein rechter Pfeil), gib ‚n' ein, um in den neuen zu gelangen (Wenn du eine alte sparc vor sun4c hast, dann hast du vielleicht keinen neuen Befehlsmodus: Hier hilft dir nur experimentieren). Wenn du mehrere CD-ROM Geräte hast, dann musst du vom richtigen starten. Verwende `probe-scsi` im neuen Befehlsmodus.

```
ok probe-scsi

Target 0
  Unit 0   Disk     QUANTUM LIGHTNING 365S
Target 1
  Unit 0   Removable Disk     QUANTUM EMPIRE_1080S
Target 3
  Unit 0   Removable Disk     Joe's CD ROM
```

Suche das richtige CD-ROM und merke dir die ‚target' Nummer.

```
ok boot /sbus/esp/sd@X,0
```

4.5 - Eine Installation durchführen

4.5.1 - Mit der Installation beginnen

Was auch immer du für eine Methode zum Booten gewählt hast, nun ist es an der Zeit sie zu benutzen. Während des Boot-Prozesses werden der Kernel und alle Programme zum Installieren von OpenBSD in den Hauptspeicher geladen. Die größten Probleme entstehen beim Booten von defekten Floppies oder bei der Zuweisung von Festplatten. Die Boot-Floppy ist ziemlich vollgepackt - jeder defekte Block wird Probleme verursachen.

An fast jeder Stelle während des OpenBSD Installationsprozesses kannst du den aktuellen Installationsversuch abbrechen, indem du STRG-C drückst, und kannst ihn ohne Neustart durch Aufruf von `install` im Shell Prompt wieder beginnen.

Wenn das Booten erfolgreich war, wirst du eine Menge von Text-Meldungen vorbeiziehen sehen. Dieser Text, bei vielen Plattformen in weißer Schrift auf blauem Hintergrund, ist die sogenannte dmesg, der Kernel erzählt dir, welche Geräte gefunden wurden und wo. Mach dir nicht die Mühe, den Text zu behalten, eine Kopie davon wird in `/var/run/dmesg.boot` gespeichert. Auf den meisten Architekturen kannst du mittels SHIFT+BILD AUF den Text wieder zurückholen, der bereits vorbeigelaufen ist.

Dann wirst du das folgende sehen:

```
rootdev=0x1100 rrootdev=0x2f00 rawdev=0x2f02
erase ^?, werase ^W, kill ^U, intr ^C, status ^T
(I)nstall, (U)pgrade or (S)hell? i
```

Und damit erreichen wir auch schon die erste Frage. Meistens werden die drei Optionen angeboten:

- **Install:** OpenBSD auf dein System installieren und dabei alles überschreiben, was dort vorher war. Natürlich kann man hier auch einige Teile der Festplatte unberührt lassen, wie zum Beispiel `/home`, aber ansonsten wird alles andere überschrieben.
- **Upgrade:** Installiere neue Sets von Installations-Dateien auf dieser Maschine, aber überschreibe keine Konfigurations-Dateien, Benutzerdaten oder zusätzliche Programme. Es wird keine Festplatten-Formatierung vorgenommen und auch die Verzeichnisse /etc und /var werden nicht überschrieben. Ein paar wichtige Hinweise:
  - Dir wird nicht die Möglichkeit angeboten, `etc36.tgz` zu installieren. Nach der Installation musst du die Änderungen von `etc36.tgz` selbst in dein System einbinden, bevor du erwarten kannst, dass es voll funktionsfähig sein wird. Das zu tun ist dann sehr wichtig, da sonst einige Dienste (wie z.B. pf(4)) möglicherweise nicht starten werden.
  - Der Upgrade-Prozess ist nicht dazu da, um ganze Versionen zu überspringen! Obwohl das meist funktioniert, wird es in keinem Fall unterstützt. Bei OpenBSD 3.6 wird nur das Upgrade von 3.5 her unterstützt. Bei allen älteren Versionen sollte man unbedingt neu installieren.
- **Shell:** Manchmal muss man Reparaturen oder Schritte unternehmen, da das System sonst nicht mit einem normalen Kernel booten wird. Dazu ist diese Option gedacht.

In einigen Fällen wirst du die "Upgrade" Option nicht vorfinden. Nach einem ‚*flag day*' ist es zum Beispiel nicht möglich, ein direktes Upgrade zu machen, daher muss man das System von Grund auf neuinstallieren.

In diesem Beispiel machen wir eine Installation, der Upgrade-Prozess ist aber recht ähnlich.

```
Welcome to the OpenBSD/i386 3.6 install program.

This program will help you install OpenBSD in a simple and rational way. At
any prompt except password prompts you can run a shell command by typing
'!foo', or escape to a shell by typing '!'. Default answers are shown in []'s
and are selected by pressing RETURN. At any time you can exit this program by
pressing Control-C and then RETURN, but quitting during an install can leave
your system in an inconsistent state.

Specify terminal type: [vt220] Enter
Do you wish to select a keyboard encoding table? [no] Enter
```

In den meisten Fällen ist der vorgeschlagene Terminal-Typ passend, solltest du allerdings eine serielle Konsole zur Installation benutzen, wähle bitte die passende Konsole aus, nicht die vorgeschlagene.

Wenn du keine Tastaturbelegung (keyboard encoding table) auswählst, wird das US Tastatur-Layout benutzt.

```
IS YOUR DATA BACKED UP? As with anything that modifies disk contents, this
program can cause SIGNIFICANT data loss.

It is often helpful to have the installation notes handy. For complex disk
configurations, relevant disk hardware manuals and a calculator are useful.

Proceed with install? [no] y
```

Wenn du hier die vorgeschlagene Antwort gibst, landest du in einer Shell und bekommst einen Prompt, nachdem die Installation sich beendet hat.

4.5.2 - Festplatte(n) einrichten

Die Laufwerke in OpenBSD einzurichten ist für alle Plattformen ein wenig unterschiedlich. Bei i386 und macppc geschieht das Einrichten in zwei Etappen. Die eine geschieht mit fdisk(8) und die andere mit disklabel(8).

Der ein oder andere wird ein wenig verwundert über die hier verwendete Terminologie sein. Es sieht so aus, als ob wir das Wort "Partition" in zwei verschiedenen Weisen benutzen. Diese Beobachtung ist richtig. Es gibt zwei Schichten von Partitionierung auf verschiedenen OpenBSD-Plattformen, die erste könnte man als Betriebssystem-Partitionierung bezeichnen, so etwa legen viele Betriebssysteme ihre Partitionen an, und die zweite bezieht sich darauf, wie OpenBSD seine Partition in weitere individuelle Dateisysteme einteilt. Die erste Schicht ist sichtbar als eine Partition für DOS, Windows und jedes weitere Betriebssystem auf den IBM-kompatiblen Maschinen. Die zweite Schicht dagegen ist nur für OpenBSD und solche Systeme sichtbar, die OpenBSD-Dateisysteme direkt lesen können.

```
Cool!  Let's get to it...

You will now initialize the disk(s) that OpenBSD will use. To enable all
available security features you should configure the disk(s) to allow the
creation of separate filesystems for /, /tmp, /var, /usr, and /home.

Available disks are: wd0.
Which one is the root disk? (or done) [wd0] Enter
```

Die ‚root disk' ist die Festplatte, von der das System booten soll und wo sich normalerweise der Swap-Space befindet. IDE Fesplatten werden als `wd0`, `wd1`, etc. angezeigt, SCSI Festplatten und RAIDs als `sd0`, `sd1` und so weiter. Alle Festplatten, die OpenBSD finden kann, sind hier aufgelistet -- wenn du welche hast, die hier nicht auftauchen, sind sie vermutlich falsch konfiguriert oder werden gar nicht unterstützt.

```
Do you want to use *all* of wd0 for OpenBSD? [no] Enter
```

Wenn du diese Frage mit "yes" beantwortest, wird die gesamte Festplatte für OpenBSD benutzt. Das resultiert in einem standardmäßigen Master Boot Record und einer Partitionstabelle, die beide auf die Festplatte geschrieben werden - eine Partition, die die ganze Festplatte einnimmt, den OpenBSD Partitionstyp hat und als bootbar gekennzeichnet ist. Das ist für die meisten Produktionssysteme unter OpenBSD die brauchbarste Lösung, auf manchen Systemen sollte man das allerdings nicht so machen. Viele Compaq Systeme, viele Laptops, einige Dell und ein paar andere Systeme benutzen eine "maintenance" Partition, die unbedingt intakt bleiben muss. Wenn dein System einige Partitionen hat, die du nicht löschen willst, solltest du auf keinen Fall mit "yes" antworten.

Für dieses Beispiel nehmen wir an, dass die Festplatte zwischen OpenBSD und einer bereits existierenden Installation von Windows 2000 aufgeteilt werden soll, also beantworten wir die Frage hier mit "no", was uns in das fdisk(8) Programm bringt. Mehr Informationen über fdisk findest du hier.

Wichtiger Hinweis: Benutzer mit einer großen Festplatte (größer als 8GB auf neueren i386, auf älteren Maschinen und anderen Plattformen oftmals noch kleiner) sollten unbedingt diese Sektion lesen, bevor sie weitermachen.

```
You will now create a single MBR partition to contain your OpenBSD data. This
partition must have an id of 'A6'; must *NOT* overlap other partitions; and
must be marked as the only active partition.

The 'manual' command describes all the fdisk commands in detail.

Disk: wd0       geometry: 2586/240/63 [39100320 Sectors]
Offset: 0       Signature: 0xAA55
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
*0: 06    0   1  1 -  202 239 63 [          63:     3069297 ] DOS > 32MB
 1: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
 2: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
 3: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
Enter 'help' for information
fdisk: 1> help
        help            Command help list
        manual          Show entire OpenBSD man page for fdisk
        reinit          Re-initialize loaded MBR (to defaults)
        setpid          Set the identifier of a given table entry
        disk            Edit current drive stats
        edit            Edit given table entry
        flag            Flag given table entry as bootable
        update          Update machine code in loaded MBR
        select          Select extended partition table entry MBR
        print           Print loaded MBR partition table
        write           Write loaded MBR to disk
        exit            Exit edit of current MBR, without saving changes
        quit            Quit edit of current MBR, saving current changes
        abort           Abort program without saving current changes
fdisk: 1>
```

Ein paar Befehle sollten wir uns näher ansehen:

- **r** oder **reinit:** Löscht eine vorhandene Partitionstabelle und erzeugt eine große OpenBSD Partition, diese wird als aktiv markiert, und installiert den OpenBSD MBR Code. Ist das gleiche, als ob man mit "yes" auf die "`use *all* of ...`" Frage antwortet.
- **p** oder **print:** Zeigt die aktuelle Partitionstabelle in Sektoren an. "`p m`" dagegen zeigt die Tabelle in MegaBytes an, "`p g`" in GigaBytes.
- **e** oder **edit:** Editieren oder Ändern eines Tabelleneintrags.
- **f** oder **flag:** Markiert eine Partition als die aktive Partition, diejenige, von der gebootet wird.
- **u** oder **update:** Aktualisiert den MBR mit dem OpenBSD Bootcode, ähnlich wie "`reinit`", nur dass es die bestehende Partitionstabelle nicht verändert.
- **exit** und **quit:** Sei hiermit vorsichtig, da einige Anwender nicht daran gewöhnt sind, dass "exit" und "quit" verschiedene Bedeutungen haben.

Es ist nochmals wichtig darauf hinzuweisen, dass ein Fehler hier durchaus im Verlust von Daten resultieren kann. Wenn du das hier auf einer Platte mit wertvollen Daten machst, solltest du vielleicht vorher auf einer "Test"-Platte proben, zumal du dann noch ein gutes Backup hast.

Unsere Festplatte hier hat eine 1,5G Partition für Windows 2000 (mit dem FAT Dateisystem). Die Info oben zeigt, dass die Windows-Partition bis zum Zylinder 202 reicht. Also werden wir den Rest, sprich ab Zylinder 203, für OpenBSD reservieren. Du kannst auch den Start-Sektor von OpenBSD berechnen (3069360), indem du den Startsektor (63) und die Größe (3069297) zu der bereits vorhandenen Partition (hier Windows 2000 mit FAT) addierst.

Du kannst das Festplatten-Layout entweder in Form von Cylinder/Heads/Sectors editieren, oder auch in Form reiner Sektoren. Was einfacher ist, hängt davon ab, was du tun willst, in diesem Fall ist es wahrscheinlich einfacher das CHS Format zu benutzen. Wenn du die erste Partition auf der Festplatte erstellst, ist das Verwenden von reinen Sektoren wahrscheinlich einfacher.

```
fdisk: 1> e 1
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
 1: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
Partition id ('0' to disable)  [0 - FF]: [0] (? for help) a6
Do you wish to edit in CHS mode? [n] y
BIOS Starting cylinder [0 - 2585]: [0] 203
BIOS Starting head [0 - 239]: [0] Enter
BIOS Starting sector [1 - 63]: [0] 1
BIOS Ending cylinder [0 - 2585]: [0] 2585
BIOS Ending head [0 - 239]: [0] 239
BIOS Ending sector [1 - 63]: [0] 63
fdisk:*1> p
Disk: wd0       geometry: 2586/240/63 [39100320 Sectors]
Offset: 0       Signature: 0xAA55
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
*0: 06    0   1  1 -  202 239 63 [          63:     3069297 ] DOS > 32MB
 1: A6  203   0  1 - 2585 239 63 [     3069360:    36030960 ] OpenBSD
 2: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
 3: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
fdisk:*1> p m
Disk: wd0       geometry: 2586/240/63 [19092 Megabytes]
Offset: 0       Signature: 0xAA55
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
*0: 06    0   1  1 -  202 239 63 [          63:        1499M] DOS > 32MB
 1: A6  203   0  1 - 2585 239 63 [     3069360:       17593M] OpenBSD
 2: 00    0   0  0 -    0   0  0 [           0:           0M] unused
 3: 00    0   0  0 -    0   0  0 [           0:           0M] unused
fdisk:*1>
```

Es ist wichtig, dass die erste Partition den ersten Track der Platte auslässt, in diesem Fall also mit Sektor `63` beginnt. Wenn eine OpenBSD Partition so erstellt wurde, dass sie mit einen Offset von `0` beginnt, wird diese Partitionstabelle am Ende von dem OpenBSD Partitions Partition Boot Record überschrieben. Das System könnte weiterhin bootfähig sein, aber es wird sehr schwer verwaltbar sein und diese Konfiguration ist *weder empfohlen noch unterstützt*.

Wie du sehen kannst, wurde der Prompt um einen Asterisk (,*') erweitert, um anzuzeigen, dass du noch nicht gespeicherte Änderungen gemacht hast. Wie wir an der Ausgabe von `p m` sehen können, haben wir unsere Windows Partition nicht geändert, wir haben den Rest der Platte erfolgreich an OpenBSD vergeben. Wir sind fertig. Fast.

Was wir noch nicht gemacht haben, ist die Partition als aktiv zu markieren, so dass nach dem nächsten Reboot OpenBSD geladen wird:

```
fdisk:*1> f 1
Partition 1 marked active.
fdisk:*1> p
Disk: wd0       geometry: 2586/240/63 [39100320 Sectors]
Offset: 0       Signature: 0xAA55
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
 0: 06    0   1  1 -  202 239 63 [          63:     3069297 ] DOS > 32MB
*1: A6  203   0  1 - 2585 239 63 [     3069360:    36030960 ] OpenBSD
 2: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
 3: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
fdisk:*1>
```

Und nun sind wir soweit, dass wir unsere Änderungen speichern können:

```
fdisk:*1> w
Writing MBR at offset 0.
wd0: no disk label
fdisk: 1> q
```

Ein Disklabel erzeugen

Im nächsten Schritt benutzen wir disklabel(8), um die OpenBSD Partition aufzuteilen. Mehr Details über Disklabel finden sich in der FAQ 14, Disklabel.

```
Here is the partition information you chose:

Disk: wd0       geometry: 2586/240/63 [39100320 Sectors]
Offset: 0       Signature: 0xAA55
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
 0: 06    0   1  1 -  202 239 63 [          63:     3069297 ] DOS > 32MB
*1: A6  203   0  1 - 2585 239 63 [     3069360:    36030960 ] OpenBSD
 2: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
 3: 00    0   0  0 -    0   0  0 [           0:           0 ] unused

You will now create an OpenBSD disklabel inside the OpenBSD MBR
partition. The disklabel defines how OpenBSD splits up the MBR partition
into OpenBSD partitions in which filesystems and swap space are created.

The offsets used in the disklabel are ABSOLUTE, i.e. relative to the
start of the disk, NOT the start of the OpenBSD MBR partition.

disklabel: no disk label
WARNING: Disk wd0 has no label. You will be creating a new one.

# using MBR partition 1: type A6 off 3069360 (0x2ed5b0) size 36030960 (0x225c9f0)

Treating sectors 3069360-39100320 as the OpenBSD portion of the disk.
You can use the 'b' command to change this.

Initial label editor (enter '?' for help at any prompt)
> ?
Available commands:
        ? [cmnd]  - this message or command specific help.
        a [part]  - add new partition.
        b         - set OpenBSD disk boundaries.
        c [part]  - change partition size.
        D         - set label to default.
        d [part]  - delete partition.
        e         - edit drive parameters.
        g [b|d|u] - use [b]ios, [d]isk or [u]ser geometry.
        M         - show entire OpenBSD man page for disklabel.
        m [part]  - modify existing partition.
        n [part]  - set the mount point for a partition.
        p [unit]  - print label.
        q         - quit and save changes.
        r         - recalculate free space.
        s [path]  - save label to file.
        u         - undo last change.
        w         - write label to disk.
        X         - toggle expert mode.
        x         - exit without saving changes.
        z         - zero out partition table.
        ? [cmnd]  - this message or command specific help.
```

```
Numeric parameters may use suffixes to indicate units:
        'b' for bytes, 'c' for cylinders, 'k' for kilobytes, 'm' for megabytes,
        'g' for gigabytes or no suffix for sectors (usually 512 bytes).
        '%' for percent of total disk size, '&' for percent of free space.
        Non-sector units will be rounded to the nearest cylinder.
Entering '?' at most prompts will give you (simple) context sensitive help.
>
```

Wieder sollten wir uns einige Kommandos näher ansehen:

- **p** - zeigt (prints) den aktuellen Disklabel auf dem Schirm und du kannst `k, m` oder `g` für Kilobytes, Megabytes oder Gigabytes benutzen.
- **D** - Löscht jeglichen exisitierenden Disklabel, erzeugt einen neuen Disklabel, der nur die aktuelle OpenBSD Partition beinhaltet. Das kann nützlich sein, wenn die Festplatte vorher ein Disklabel hatte und die OpenBSD Partition mit einer anderen Größe erneut erzeugt wurde -- das alte Disklabel wird vielleicht nicht gelöscht und könnte Verwirrung stiften.
- **m** - Modifiziert einen existierenden Eintrag in einem Disklabel. Überschätze nicht, was das hier bedeutet. Obwohl es die Größe einer Disklabel-Partition ändern kann, wird das Dateisystem auf der Festplatte NICHT geändert. Diese Option zu benutzen und zu erwarten, man könne damit bereits existierende Partitionen in der Größe verändern, ist ein guter Weg, große Mengen an Daten zu verlieren.

Es ist wichtig, deine Festplatte gut aufzuteilen. Die Antwort auf die Frage "Wie soll ich mein System partitionieren?" ist "Genauso, wie du es benötigst". Das ist nunmal von Fall zu Fall verschieden. Es gibt keine universelle Antwort. Wenn du nicht sicher bist, wie du dein System partitionieren sollst, sieh dir diesen Abschnitt an.

In diesem System haben wir für OpenBSD über 17G verfügbar. Das ist eine Menge Platz, und wir werden wahrscheinlich nie den ganzen Platz brauchen. Wir werden also mit der Platzverteilung etwas großzügig sein. Es ist besser, ein paar hundert Megabytes zuviel und unbenutzt zu haben, als auch nur ein Kilobyte zu wenig.

Auf der ‚root disk' **müssen** mindestens die Partitionen ‚a' und ‚b' erzeugt werden. Die Installation wird ohne diese beiden Partitionen abgebrochen. ‚a' wird für das root Dateisystem (/) und ‚b' für den Swapbereich benutzt.

Nach einer kleinen Denkpause haben wir uns entschieden, die Menge an Partitionen zu erzeugen, die wir für die empfohlenen separaten Dateisysteme brauchen (`/, /tmp, /var, /usr, /home`), zusätzlich zu einer Swap-Partition:

- ***wd0a:*** */* (root) - 150M. Sollte mehr als genug sein.
- ***wd0b:*** (swap) - 300M.
- ***wd0d: /tmp*** - 120M. `/tmp` wird für das Erzeugen von Software gebraucht, 120M wird für die meisten Dinge genug sein.
- ***wd0e: /var*** - 80M. Wenn das ein Web- oder Mail-Server wäre, müsste diese Partition viel größer sein, aber das machen wir ja hier nicht.
- ***wd0g: /usr*** - 2G. Wir brauchen den Platz hier für Menge User-Anwendungen, und außerdem wollen wir das System updaten und neu erzeugen können, wenn es notwendig wird. Der Ports Tree wird sich hier auch befinden, und vor dem Erzeugen von Ports bereits 100Megabyte Platz brauchen. Wenn du viele Applikationen aus den ports bauen willst, statt dich der fertig kompilierten packages zu bedienen, benötigst du hier eine Menge mehr Platz.
- ***wd0h: /home*** - 4G. Das lässt jede Menge Platz für Benutzer-Dateien.

Wenn du das alles zusammenzählst, wirst du bemerken, dass über 10G Plattenplatz unbenutzt sind! Unbenutzter Platz tut uns nicht weh und gibt uns die Flexibilität, die Dinge in Zukunft zu vergrößern. Du brauchst mehr */tmp*? Kein Problem, erzeuge welchen im unbenutzten Platz, ändere die Datei */etc/fstab* und das Problem ist gelöst.

```
> p m
device: /dev/rwd0c
type: ESDI
disk: ESDI/IDE disk
label: ST320011A
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
cylinders: 16383
total sectors: 39102336
free sectors: 36030960
rpm: 3600
```

```
16 partitions:
#        size   offset    fstype   [fsize bsize   cpg]
  a: 17593.2M  1498.7M    unused        0     0
  c: 19092.9M     0.0M    unused        0     0
  i:  1498.7M     0.0M     MSDOS
> d a
> a a
offset: [3069360] Enter
size: [36030960] 150M
Rounding to nearest cylinder: 307440
FS type: [4.2BSD] Enter
mount point: [none] /
> a b
offset: [3376800] Enter
size: [35723520] 300M
Rounding to nearest cylinder: 614880
FS type: [swap] Enter
> a d
offset: [3991680] Enter
size: [35108640] 120m
Rounding to nearest cylinder: 245952
FS type: [4.2BSD] Enter
mount point: [none] /tmp
> a e
offset: [4237632] Enter
size: [34862688] 80m
Rounding to nearest cylinder: 164304
FS type: [4.2BSD] Enter
mount point: [none] /var
> a g
offset: [4401936] Enter
size: [34698384] 2g
Rounding to nearest cylinder: 4194288
FS type: [4.2BSD] Enter
mount point: [none] /usr
> a h
offset: [8596224] Enter
size: [30504096] 4g
Rounding to nearest cylinder: 8388576
FS type: [4.2BSD] Enter
mount point: [none] /home
> p m
device: /dev/rwd0c
type: ESDI
disk: ESDI/IDE disk
label: ST320011A
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
cylinders: 16383
total sectors: 39102336
free sectors: 22115520
rpm: 3600

16 partitions:
#        size   offset    fstype   [fsize bsize   cpg]
  a:   150.1M  1498.7M    4.2BSD     1024  8192    16  # /
  b:   300.2M  1648.8M      swap
  c: 19092.9M     0.0M    unused        0     0
  d:   120.1M  1949.1M    4.2BSD     1024  8192    16  # /tmp
  e:    80.2M  2069.2M    4.2BSD     1024  8192    16  # /var
```

```
  g:  2048.0M  2149.4M    4.2BSD     1024  8192    16  # /usr
  h:  4096.0M  4197.4M    4.2BSD     1024  8192    16  # /home
  i:  1498.7M     0.0M     MSDOS
> q
Write new label?: [y] Enter
```

Du wirst bemerken, dass wir die *c* Partition scheinbar komplett ignoriert haben. Diese Partition ist deine gesamte Festplatte, versuche sie nicht zu verändern. Du wirst außerdem bemerkt haben, dass wir die *i*-Partition nicht angelegt haben, denn das ist die bereits existierende Windows 2000 Partition. Partitionen werden hier keinem vorbestimmten Buchstaben zugewiesen, abgesehen von *a* (root), *b* (swap) und *c* (gesamte Festplatte), der Rest der Partitionen kann verwendet werden, wie man will.

Wenn du dir Ausgaben von disklabel genau ansiehst, wirst du vermutlich feststellen, dass die Angabe des RPM falsch ist. Das ist historisch bedingt, die Plattengeschwindigkeit wird in keiner Weise vom System benutzt. Mach dir also keine Sorgen darüber.

Deine Mount Points konfigurieren und deine Dateisysteme formatieren

Jetzt kommt die finale Konfiguration deiner Mount Points. Wenn du deine Mount Points mittels disklabel(8) konfiguriert hast, reicht es in diesem Schritt aus, deine Auswahl zu bestätigen, und wenn nicht, kannst du sie nun angeben.

```
The root filesystem will be mounted on wd0a.
wd0b will be used for swap space.
Mount point for wd0d (size=122976k), none or done? [/tmp] Enter
Mount point for wd0e (size=82152k), none or done? [/var] Enter
Mount point for wd0g (size=2097144k), none or done? [/usr] Enter
Mount point for wd0h (size=4194288k), none or done? [/home] Enter
Mount point for wd0d (size=122976k), none or done? [/tmp] done
Done - no available disks found.

You have configured the following partitions and mount points:

wd0a /
wd0d /tmp
wd0e /var
wd0g /usr
wd0h /home

The next step creates a filesystem on each partition, ERASING existing data.
Are you really sure that you're ready to proceed? [no] y
/dev/rwd0a:     307440 sectors in 305 cylinders of 16 tracks, 63 sectors
        150.1MB in 20 cyl groups (16 c/g, 7.88MB/g, 1920 i/g)
/dev/rwd0d:     245952 sectors in 244 cylinders of 16 tracks, 63 sectors
        120.1MB in 16 cyl groups (16 c/g, 7.88MB/g, 1920 i/g)
/dev/rwd0e:     164304 sectors in 163 cylinders of 16 tracks, 63 sectors
        80.2MB in 11 cyl groups (16 c/g, 7.88MB/g, 1920 i/g)
/dev/rwd0g:     4194288 sectors in 4161 cylinders of 16 tracks, 63 sectors
        2048.0MB in 261 cyl groups (16 c/g, 7.88MB/g, 1920 i/g)
/dev/rwd0h:     8388576 sectors in 8322 cylinders of 16 tracks, 63 sectors
        4096.0MB in 521 cyl groups (16 c/g, 7.88MB/g, 1920 i/g)
/dev/wd0a on /mnt type ffs (rw, asynchronous, local, ctime=Thu Oct 10 21:
50:36 2004)
/dev/wd0h on /mnt/home type ffs (rw, asynchronous, local, nodev, nosuid,
 ctime=Thu Oct 10 21:50:36 2004)
/dev/wd0d on /mnt/tmp type ffs (rw, asynchronous, local, nodev, nosuid,
ctime=Thu Oct 10 21:50:36 2004)
/dev/wd0g on /mnt/usr type ffs (rw, asynchronous, local, nodev, ctime=Th
u Oct 10 21:50:36 2004)
/dev/wd0e on /mnt/var type ffs (rw, asynchronous, local, nodev, nosuid,
ctime=Th u Oct 10 21:50:36 2004)
```

Du fragst dich wahrscheinlich, wieso dich das Installationsprogramm schon wieder nach den Mount Points fragt. Das erlaubt dir einfach Fehler oder Probleme mit der bisherigen Auswahl auszuräumen. Der Installationsprozess wird zum Beispiel doppelte Mount Points einfach löschen. Im disklabel Programm kannst du solche Doppel-Eingaben machen, und daher muss das überprüft werden. Eine solche doppelte Nennung wird zu Partitionen ohne Mount Point führen, denen du dann neue

Mount Points zuweisen musst, wenn du den Platz darin benutzen willst.

Wie du sehen kannst, lautet die vorgegebene Antwort auf die Frage "Are you really sure that you are ready to proceed?" "no", du musst also ausdrücklich "yes" antworten, damit das Programm weitermacht und deine Partitionen formatiert. Wenn du das nicht tust, landest du wieder in der Shell und könntest die Installation erneut starten, indem du "install" eingibst oder einfach erneut mit deiner Bootdisk bootest.

An diesem Punkt werden nun alle deine Dateisysteme formatiert. Wie lange das dauert hängt von der Größe der Partitionen und der Geschwindigkeit deiner Festplatte ab.

4.5.3 - Den Hostnamen des Systems setzen

Du musst dem System nun seinen Hostnamen geben. Dieser Wert, neben dem DNS domain name (weiter unten angegeben), wird in der Datei `/etc/myname`, die beim normalen Booten des Systems ausgelesen wird, gespeichert. Wenn du den FQDN des Systems nicht setzt, wird der vorgeschlagene Wert ‚my.domain' benutzt.

Es ist wichtig diesen Namen jetzt zu setzen, da er beim ersten Booten des Systems nach der Installation dabei benutzt wird, die kryptographischen Schlüssel zu erzeugen. Dieses Erzeugen geschieht unabhängig davon, ob das Netzwerk konfiguriert ist oder nicht.

```
Enter system hostname (short form, e.g. 'foo'): puffy
```

4.5.4 - Das Netzwerk konfigurieren

Jetzt ist es an der Zeit, dein Netzwerk zu konfigurieren. Das muss unbedingt geschehen, wenn du eine FTP- oder NFS-basierte Installation durchführen willst, da die Daten dazu notwendig sind. Hier ein Beispieldurchlauf für die Netzwerk-Konfiguration.

```
Configure the network? [yes] Enter
Available interfaces are: fxp0.
Which one do you wish to initialize? (or 'done') [fxp0] Enter
Symbolic (host) name for fxp0? [puffy] Enter
The default media for fxp0 is
        media: Ethernet autoselect (100baseTX full-duplex)
Do you want to change the default media? [no] Enter
IP address for fxp0 (or 'dhcp')? 199.185.137.55
Netmask? [255.255.255.0] Enter
Done - no available interfaces found.
DNS domain name? (e.g. 'bar.com') [my.domain] example.com
DNS nameserver? (IP address or 'none') [none] 199.185.137.1
Use the nameserver now? [yes] Enter
Default route? (IP address, 'dhcp' or 'none') 199.185.137.128
add net default: gateway 199.185.137.128
Edit hosts with ed? [no] Enter
Do you want to do any manual network configuration? [no] Enter
```

Im obigen Beispiel nutzen wir eine statische IP Adresse. Wie angedeutet, kannst du aber auch "`dhcp`" auf den meisten Plattformen (nicht Alpha) auswählen, davon ausgehend, dass deine Netzwerkumgebung es unterstützt. Im Falle von DHCP werden die meisten dieser Daten von dem entfernten DHCP Server geholt; du wirst eine Gelegenheit bekommen, das zu bestätigen. Hier ist ein Beispiel des Netzwerkkonfigurationsteils der Installation, dieses Mal mit DHCP:

```
Configure the network? [yes] Enter
Available interfaces are: fxp0.
Which one do you wish to initialize? (or 'done') [fxp0] Enter
Symbolic (host) name for fxp0? [puffy] Enter
The default media for fxp0 is
        media: Ethernet autoselect (100baseTX full-duplex)
Do you want to change the default media? [no] Enter
IP address for fxp0 (or 'dhcp')? dhcp
Issuing hostname-associated DHCP request for fxp0.
Sending on   Socket/fallback/fallback-net
DHCPDISCOVER on fxp0 to 255.255.255.255 port 67 interval 1
DHCPOFFER from 199.185.137.128
```

```
DHCPREQUEST on fxp0 to 255.255.255.255 port 67
DHCPACK from 199.185.137.128
New Network Number: 199.185.137.0
New Broadcast Address: 199.185.137.255
bound to 199.185.137.55 -- renewal in 43200 seconds.
Done - no available interfaces found.
DNS domain name? (e.g. 'bar.com') [example.org] Enter
DNS nameserver? (IP address or 'none') [199.185.137.1] Enter
Use the nameserver now? [yes] Enter
Default route? (IP address, 'dhcp' or 'none') [199.185.137.128] Enter
Edit hosts with ed? [no] Enter
Do you want to do any manual network configuration? [no] Enter
```

HINWEIS: Nur **ein** Interface kann während der Installation einfach mit DHCP konfiguriert werden. Es wird Fehler geben, wenn du das mehrmals versuchst. Die anderen Interfaces musst du dann nach der Installation per Hand konfigurieren.

Nun setzen wir das Passwort für den root Account:

```
Password for root account? (will not echo) pAssWOrd
Password for root account? (again) pAssWOrd
```

Verwende ein sicheres Passwort für den root Account. Du wirst weitere Benutzerkonten anlegen, wenn das System gebootet wurde. Aus passwd(1):

```
The new password should be at least six characters long and not purely
alphabetic.  Its total length must be less than _PASSWORD_LEN (currently
128 characters).  A mixture of both lower and uppercase letters, numbers,
and meta-characters is encouraged.
```

4.5.5 - Das Installationsmedium auswählen

Nachdem dein Netzwerk eingerichtet ist, hast du die Chance deine Einstellungen per Hand zu justieren. Danach werden die erzeugten Dateisysteme gemountet und ein Passwort für den Benutzer root gesetzt. Nun sind deine lokalen Festplatten bereit für die Aufnahme der OpenBSD Datei Sets.

Danach bekommst du die Chance dein Installationsmedium auszuwählen. Die Optionen sind unten aufgelistet.

```
You will now specify the location and names of the install sets you want to
load. You will be able to repeat this step until all of your sets have been
successfully loaded. If you are not sure what sets to install, refer to the
installation notes for details on the contents of each.

Sets can be located on a (m)ounted filesystem; a (c)drom, (d)isk or (t)ape
device; or a (f)tp, (n)fs or (h)ttp server.
Where are the install sets? c
Available CD-ROMs are: cd0.
```

In diesem Beispiel installierst du von CD-ROM. Dazu wird eine Liste von Geräten erzeugt, die auf deiner Maschine als CD-ROM identifiziert wurden. Die meisten Leute haben aber nur eins. Wähle einfach das Gerät aus, von dem aus du OpenBSD installieren willst.

HINWEIS: Alle möglichen Quellen für Installations-Sets werden aufgelistet, aber es sind nicht unbedingt alle auf deinem System verfügbar. "`(n)fs`" z.B. wird zwar angezeigt, aber nicht alle Architekturen erlauben auch solche Installationen. Eine Fehlermeldung und eine weitere Chance ein anderes Medium zu wählen folgen, nachdem du eines ausgewählt hast, das nicht zur Verfügung steht.

```
Available CD-ROMs are: cd0.
Which one contains the install media? (or 'done') [cd0] Enter

Pathname to the sets? (or 'done') [3.6/i386] Enter
```

Hier wirst du gefragt, in welchem Verzeichnis sich die Datei Sets befinden, auf der offiziellen CD-ROM ist das `3.6/i386/`.

4.5.6 - Datei Sets auswählen

Nun ist es an der Zeit, auszuwählen, welche Datei Sets du haben willst. Eine Beschreibung dieser Pakete findest du im nächsten Abschnitt. Die Dateien, die das Installationsprogramm findet, werden auf dem Bildschirm angezeigt. Deine Aufgabe ist es, anzugeben, welche Dateien du haben willst. Standardmäßig sind bis auf die X-Datei Sets alle markiert, manche Leute wollen aber wirklich nur das absolute Minimum haben, mit dem OpenBSD noch funktioniert, also `base36.tar.gz`, `etc36.tar.gz` und `bsd`. Andere wiederum wollen alle Datei Sets haben. Das Beispiel unten zeigt eine komplette Installation mit allen Paketen.

```
The following sets are available. Enter a filename, 'all' to select
all the sets, or 'done'. You may de-select a set by prepending a '-'
to its name.

        [ ] bsd.rd
        [X] base36.tgz
        [X] etc36.tgz
        [X] misc36.tgz
        [X] comp36.tgz
        [X] man36.tgz
        [X] game36.tgz
        [ ] xbase36.tgz
        [ ] xshare36.tgz
        [ ] xfont36.tgz
        [ ] xserv36.tgz
        [X] bsd

File Name? (or 'done') [bsd.rd] all

The following sets are available. Enter a filename, 'all' to select
all the sets, or 'done'. You may de-select a set by prepending a '-'
to its name.

        [X] bsd.rd
        [X] base36.tgz
        [X] etc36.tgz
        [X] misc36.tgz
        [X] comp36.tgz
        [X] man36.tgz
        [X] game36.tgz
        [X] xbase36.tgz
        [X] xshare36.tgz
        [X] xfont36.tgz
        [X] xserv36.tgz
        [X] bsd
```

Du kannst hier alle möglichen Sachen machen -- `-x*` zum Beispiel würde alle X-Komponenten entfernen, falls du es dir nochmal anders überlegst. In diesem Fall wollen wir aber alle Sets. Wenn das System auch mit viel weniger Sets läuft, ist der empfohlene Weg, entweder die standardmäßigen Vorgaben oder alles zu installieren. Weitere Details zum Auswählen der Sets befinden sich hier.

Nachdem du deine gewünschten Datei Sets ausgewählt hast, wirst du noch einmal gefragt, ob du die gewählten Datei Sets wirklich entpacken willst, und dann werden sie installiert. Es wird ein Fortschrittsbalken angezeigt, der dir zeigt, wie lange das ganze noch etwa dauert. Die Dauer hängt natürlch stark von der Geschwindigkeit deines Systems ab, den Datei Sets, die du installieren willst, und der Geschwindigkeit des Installationsmediums. Somit liegt die Dauer irgendwo zwischen ein paar Minuten und ein paar Stunden, je nachdem.

```
File Name? (or 'done') [done] Enter
Ready to install sets? [yes] Enter

Getting bsd ...
100% |*************************************************|  5232 KB    00:08
Getting bsd.rd ...
100% |*************************************************|  4614 KB    00:02
Getting bsd.mp ...
100% |*************************************************|  5285 KB    00:03
Getting base36.tgz ...
100% |*************************************************| 31396 KB    00:22
Getting etc36.tgz ...
100% |*************************************************|  1655 KB    00:01
Getting misc36.tgz ...
100% |*************************************************|  2193 KB    00:01
Getting comp36.tgz ...
100% |*************************************************| 18232 KB    00:15
Getting man36.tgz ...
100% |*************************************************|  6792 KB    00:05
Getting game36.tgz ...
100% |*************************************************|  2536 KB    00:01
Getting xbase36.tgz ...
100% |*************************************************| 10121 KB    00:07
Getting xetc36.tgz ...
100% |*************************************************|   430 KB    00:00
Getting xshare36.tgz ...
100% |*************************************************|  1888 KB    00:02
Getting xfont36.tgz ...
100% |*************************************************| 31742 KB    00:22
Getting xserv36.tgz ...
100% |*************************************************| 15460 KB    00:11

Sets can be located on a (m)ounted filesystem; a (c)drom, (d)isk or (t)ape
device; or a (f)tp, (n)fs or (h)ttp server.
Where are the install sets? (or 'done') [done] Enter
```

An dieser Stelle hast du die Möglichkeit, weitere Dateien von anderen Quellen zu installieren (inklusive individuellen Datei Sets), wenn du das möchtest oder gib ‚done' ein, wenn du alle Datei Sets installiert hast, die du benötigst.

4.5.7 - Zum Ende kommen

Als nächstest werden dir einige Fragen bezüglich der Einstellungen deines installierten Systems gestellt. Die erste ist, ob sshd(8) beim Hochfahren gestartet werden soll. Üblicherweise möchtest du, dass sshd(8) läuft, aber gelegentlich vielleicht auch nicht. Wenn deine Verwendung keinen Gebrauch für sshd(8) hat, liegt ein kleiner theoretischer Sicherheitsvorteil vor, wenn es nicht läuft.

```
Started sshd(8) by default? [yes] y
```

Du wirst nun gefragt werden, ob du auf dem System X laufen lassen willst. Wenn du mit ‚Y' antwortest, wird die Datei `/etc/sysctl.conf` so modifiziert, dass die Zeile `machdep.allowaperture=1` oder `machdep.allowaperture=2` darinsteht, abhängig von deiner Plattform. Einige Plattformen werden diese Frage überhaupt nicht stellen.

```
Do you expect to run the X Window System? [yes] y
```

Als nächstes wirst du gefragt, ob du eine serielle Konsole an diesem Computer anstatt von einer Standard Tastatur und einem Monitor betreiben willst. Wenn du "`yes`" auswählst und ein paar weitere einfache Fragen beantwortest, werden `/etc/boot.conf` und `/etc/ttys` automatisch für dich angepasst. Die meisten User werden hier jedoch die Voreinstellung **no** verwenden.

```
Change the default console to com0? [no] Enter
```

Deine letzte Aufgabe ist es, deine entsprechende Zeitzone einzustellen. Abhängig davon, wo deine Maschine steht, kann es mehrere richtige Antworten geben. Im folgenden Beispiel benutzen wir `US/Eastern`, könnten aber auch `EST5EDT` oder `US/Michigan` benutzen und zum selben Ergebnis kommen. Wenn man `?` drückt, bekommt man eine Liste mit möglichen Auswahlen.

```
Saving configuration files......done.
Generating initial host.random file ......done.
What timezone are you in? ('?' for list) [US/Pacific] ?
Africa/      Chile/       GB-Eire      Israel       NZ-CHAT      Turkey
America/     Cuba         GMT          Jamaica      Navajo       UCT
Antarctica/  EET          GMT+0        Japan        PRC          US/
Arctic/      EST          GMT-0        Kwajalein    PST8PDT      UTC
Asia/        EST5EDT      GMT0         Libya        Pacific/     Universal
Atlantic/    Egypt        Greenwich    MET          Poland       W-SU
Australia/   Eire         HST          MST          Portugal     WET
Brazil/      Etc/         Hongkong     MST7MDT      ROC          Zulu
CET          Europe/      Iceland      Mexico/      ROK          posix/
CST6CDT      Factory      Indian/      Mideast/     Singapore    posixrules
Canada/      GB           Iran         NZ           SystemV/     right/
What timezone are you in? ('?' for list) [US/Pacific] US
What sub-timezone of 'US' are you in? ('?' for list) ?
Alaska          Central         Hawaii          Mountain        Samoa
Aleutian        East-Indiana    Indiana-Starke  Pacific
Arizona         Eastern         Michigan        Pacific-New
Select a sub-timezone of 'US' ('?' for list): Eastern
Setting local timezone to 'US/Eastern'...done
```

Wenn du auf besonders genaue Zeitangaben aus bist, möchtest du vielleicht das hier lesen.

Die letzten Schritte für das System sind das Erzeugen des `/dev` Verzeichnisses (was auf manchen Maschinen ganz schön lange dauert, insbesondere, wenn du nur wenig RAM besitzt) und das Installieren des Bootblocks.

```
Making all device nodes...done.
Installing boot block...
boot: /mnt/boot
proto: /usr/mdec/biosboot
device: /dev/rwd0c
/usr/mdec/biosboot: entry point 0
proto bootblock size 512
/mnt/boot is 3 blocks x 16384 bytes
fs block shift 2; part offset 3069360; inode block 152, offset 4136
using MBR partition 1: type 166 (0xa6) offset 3069360 (0x2ed5b0)
done.

CONGRATULATIONS! Your OpenBSD install has been successfully completed!
To boot the new system, enter halt at the command prompt. Once the
system has halted, reset the machine and boot from the disk.
# halt
syncing disks... done

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

OpenBSD ist jetzt auf deinem System installiert und bereit für den ersten Boot, aber vorher...

Bevor du neustartest

Nun ist dein System also installiert und fertig zum ersten Booten, um danach noch konfiguriert zu werden. Zuvor wäre es allerdings weise, die Errata Seite zu überprüfen, ob es nicht irgendwelche Bugs gibt, die dich sofort betreffen könnten.

Ein Trick um eine ‚vor dem ersten Start'-Konfiguration durchführen zu können ist:

```
# /mnt/usr/sbin/chroot /mnt
```

beim Shell-Prompt einzugeben. Dies setzt deine Mountpoints so, wie sie nach einem normalen Neustart deines frisch installierten Systems sein werden. Du kannst grundlegende Systemkonfiguration durchführen, wie das Hinzufügen von Benutzern, Mountpoints ändern, etc.

Nach dem Neustart

Eines der ersten Dinge, die du nach der Installation lesen solltest, ist afterboot(8).

Die folgenden Links könnten ebenfalls recht nützlich sein:

- Benutzer in OpenBSD hinzufügen
- Erste Netzwerk-Konfiguration
- Manual Seiten von nützlichen/häufigen Befehlen
- OpenBSD Manual Seiten im Web
- Das OpenBSD Ports und Packages System zum Installieren von Software, genauso hier und hier

Noch eine Sache...

Die OpenBSD Entwickler bitten dich darum, eine Kopie deiner dmesg einzuschicken. Das hilft den Entwicklern und schlussendlich auch den Anwendern.

4.6 - Welche Dateien werden zur Installation benötigt?

Die komplette OpenBSD Installation ist in eine Vielzahl einzelner *Datei Sets* aufgeteilt. Nicht jede Verwendung verlangt nach jedem Datei Set. Hier ist eine Übersicht:

- ***bsd*** - Dies ist der Kernel. **Zwingend notwendig**
- ***bsd.mp*** - Multi-Prozessor (SMP) Kernel (nur auf einigen Plattformen)
- ***bsd.rd*** - RAM Disk Kernel
- ***base36.tgz*** - Enthät das Basis-System von OpenBSD **Zwingend notwendig**
- ***etc36.tgz*** - Enthält alle Dateien in /etc **Zwingend notwendig**
- ***comp36.tgz*** - Enthält den Compiler und alle seine Anwendungen, ‚headers' und Bibliotheken. **Empfohlen**
- ***man36.tgz*** - Enthält alle Manual Seiten **Empfohlen**
- ***misc36.tgz*** - Enthält alle misc info und setup Dokumentationen
- ***game36.tgz*** - Enthält die Spiele für OpenBSD
- ***xbase36.tgz*** - Enthält die Basis-Installation für X11
- ***xfont36.tgz*** - Enthält den X11 Font-Server und Fonts
- ***xserv36.tgz*** - Enthält die X11 X Server
- ***xshare32.tgz*** - Enthält Manual Seiten, lokale Einstellungen, ‚includes', etc. für X

Die **etc36.tgz** und **xetc36.tgz** Sets werden nicht als Teil eines Upgrades installiert, sondern nur als Teil einer kompletten Installation, so dass jegliche Änderungen, die du vorgenommen hast, nicht verloren gehen. Du musst deine `/etc`, `/dev` und `/var` Verzeichnisse manuell aktualisieren.

4.7 - Wieviel Platz brauche ich für eine OpenBSD Installation?

Die folgenden Angaben sind die minimalen Platzbedürfnisse für eine Vollinstallation. Die Zahlen enthalten genug Platz, um dir ein typisches Heim-System zu ermöglichen, das mit dem Internet verbunden ist.

- Dies sind Minimum-Werte.
- Wenn du planst, eine Menge Software von dritter Seite zu installieren, wähle deine `/usr` Partition groß! Du solltest diese Werte **mindestens** verdreifachen!
- Für ein System, das eine Menge E-Mails oder Web-Seiten handhabt (die in `/var/mail` und `/var/www` gespeichert werden), solltest du deine `/var` Partition bedeutend vergrößern oder lege sie auf separate Partitionen.
- Für ein Mehrbenutzer-System, das eine Menge Logs erzeugt, solltest du deine `/var` Partition noch größer wählen (`/var/log`).
- Wenn du planst, den Kernel oder das ganze System aus dem Source neu zu erzeugen, sollte deine `/usr` Partition deutlich größer ausfallen, **mindestens** 2G größer als unten angegeben.

Behalte bitte im Auge, dass `/usr` und `/usr/X11R6` im Normalfall Teile des selben Dateisystems sind, nämlich `/usr`, und dass es keinen Vorteil bringt, sie in getrennte Dateisysteme zu verlegen.

SYSTEM	/	/usr	/var	/usr/X11R6
alpha	80M	250M	25M	140M
hp300	80M	250M	25M	140M
hppa	100M	200M	25M	120M
i386	60M	250M	25M	140M
mac68k	80M	250M	25M	100M
macppc	80M	250M	25M	140M
mvme68k	80M	250M	25M	100M
sparc	80M	250M	25M	120M
sparc64	80M	250M	25M	100M
vax	100M	200M	25M	180M

Zusätzlich wird empfohlen, dass man eine `/tmp` Partition benutzt. Die `/tmp` Partition wird beim Kompilieren von Ports und anderen Dingen benutzt, die Größe hängt also davon ab, was du damit machen willst. 50MB mag für viele Leute viel sein, aber einige große Applikationen brauchen 100MB und mehr `/tmp` Platz.

Wenn du im disklabel-Editor bist, kannst du dich entschließen, deinem System nur ein ‚a' (Haupt-Dateisystem) und ‚b' (Swap) zu geben. Das ‚a' Dateisystem, das du in disklabel als deine root-Partition eintragen willst, sollte die Summe aller 3 Haupt-Werte größer sein als (`/`, `/usr`, und `/var`) plus etwas Extra-Platz für `/tmp`. Die ‚b' Partition, die du aufsetzt, wird automatisch zur Swap-Partition -- wir empfehlen ein Minimum von 32MB, aber wenn du genug Platz hast, mach wenigstens 64MB daraus. Wenn du aber wirklich viel Plattenplatz hast, scheu dich nicht, dein Swap mit 256MB oder sogar 512MB einzurichten.

Der Swapbereich wird verwendet, um ‚core dumps' zu speichern, falls ein crash(8) auftritt. Falls dies eine Überlegung von dir ist, sollte dein Swapbereich ein wenig größer sein als der Speicherplatz des Hauptspeiches, den du jemals in deinem System verwenden wirst. Bedenke, dass beim Neustart savecore(8) versucht, den Inhalt der Swap Partition in eine Datei unter `/var/crash` zu speichern, also wieder einmal, wenn dies wichtig für dich ist, sollte deine `/var` Partition über genug *freien Speicher* verfügen, um diese ‚dump' Dateien halten zu können.

Es gibt mehrere Hauptgründe, anstelle von nur ein oder zwei Dateisystemen mehr separate Dateisysteme zu benutzen:

- **Sicherheit**: Du kannst einige Dateisysteme als ‚nosuid', ‚nodev', ‚noexec', ‚readonly', etc. markieren. Das wird jetzt vom Installationsprozess erledigt, wenn du die oben beschriebenen Partitionen benutzt.
- **Stabilität**: Ein Anwender oder amoklaufendes Programm kann ein Dateisystem mit Müll auffüllen, wenn sie darauf Schreibrechte haben. Deine kritischen Programme, dann natürlich auf einem anderen Dateisystem laufend, werden nicht unterbrochen.
- **Geschwindigkeit**: Ein Dateisystem, in das dauernd hineingeschrieben wird, kann schnell fragmentieren. (Glücklicherweise ist das ffs-Dateisystem, das von OpenBSD benutzt wird, nicht für sowas anfällig.)
- **Integrität**: Wenn ein Dateisystem aus irgendeinem Grund defekt ist, sind deine anderen Dateisysteme immer noch OK.
- **Größe**: Viele Maschinen haben Grenzen in Bezug auf den Ort, von dem aus das ROM den Kernel laden kann. In einigen Fällen kann das Limit sehr klein sein (504M für ältere 486), in anderen Fällen ist das Limit recht groß (zum Beispiel 2G, 8G oder 128G auf i386 Systemen). Da sich der Kernel irgendwo in der root-Partition befinden kann, muss die gesamte root-Partition innerhalb dieser Grenze liegen. Mehr Details gibt es in diesem Kapitel. Eine gute Richtlinie ist, deine / Partition einfach komplett unterhalb der ersten 2GB zu lassen, es sei denn, du weißt, dass deine Plattform (und besonders deine Maschine!) damit umgehen kann.

Noch ein paar Gedanken zur Partitionierung:

- Bei deinem ersten Versuch und einem experimentellen System ist eine große `/` Partition und Swap das einfachste, bis du weißt, wieviel Platz du brauchst. Dadurch wirst du einige der Standard-Sicherheitsmechanismen opfern, die OpenBSD bietet, die separate Dateisysteme für `/, /tmp, /var, /usr` und `/home` benötigen.
- Ein sich im Internet befindliches oder anderweitig angreifbares System sollte eine separate Partition für `/var` haben (und vielleicht sogar eine separate für `/var/log`) für die Logdateien.
- Eine `/home` Partition ist durchaus nützlich. Neue Version des OS? Alles andere löschen und neu machen, aber die `/home` Partition einfach unberührt lassen. Denke aber trotzdem daran, eine Sicherungskopie deiner Konfigurationsdateien anzulegen!
- Eine separate Partition für alles, was eine große Anzahl Dateien erzeugen kann, kann man gegebenenfalls schneller formatieren und neu anlegen, als alle Dateien zu löschen. Siehe auch die Vom-Source-aus-erzeugen-FAQ für ein

Beispiel (`/usr/obj`).

- Wenn du dein System aus irgendeinem Grund aus dem Quellcode neu erzeugen willst, sollte der Source sich in `/usr/src` befinden. Wenn du kein separates Dateisystem für `/usr/src` hast, dann sorge unbedingt dafür, dass du in `/usr` genug Platz hast.
- Ein oftmals vergessener Punkt: du musst auf deinem System **nicht** allen Platz sofort vergeben! Da du jetzt vermutlich kaum noch eine Festplatte unter 20GB finden wirst, kann es durchaus Sinn machen, einfach ein bisschen Platz zu lassen. Wenn dir dann eine Partition zu klein wird, kannst du dann einfach aus deinem ungenutzten Platz eine neue Partition erstellen, deine existierende Partition auf die neue duplizieren, die /etc/fstab ändern und auf die neue Partition verweisen lassen, neu mounten, und schon hast du mehr Platz.
- Wenn du deine Partitionen zu klein wählst, wirst du das vermutlich spälter bereuen, wenn es an der Zeit ist, dein System auf den neuesten Stand zu bringen.
- Wenn du deinen Benutzern erlaubst auf `/var/www` zu schreiben (z.B. persönliche Webseiten), möchtest du es vielleicht auf eine separate Partition legen, so dass du quotas verwenden kannst, um den Speicher, den sie verwenden, zu begrenzen, und wenn sie die Partition füllen, keine anderen Teile des Systems betroffen sind.

4.8 - Multibooting OpenBSD/i386

Multibooting ist, wenn man mehrere Betriebssysteme auf einem Computer hat und auf irgendeine Art und Weise auswählen kann, welches OS gebootet werden soll. Das ist *keine* unbedeutende Aufgabe! Wenn du nicht verstehst, was du da machst, könntest du am Ende große Mengen an Daten von deinem Computer gelöscht haben. Neuen OpenBSD Benutzern wird *dringend* dazu geraten, mit einer leeren Festplatte auf einer extra hierfür freigestellten Maschine anzufangen und dann mit der gewünschten Konfiguration auf einer nicht im Einsatz befindlichen Maschine zu üben, bevor eine Multiboot Konfiguration auf einer eingesetzten Maschine versucht wird. FAQ 14 hat weitere Informationen über den OpenBSD Bootprozess.

Hier sind einige Optionen zum Multibooten:

Aktive Partitionen markieren

Dieses ist die wahrscheinlich am häufigsten übersehende und doch ab und zu beste Lösung für das Multibooten. Markiere einfach die aktive Partition in welchem OS auch immer du dich befindest so, dass sie diejenige ist, die beim nächsten Booten standardmäßig gebootet wird. So gut wie jedes OS bietet ein Programm, mit dem das bewerkstelligt werden kann; OpenBSDs ist fdisk(8), ähnlich genannte Programme gibt es unter Windows 9x und DOS, sowie unter vielen anderen Betriebssystemen. Dies kann sehr wünschenswert für Betriebssysteme oder Systeme sein, die sehr lange zum Herunterfahren und Neustarten brauchen -- du kannst sie markieren und das System neustarten, dann weggehen, eine Tasse Kaffee holen und dann zum System wiederkehren, das so hochgefahren ist, wie du es wolltest -- kein Warten auf den magischen Moment, wenn du das nächste OS auswählen musst.

Boot Floppy

Wenn du ein System hast, das nicht häufig zum Booten von OpenBSD verwendet wird (oder nicht möchtest, dass andere Benutzer des Computers mitbekommen, dass sich irgendwas geändert hat), ziehe das Verwenden einer Boot Floppy in Betracht. Verwende einfach eine der normalen OpenBSD Installationsdisketten und erstelle eine `/etc/boot.conf` Datei (ja, du musst außerdem ein `/etc` Verzeichnis auf der Floppy anlegen) mit dem Inhalt:

```
boot hd0a:/bsd
```

um das System zu veranlassen, von Festplatte 0, OpenBSD Partition ‚a', Kernel Datei `/bsd` zu booten. Denke daran, dass du auch von anderen Laufwerken booten kannst mit einer Zeile wie: "`boot hd2a:/bsd`" von der dritten Festplatte deines Systems. Um OpenBSD zu booten, lege die Floppy ein und starte neu. Um das andere OS zu booten, nehme die Floppy aus dem Laufwerk und starte neu.

In diesem Fall wird das boot(8) Programm geladen, sucht nach `/etc/boot.conf` und ließt sie aus. Die Zeile "`boot hd0a:/bsd`" weist boot(8) zu, von wo aus der Kernel geladen werden soll -- in diesem Fall von der ersten HD, die das BIOS sieht. Behalte im Hinterkopf, dass nur eine kleine Datei (`/boot`) von der Floppy geladen wird -- das System lädt den gesamten Kernel von der Festplatte, so dass ungefähr nur fünf Sekunden zum Bootprozess hinzugefügt werden.

Windows NT/2000/XP NTLDR

Um OpenBSD und Windows NT/2000/XP starten zu können, kannst du den NTLDR, den Bootloader von NT, benutzen. Um mit NT einen Multiboot durchführen zu können, benötigst du eine Kopie des OpenBSD Partition Boot Record (PBR). Nach dem Ausführen von installboot kannst du ihn mit dd(1) in eine Datei kopieren:

```
# dd if=/dev/rsd0a of=openbsd.pbr bs=512 count=1
```

Starte nun NT und platziere `openbsd.pbr` auf C:. Füge eine Zeile wie folgende am Ende von `C:\BOOT.INI` ein:

```
c:\openbsd.pbr="OpenBSD"
```

Nach einem Neustart solltest du OpenBSD im NT Loader Menü auswählen können. Zum NTLDR gibt es im NTLDR Hacking Guide mehr Informationen.

Auf Windows XP kannst du die Bootinformationen auch mit Hilfe der GUI editieren; siehe auch das XP Boot.ini HOWTO.

Programme, die vieles von diesen Tätigkeiten dir abnehmen stehen zur Verfügung, zum Beispiel BootPart. Dieses Programm kann unter Windows NT/2000/XP ausgeführt werden und lädt den OpenBSD PBR, platziert ihn auf deine NT/2000/XP Partition und fügt ihn zur `C:\BOOT.INI` hinzu.

Die OpenBSD Installation und der Upgrade Prozess installieren den OpenBSD Bootloader, dessen Pfad im PBR eingetragen ist, so dass du, wenn du deine OpenBSD Installation neuinstallierst oder aktualisierst, den oben angegebenen Prozess wiederholen musst, damit eine neue Kopie des OpenBSD PBR geladen wird.

Hinweis: Der Windows NT/2000/XP Bootloader ist nur in der Lage, Betriebssysteme von der primären Festplatte zu booten. Du kannst ihn nicht verwenden, um OpenBSD von der zweiten Festplatte zu laden.

Andere Bootloader

Zu einigen anderen Bootloadern, die von OpenBSD Anwendern erfolgreich eingesetzt worden sind, gehöhren GAG, OS-BS, The Ranish Partition Manager und GRUB.

OpenBSD und Linux (i386)

Bitte lies dir INSTALL.linux durch, um genaue Informationen zur Zusammenarbeit zwischen OpenBSD und Linux zu bekommen.

4.9 - Nach der Installation deine dmesg an dmesg@openbsd.org schicken

Um es nochmal allen ins Gedächtnis zu rufen: Es ist wichtig für die OpenBSD-Entwickler im Auge zu behalten, welche Art von Hardware funktioniert und welche eben nicht perfekt funktioniert.

Ein Zitat aus /usr/src/etc/root/root.mail

```
If you wish to ensure that OpenBSD runs better on your machines, please do us
a favor (after you have your mail system configured!) and type something like:
 # dmesg | mail -s "Sony VAIO 505R laptop, apm works OK" dmesg@openbsd.org
so that we can see what kinds of configurations people are running.  As shown,
including a bit of information about your machine in the subject or the body
can help us even further.  We will use this information to improve device driver
support in future releases.  (Please do this using the supplied GENERIC kernel,
not for a custom compiled kernel, unless you're unable to boot the GENERIC
kernel).  The device driver information we get from this helps us fix existing
drivers. Thank you!
```

Stelle sicher, dass du nicht nur E-Mail von deinem Konto senden, sondern auch empfangen kannst, für den Fall, dass dich ein Entwickler kontaktieren will, um etwas zu testen oder um deine Konfiguration zum Laufen zu bringen. Es ist nicht wichtig, dass du die E-Mail von einem Rechner mit OpenBSD verschickst, wenn also dieser Rechner keine E-Mail empfangen kann, dann gib einfach

```
$ dmesg | mail your-account@yourmail.dom
```

ein und leite dann diese Nachricht weiter an

```
dmesg@openbsd.org
```

wobei `your-account@yourmail.dom` dein regulärer Email-Account ist. (oder transferiere die dmesg Ausgabe mittels

FTP/scp/floppydisk/carrier-pigeon/...)

ANMERKUNG - Bitte schicke nur GENERIC Kernel dmesgs. Eigenkompilierte Kernel sind nicht hilfreich, wenn Gerätetreiber fehlen.

Bedenke auch, dass die dmesgs von einem Computer erhalten werden, der das spamd Spam Blocker System einsetzt. Dies könnte dazu führen, dass deine dmesg vom Mail Server für eine gewisse Zeit nicht angenommen wird. Habe Geduld, nach einer halben Stunde, Stunde oder so wird sie durchkommen.

4.10 - Ein Datei Set nach der Installation hinzufügen

"Oh nein! Ich habe bei der Installation eines der Datei Sets vergessen!"

Manchmal merkt man, dass man `comp36.tgz` (oder irgendeine andere Systemkomponente) DOCH benötigt, erst nachdem man mit der Installation schon fertig ist. Kein Problem: Es gibt zwei einfache Wege, diese Komponenten aus der Installation noch nachträglich einzuspielen:

Indem man den Upgrade Prozess benutzt

Boote einfach von deinem Installationsmedium (CD-ROM oder Floppy) und wähle "Upgrade" (und nicht Install). Wenn du zur Liste mit den Datei Sets kommst, wähle die Sets, die du beim ersten Mal vergessen hast, wähle deine Quelle aus und lass die Installation beginnen.

Indem man tar(1) benutzt

Die Installationsdateien sind einfach komprimierte tar Dateien und du kannst sie dementsprechend am root (/) des Dateisystems auspacken:

```
# cd /
# tar xzvpf comp36.tgz
```

Du darfst NICHT die `,p'` Option im oben genannten Kommando vergessen, sonst werden die Dateirechte nicht richtig erzeugt!

Ein häufig gemachter Fehler ist, zu denken, man könne pkg_add(1) benutzen, um fehlende Datei Sets einzuspielen. Das funktioniert aber nicht. pkg_add(1) ist für Package Dateien, nicht für einfache tar Dateien wie die Installations-Sets.

4.11 - Was ist ‚bsd.rd'?

`bsd.rd` ist ein "RAM Disk" Kernel. Diese Datei kann sehr nützlich sein; viele Entwickler lassen sie mit Absicht immer im Quellverzeichnis ihres Dateisystems.

Ihn einen "RAM Disk Kernel" zu nennen, beschreibt das root Dateisystem des Kernels -- es ist kein physikalisches Laufwerk, die Werkzeuge, die nach dem Booten von `bsd.rd` nutzbar sind, befinden sich im Kernel und werden aus einem RAM-basierten Dateisystem geladen. `bsd.rd` enthält auch ein brauchbares Set an Werkzeugen für System-Wartung und Installation.

Auf manchen Plattformen ist `bsd.rd` sogar die bevorzugte Installationstechnik. Du plazierst diesen Kernel in ein vorhandenes Dateisystem, bootest ihn und führst daraus die Installation aus. Auf den meisten Plattformen kannst du, falls du eine ältere Version von OpenBSD hast, eine neue Version von `bsd.rd` per FTP holen, davon booten und eine neue Version von OpenBSD installieren, ohne eine Floppy oder CD-ROM zu benutzen.

Hier zum Beispiel das Booten von `bsd.rd` auf einem i386 System:

```
Using Drive: 0 Partition: 3
reading boot.....
probing: pc0 com0 com1 apm mem[639k 255M a20=on]
disk: fd0 hd0
>> OpenBSD/i386 BOOT 2.02
boot> boot hd0a:/bsd.rd
. . . normal boot to install . . .
```

Wie schon gesagt, wirst du in das Installationsprogramm geführt, aber du kannst natürlich auch auf die Shell gehen, um dein System zu pflegen oder zu administrieren.

Die einfache Regel, um `bsd.rd` booten zu können, lautet: Tausche `/bsd` durch `bsd.rd` aus, was auch immer das auf deiner Plattform heißen mag.

4.12 - Allgemeine Installationsprobleme

4.12.1 - Mein Compaq erkennt nur 16M RAM

Einige Compaq Systeme haben das Problem, dass der gesamte Hauptspeicher vom OpenBSD ‚second stage' Bootloader nicht ordentlich erkannt wird und nur 16M erkannt und von OpenBSD verwendet werden. Dies kann entweder durch das Erstellen/Editieren von der `/etc/boot.conf` oder durch das Eingeben von Befehlen am "`boot>`" Prompt, bevor OpenBSD lädt, behoben werden. Falls du eine Maschine mit 64M RAM hast, aber OpenBSD nur die ersten 16M erkennt, sieht der Befehl, den du benutzen würdest, wie folgt aus:

```
machine mem +0x3000000@0x1000000
```

um 48M (`0x3000000`) nach den ersten 16M (`0x1000000`) hinzuzufügen. Typischerweise würdest du den oben angegeben Befehl am Installationsprompt `boot>` der Floppy/CD-ROM eingeben, neustarten und eine `/etc/boot.conf` Datei anlegen, die den oben angegebenen Befehl beinhaltet, so dass alle zukünftigen Starts des Systems den ganzen verfügbaren Speicher erkennen können, wenn du ein System mit diesem Problem hättest.

Es wurde ebenfalls berichtet, dass ein ROM Update dieses Problem auf *einigen* Systemen löst.

4.12.2 - Mein i386 bootet nach der Installation nicht

Deine Installation verlief gut, doch beim ersten Hochfahren erkennst du keine Anzeichen dafür, dass OpenBSD versucht zu starten. Es existieren einige bekannte Gründe für dieses Problem:

- **Keine Partition wurde in fdisk(8) als aktiv markiert.** Um dies zu beheben, starte das System unter Verwendung der Boot Floppy oder anderer Medien neu und "markiere" eine Partition als "aktiv" (bootfähig). Siehe hier und hier.
- **Kein gültiger Bootloader wurde jemals auf die Platte geschrieben** Falls du mit "Y" auf die Frage "Use entire disk for OpenBSD?" während der Installation geantwortet hast oder die "`reinit`" Option von fdisk(8) verwendet hast, wurde der OpenBSD Boot Record im Master Boot Record der Platte installiert; ansonsten wurde der existierende Master Boot Record Code nicht berührt. Dies wird ein Problem sein, wenn kein anderer Boot Record existierte. Eine Lösung ist, das Installationsmedium wieder zu booten, auf die Shell zurückzugreifen und fdisk(8) aufzurufen, um den MBR Code von der Befehlszeile aus zu aktualisieren:

  ```
  # fdisk -u wd0
  ```

  Hinweis: Die "update" Option im interaktiven ("-e") Modus von fdisk wird keine Signatur Bytes schreiben, die benötigt sind, um die Platte bootfähig zu machen.
- **In einigen wenigen Fällen kann etwas mit der ‚second stage' Bootloader Installation schief gelaufen sein.** Das Neuinstallieren des ‚second stage' Bootloaders wird hier besprochen.

4.12.3 - Meine (ältere, langsamere) Maschine bootet, aber hängte beim ssh-keygen Prozess

Es ist sehr wahrscheinlich, dass dein System einwandfrei funktioniert, nur dass die ssh Schlüsselgenerierung eine Weile dauert. Eine SPARCStation2 oder ein Macintosh Quadra können bis zu 45 Minuten oder länger in Anspruch nehmen, um die drei ssh-keygen(1) Schritte auszuführen, einige Systeme brauchen sogar noch länger. Lass es einfach abschließen; es wird nur ein einziges Mal pro Installation gemacht.

4.12.4 - Ich bekam die Meldung "Failed to change directory", als ich die Installation durchführte

Wenn du eine FTP Installation eines ‚flavors' während der -*beta* Phase des OpenBSD Entwicklungs Zyklus durchführst, könntest du dies lesen:

```
Do you want to see a list of potential FTP servers?  [yes] Enter
Getting the list from 192.128.5.191 (ftp.openbsd.org)... FAILED
Failed to change directory.
Server IP address or hostname?
```

Dies ist normal und ein erwartetes Verhalten während der ‚pre-release' Phase des Zyklus. Das Installationsprogramm sucht nach der FTP Liste auf dem primären FTP Server in einem Verzeichnis, das bis zum Release Datum nicht verfügbar ist, so dass du die oben genannte Meldung bekommst.

Verwende einfach eine FTP mirror Liste um deinen favorisierten FTP mirror zu finden und gib ihn manuell ein, wenn du danach gefragt wirst.

Hinweis: Du solltest die Meldung nicht sehen, wenn du *-release* oder von einer CD-ROM installierst.

4.12.5 - Wenn ich mich einlogge, bekomme ich "login_krb4-or-pwd: Exec format error"

Kerberos IV wurde aus OpenBSD 3.4 entfernt, aber wenn du ein Upgrade ausgeführt hast, werden die Kerberos IV Binaries weiterhin auf deinem System sein. Dies ist ein Problem auf der i386 Plattform, da die alten Kerberos Dateien im a.out Format sind und daher nicht mit einem ELF Kernel (der die a.out Emulation nicht aktiviert hat, wie hier vermerkt) ausgeführt werden können. Falls du auf dieses Problem gestoßen bist, musst du die krb4 Authentifikation überschreiben, wenn du dich einloggst:

```
OpenBSD/i386 (puffy.openbsd.org) (ttyC0)

login: joeuser:passwd
password:
```

Du kannst die gleiche "*username*`:passwd`" Syntax mit einer ssh Verbindung und mit su(1) verwenden, um auf dein System zuzugreifen. Nun editiere `/etc/login.conf` und entferne die krb4 Referenzen.

4.12.6 - Meine fdisk Partitionstabelle ist kaputt oder leer!

Gelegentlich finden Benutzer ein funktionsfähiges System vor, das aber beim Verwenden von `fdisk wd0` eine vollständig leere (oder ab und zu, vermüllte) Partitionstabelle aufweist. Dies ist normalerweise verursacht, wenn eine Partition in fdisk(8) erstellt wurde, die ein Offset von 0 Sektoren hat, anstatt dem einem Track Offset, den sie haben sollte (Hinweis: Dies nimmt an, dass es sich um die i386 oder amd64 Plattform handelt. Andere Plattformen benötigen andere Offsets, einige sogar GAR KEINEN Offset). Das System bootet dann unter Verwendung des PBR und nicht des MBR.

Obwohl diese Konfiguration funktioniert, kann es ein Wartungsproblem sein und sollte korrigiert werden. Um dies zu korrigieren, muss das Dateisystem der Platte normalerweise von Grund auf neu erstellt werden (obwohl, wenn du WIRKLICH weißt was du tust, kannst du in der Lage sein, nur dein Disklabel und den MBR neu zu erstellen und musst nur die erste OpenBSD Partition der Festplatte neu errichten).

4.13 - Anpassen des Installationsprozesses

`siteXX.tgz` Datei

Die OpenBSD Installationsskripte erlauben das Auswählen eines Benutzer-erstellten Sets, genannt "`siteXX.tgz`", wobei XX die Release Version (z.B. 36) ist. Das `siteXX.tgz`" Datei Set ist, wie die anderen Datei Sets, ein gzip(1) komprimiertes tar(1) Archiv basierend auf ‚/' und wird wie die anderen Sets mit den Optionen `xzpf` ent‚tar'rt. Dieses Set wird zuletzt installiert, nach allen anderen Datei Sets.

Dieses Datei Set erlaubt es dem Benutzer, Dateien hinzuzufügen oder Dateien zu überschreiben, die von den ‚normalen' Sets installiert worden sind und daher die Installation oder das Upgrade anpassen zu können.

Einige Beispielverwendungen einer `siteXX.tgz` Datei:

- Erstelle eine siteXX.tgz Datei, die alle Änderungen beinhaltet, die du seit dem ersten Installieren von OpenBSD gemacht hast. Dann, wenn du das System neuerstellen musst, wählst du einfach siteXX.tgz während der Neuinstallation aus und alle Änderungen von dir werden auf dem neuen System nachgebildet.
- Erstelle eine Serie von Maschinen-spezifischen Verzeichnissen, die jeweils eine siteXX.tgz Datei beinhalten, die

wiederum spezifische Dateien für diese Maschine beinhalten. Installationen von Maschinen (z.B. Systeme mit unterschiedlichen Grafikkarten) einer bestimmten Kategorie können durch das Auswählen der passenden siteXX.tgz Datei abgeschlossen werden.

- Stecke die Dateien, die du routinemäßig auf einen gleichen oder ähnlichen Weg anpasst, in eine siteXX.tgz Datei -- `/etc/skel` Dateien, `/etc/pf.conf`, `/var/www/conf/httpd.conf`, `/etc/rc.conf.local`, etc.

`install.site/upgrade.site` Skripte

Als letzten Schritt im Installations/Upgrade Prozess sucht das Skript im root Verzeichnis des neu installierten/aktualisierten System nach `install.site` oder `upgrade.site`, je nach aktuellem Prozess, und führt das Skript in einer Umgebung, die auf das Wurzelverzeichnis des installierten/aktualisierten System ‚chroot'ed' ist. Denke daran, dass das Upgrade von einem gebootetem Dateisystem aus ausgeführt wird, so dass dein Ziel tatsächlich auf `/mnt` gemountet ist. Trotzdem kann dein Skript so geschrieben werden, als ob es im "normalen" Wurzelverzeichnis deines Dateisystems ausgeführt wird. Da das Skript ausgeführt wird, wenn alle Dateien bereits installiert worden sind, hast du fast volle Funktionalität deines Systems (allerdings im Single User Modus), wenn dein Skript läuft.

Bedenke, dass das `install.site` Skript in der `siteXX.tgz Datei` liegen sollte, während das `upgrade.site` Skript vor dem Upgrade in das Wurzelverzeichnis gelegt werden kann, oder in eine `siteXX.tgz` Datei.

Die Skripte können für viele Dinge verwendet werden:

- Entferne Dateien, die installiert/aktualisiert worden sind, die du auf dem System aber nicht haben möchtest.
- Erstelle ein sofortiges Backup/Archiv des neuen Systems, bevor du es dem Rest der Welt aussetzt.

Unglücklicherweise ist es nicht möglich, pkg_add(1) von dieser Umgebung aus zu verwenden.

Die Kombination von `siteXX.tgz` und `install.site/upgrade.site` Dateien sollen den Benutzern eine breite Palette von Fähigkeiten bieten, ohne dass sie selbst eigene Installations-Sets erstellen müssen.

4.14 - Wie kann ich eine Anzahl gleichartiger Systeme installieren?

Hier sind einige Anwendungen aufgelistet, die du verwenden kannst, wenn du eine Anzahl von ähnlichen OpenBSD Systemen installieren musst.

`siteXX.tgz` und `install/upgrade.site` Dateien

Siehe den vorherigen Artikel.

Von dump(8) wiederherstellen

Auf den meisten Plattformen beinhaltet das Boot Medium das restore(8) Programm, welches verwendet werden kann, um ein Backup, das mit dump(8) erstellt wurde, zu extrahieren. Daher kannst du von Floppy, CD oder bsd.rd Datei booten, dann fdisk, disklabel ausführen und mittels restore die gewünschte Konfiguration von Band oder einem anderen Medium wiederhestellen und die Boot Blöcke installieren. Weitere Details gibt es hier.

Platten ‚imaging'

Leider existiert kein bekanntes Platten ‚imaging' Programm, das FFS unterstützt und so nur den aktiv genutzten Dateispeicher nutzen könnte. Die meisten der größeren Platten ‚imaging' Lösungen behandeln OpenBSD Partitionen als "allgemeine" Partition und können ein Image der gesamten Platte erzeugen. Damit erreichst du zwar dein Ziel, aber verbrauchst für gewöhnlich eine große Menge an Speicher -- eine leere, 10G `/home` Partition wird 10G Speicher im Image verbrauchen, selbst wenn sich dort keine einzige Datei befindet. Während du normalerweise ein Laufwerksimage auf ein größeres Laufwerk installieren kannst, wirst du nicht in der Lage sein, den zusätzlichen Speicher direkt zu nutzen, und du wirst auch nicht in der Lage sein, ein Image auf ein kleineres Laufwerk zu installieren.

Falls das eine akzeptable Situation für dich ist, könnte der dd Befehl deinen Ansprüchen, eine Platte auf eine andere zu kopieren, Sektor-für-Sektor, genügen. Dies bietet dir die gleiche Funktionalität wie die der kommerziellen Produkte ohne den Kosten.

4.15 - Woher bekomme ich eine dmesg(8), damit ich ein Problem mit der Installation melden kann?

Wenn man ein Problem meldet, ist es entscheidend, eine komlette dmesg(8) des Systems einzufügen. Wie auch immer, oft ist es so, dass du dies machen musst, wenn das System nicht ordnungsgemäß funktioniert oder nicht installieren will, so dass du weder Platten, Netzwerk oder andere Ressourcen zur Verfügung hast, um deine dmesg an die passende Mailing Liste zu senden. Es gibt aber andere Wege:

- **Floppy** Die Boot Disketten und die CD-ROM haben genügend Anwendungen, um deine dmesg auf eine MSDOS Floppy zu schreiben, damit eine andere Maschine sie lesen kann. Lege eine MSDOS formatierte Floppy in dein Diskettenlaufwerk und führe folgende Befehle aus:

  ```
  mount -t msdos /dev/fd0a /mnt
  dmesg > /mnt/dmesg.txt
  umount /mnt
  ```

  Falls du ein anderes OpenBSD System besitzt, kannst du es ebenfalls auf eine OpenBSD kompatible Floppy schreiben -- oftmals hat die Boot Floppy genügend freien Raum, um die dmesg zu halten. In dem Fall lasse das "`-t msdos`" oben weg.
- **Serielle Konsole:** Eine serielle Konsole verwenden und die Ausgabe auf einer anderen Maschine abzufangen ist oftmals der beste Weg um Diagnose Informationen erhalten zu können - insbesondere, wenn der Computer direkt nach dem Hochfahren in einem ‚panic' endet. Neben einem weiteren Computer benötigst du ebenfalls ein passendes serielles Kabel (oft ein Null-Modem Kabel) und ein Terminal Emulator Programm, das die Bildschirmausgabe in eine Datei schreiben kann.

  Generelle Informationen zum Aufsetzen eines seriellen Terminals sind an einer anderen Stelle des FAQs verfügbar; um eine Log der Installation abfangen zu können, sind in der Regel folgende Befehle ausreichend.

  i386

  Gib am Bootloader Prompt ein

  ```
  boot> set tty com0
  ```

  Dies teilt OpenBSD mit, dass die erste serielle Schnittstelle (in PC Dokumentationen oft COM1 oder COMA genannt) als serielle Konsole verwendet werden soll. Die standardmäßige Baudrate ist 9600.

  Sparc/Sparc64

  Diese Maschinen starten automatisch mit einer seriellen Konsole, wenn sie ohne angeschlossener Tastatur gestartet wurden. Falls du Tastatur und Monitor angeschlossen hast, kannst du das System trotzdem dazu bringen, eine serielle Konsole zu verwenden, indem du folgende Befehle am `ok` Prompt aufrufst.

  ```
  ok setenv input-device ttya
  ok setenv output-device ttya
  ok reset
  ```
- **FTP:** Unter bestimmten Umständen kann es dir möglich sein, mit Hilfe des ftp(1) Clients der Boot Diskette oder CD-ROM, die dmesg zu einem lokalen FTP Server zu senden, von wo aus du sie später wiederholen kannst.

4.16 - OpenBSD unter Verwendung von `bsd.rd-aout` upgraden/neuinstallieren

Es ist normalerweise möglich, Upgrades und Installationen mit dem bsd.rd Kernel durchzuführen. Mit OpenBSD 3.4 allerdings wechselte die i386 Plattform das ‚executable' Format von a.out auf ELF, so dass ältere Bootloader (OpenBSD 3.3 und früher) nicht unter dem neuen-Format `bsd.rd` Kernel ausgeführt werden können.

Um dieses Problem zu umgehen und Upgrades unter Verwendung von `bsd.rd` gewährleisten zu können, wurde eine a.out Version von `bsd.rd` in der OpenBSD/i386 v3.5 FTP Distribution zur Verfügung gestellt. Diese Datei, `bsd.rd-a.out`, kann von OpenBSD 3.3 und niedriger gebootet werden, aber es ist ein authentischer OpenBSD 3.5 Kernel, mit eingebundenem ELF Bootloader, so dass es zum Bootstrappen von OpenBSD/i386 3.5 von einem älteren System aus verwendet werden kann.

Lade einfach `bsd.rd-a.out` herunter und platziere ihn in dem root Verzeichnis deines Systems. Lade ihn anstatt dem

normalen `/bsd` oder `/bsd.rd` Kernels, so wie es hier gezeigt wird (Natürlich unter Angabe von `bsd.rd-a.out` als Boot Kernel).

Nach der Installation einer minimalen 3.5 Installation (`base35.tgz, etc35.tgz, bsd`), lade dir eine 3.6 (oder *-snapshot*) `bsd.rd` Datei herunter und installiere davon.

[Zurück zum Haupt-Index] [Zum Kapitel 3 - Wo man OpenBSD herbekommt] [Zum Kapitel 5 - Das System aus dem Source-Code erzeugen]


www@openbsd.org
$OpenBSD: faq4.html,v 1.62 2005/02/27 12:04:57 jufi Exp $

OpenBSD

[FAQ Index] [Zum Kapitel 4 - OpenBSD 3.6 Installationsanleitung] [Zum Kapitel 6 - Netzwerk]

5 - Das System aus dem Source-Code erzeugen

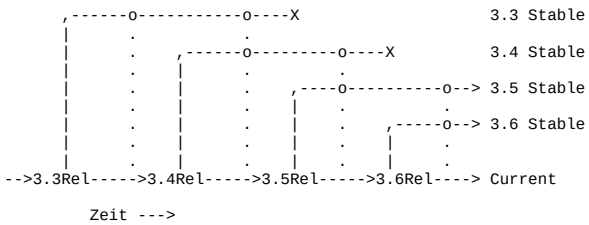
Inhaltsverzeichnis

5.1 - OpenBSDs ‚flavors'

Es gibt drei ‚flavors' von OpenBSD:

- **-release:** Die Version von OpenBSD, die alle 6 Monate auf CD ausgeliefert wird.
- **-stable:** Release und zusätzliche Patches, die für Sicherheit und Zuverlässigkeit als notwendig erachtet werden.
- **-current:** Die jetzt im Moment aktuelle Version von OpenBSD, die sich in die nächste release-Version verwandeln wird.

Grafisch sieht die Entwicklung dieser ‚flavors' ungefähr so aus:



-Current ist der Ort, an dem die aktive Arbeit gemacht wird und wird dann das nächste *-release* von OpenBSD. Alle sechs Monate, wenn eine neue Version von OpenBSD veröffentlicht wird, wird *-current* markiert (tagged) und wird *-release*: ein fester Punkt in der Geschichte des Source-Trees. Kein *-release* wird jemals verändert; es ist das, was auf den CDs und FTP-Servern vorliegt.

-Stable basiert auf *-release* und ist ein ‚branch' des Hauptentwicklungsweges von OpenBSD. Dieser ist deshalb auch als *patch branch* bekannt. Wenn wichtige Korrekturen an *-current* gemacht werden, werden sie in die *-stable-*‚branches' ‚zurückportiert' (merged). In der obigen Illustration sind die vertikalen gepunkteten Linien Fehlerkorrekturen, die in die *-stable-*‚branches' eingefügt wurden. Du wirst auch erkennen, dass der Lebenszyklus des *3.3-stable-*‚branch' im obigen Beispiel mit Erscheinen des *3.5-release* beendet war und der *3.4-stable-*‚branch' mit Erscheinen von *3.6-release* beendet war -- alte Versionen werden typischerweise noch zwei Releases lang weitergepflegt. Es braucht nunmal Ressourcen und Zeit, ältere Versionen zu pflegen, und obwohl wir das gerne tun würden, konzentrieren wir uns doch lieber auf neue Funktionen. Der *-stable-*‚branch' kann vom Design her sehr leicht aus dem *-release-*‚branch' der selben Version erzeugt werden (z.B. von *3.6-release* zu *3.6-stable*).

Der *-stable-*‚branch' ist *-release* plus Patches, die man auf der Errataseite findet, und ein paar einfachen Fixes, die keinen Errata-Eintrag verdienen. Normalerweise ist die Bedienung und der Betrieb von *-stable* genauso wie der des *-release*, auf dem er basiert. Wenn die Manualseiten geändert werden müssen, wird es wahrscheinlich nicht in *-stable* eingefügt werden. Mit anderen Worten, Unterstützung für neue Hardware wird NICHT in *-stable* eingefügt und neue Funktionen werden nur dann eingefügt, wenn das als sehr wichtig erachtet wird.

An dieser Stelle sei erwähnt, dass der Name ‚*-stable*' nicht ausdrücken soll, dass *-current* unzuverlässig sei. Stattdessen ändert und entwickelt sich *current*, während die Funktionsweise von *-stable* nicht mehr geändert wird, so dass du dein System nicht neu erlernen musst. In der Tat kann es sein, da unsere Bemühungen darin bestehen, OpenBSD zu verbessern, dass du die Zuverlässigkeit von *-current* besser als bei *-stable* empfindest.

Warnung: *-current* ist ein sich bewegendes Ziel. Es ändert sich fast minütlich und kann sich genauso gut mehrere Male in der Zeit ändern, die man braucht, um dem Source-Code zu holen. Obwohl die Entwickler hart daran arbeiten, sicherstellen zu können, dass das System immer compiliert und dass es keine großen Fehler gibt, ist es durchaus möglich, *-current*-Source herunterzuladen, der nicht compiliert, während es fünf Minuten später wieder einwandfrei funktioniert. Es gibt ebenfalls ‚flag days' und große Systemänderungen, die die Entwickler mit Tools navigieren, die nur einmal genutzt werden können, was bedeutet, dass source-basierendes Aktualisieren nicht möglich ist. **Wenn du damit nicht umgehen kannst, lass deine Finger von *-current*.**

Die meisten Benutzer sollten also einfach *-stable* oder *-release* benutzen. Da das jetzt klar ist, sollte man aber auch wissen, dass viele Leute *-current* auf Produktionssystemen fahren, und das ist insofern wichtig, weil es hilft, Bugs zu finden und neue Funktionalitäten zu testen. Wenn du aber nicht weißt, wie man sauber Probleme beschreibt, diagnostiziert oder damit umgeht, rede dir (oder jemand anderem) besser nicht ein, du würdest ‚dem Projekt helfen', indem du *-current* benutzt. ‚Es funktioniert nicht!' ist keine nützliche Fehlermeldung. ‚The recent changes to the pciide driver broke compatibility with my Slugchip-based IDE interface, dmesg of working and broken systems follow...' könnte dagegen durchaus ein sinnvoller und nützlicher Bericht sein.

Es mag Zeiten geben, in denen auch ein ‚normaler' Benutzer gerne ‚auf des Messers Schneide' leben will und *-current* benutzt. Der häufigste Grund dafür ist, dass er ein Gerät hat, was nicht von *-release* unterstützt wird (und daher natürlich auch nicht von *-stable*), oder er möchte eine neue Funktion aus *-current* benutzen. In diesem Fall hat er die Wahl zwischen *-current* oder dem Nichtbenutzen seines Gerätes, und dann ist eben das Benutzen von *-current* oftmals weniger schlimm. Man sollte dann aber auf keinen Fall ein Händchenhalten der Entwickler erwarten, falls es zu Problemen kommt.

Snapshots

Zwischen den formalen Veröffentlichungen neuer Versionen von OpenBSD werden *Snapshots* über die FTP-Seiten verfügbar gemacht. Wie der Name schon sagt, sind das Versionen vom Code, der gerade zu eben diesem Zeitpunkt aktuell war, als der Übersetzer des Snapshots sich eine Kopie des Source-Codes für eben diese Plattform gemacht hat. Denke bitte daran, dass es auf einigen Plattformen auch einige TAGE sein können, bis der Snapshot dann auch kompiliert und zur Verfügung gestellt wird. Es gibt auch keinerlei Garantien, dass die Snapshots funktionsfähig oder auch nur installationsfähig sind. Oftmals werden Snapshots dann erzeugt, wenn es eine Änderung gibt, die getestet werden muss. Bei einigen Plattformen werden Snapshots auf fast täglicher Basis erstellt, andere sind weniger regelmäßig. Wenn du unbedingt *-current* benutzen willst, ist ein aktueller Snapshot oft das einzige, was du dazu brauchst, und das Upgraden auf den aktuellen Snapshot ist auch eine Voraussetzung, um *-current* zu erzeugen.

Manchmal wird gefragt, ob es einen Weg gibt, genau den Source-Code zu bekommen, der zur Erzeugung eines Snapshot eingesetzt wurde. Die Antwort ist ‚Nein'. Erstens ergibt sich daraus kein besonderer Vorteil. Zweitens werden Snapshots nach Bedarf erzeugt, wenn die Zeit es erlaubt oder es genügend Ressourcen gibt. Auf schnellen Plattformen könnten z.B. mehrere Snapshots an einem Tag freigegeben werden. Auf den langsameren kann das schon mal eine ganze Woche dauern. Und das Platzieren einer Markierung (Tag) für jeden Snapshot im Source-Code ist nun wirklich extrem unpraktisch.

Die Dinge synchron halten

Es ist wichtig zu verstehen, dass OpenBSD ein Betriebssystem ist und als ganzes gesehen werden muss, nicht ein Kernel mit einem Schwanz an Applikationen, die drangehängt sind. Du musst sicherstellen, dass dein Kernel, dein ‚userland' (die unterstützenden Werkzeuge und Dateien) und dein `Ports`-Tree alle synchronisiert sind, oder es werden unangenehme Dinge passieren. Oder anders ausgedrückt (da es immer wieder Leute gibt, die das versuchen), kannst du keine brandneuen `Ports` auf einem System laufen lassen, das einen Monat alt ist oder einen Kernel aus *-current* neu erzeugen und dann erwarten, dass er mit einem *release*-‚userland' zusammenarbeitet. Ja, das heißt, dass du dein System auf aktuellem Stand halten musst, wenn du ein neues Programm laufen lassen willst, das z.B. erst heute in den Ports-Tree eingefügt wurde. Tut uns erneut leid, aber OpenBSD hat nunmal nur sehr begrenzte Ressourcen, so dass wir sowas nicht ändern können.

Man sollte verstehen, dass der Update-Prozess **nur in eine Richtung unterstützt wird: Von älter zu neuer** und von *-stable* zu *-current*. Du kannst kein *3.6-current* (oder einen Snapshot) laufen lassen und dich dann entscheiden, dass dir das zu gefährlich ist, und einfach nach *3.6-stable* zurückgehen. Du bist auf dich allein gestellt, wenn du einen anderen Weg wählst als den normalen und unterstützten, nämlich dein System von Grund auf neu zu installieren; erwarte bitte keinerlei Hilfeleistung vom OpenBSD-Entwicklerteam.

Ja, das soll wirklich heißen, dass du lange und intensiv darüber nachdenken sollst, bevor du dich an *-current* wagst.

5.2 - Warum muss ich mein System vom Source aus erzeugen?

Eigentlich brauchst du das vermutlich nicht.

Einige Gründe, warum man NICHT vom Source aus erzeugen sollte:

- Das Compilieren deines Systems als Weg zum Upgraden zu nutzen, wird nicht unterstützt.
- Du wirst KEINE bessere Systemleistung durch das Compilieren deines eigenen Systems erhalten.
- Compiler-Optionen verändern führt eher dazu, dass du dein System zerschießt, statt es zu verbessern.

Einige Gründe, warum du tatsächlich vom Source aus erzeugen möchtest oder musst:

- Neue Funktionalitäten testen oder entwickeln.
- Das System zu compilieren führt zu einer hohen Last auf deinem Computer, was ein Weg sein kann um sicherzustellen, dass das System, das du zusammengesetzt oder erworben hast, in ziemlich guter Rechenverfassung ist.
- Wenn du dem stable branch folgen möchtest.
- Wenn du eine stark angepasste Version von OpenBSD für einen sehr speziellen Anwendungszweck erzeugen willst.

Das OpenBSD-Team stellt neue Snapshots bereit, die auf *-current*-Code basieren, in einem sehr regulären Zeitrahmen für alle Plattformen bereit. Es ist sehr wahrscheinlich, dass das alles ist, das du benötigst, um *-current* laufen zu lassen.

Der häufigste Grund, vom Source aus zu erzeugen, ist das Folgen vom *-stable*-‚branch', da das Erzeugen vom Source aus die einzige unterstützte Option ist ...

5.3 - OpenBSD vom Source aus erzeugen

5.3.1 - Überblick über den Erzeugungsprozess

OpenBSD vom Source aus zu erzeugen beinhaltet einige Schritte:

- Zur naheliegendsten Binary aktualisieren.
- Den passenden Source-Code runterladen.
- Den neuen Kernel erzeugen und von diesem booten
- Das ‚userland' erzeugen (‚make build').

Es gibt ein paar weitere Schritte, die einige Benutzer eventuell durchführen möchten, abhängig von dem Verwendungszweck der Erzeugung und ob X installiert ist:

- Ein ‚Release' erzeugen.
- X erzeugen.

5.3.2 - Zur naheliegendsten Binary aktualisieren oder diese installieren

Der erste Schritt beim Erzeugen vom Source aus besteht darin, sicherzustellen, dass du die verfügbare Binary, die am nächsten an deinem Ziel liegt, instaliert hast. Verwende diese Tabelle um zu sehen wo du bist, wo du hingehen möchtest und mit welcher Binary du starten solltest: +

Du bist beim	Ziel	Binary-Upgrade auf	dann ...
Alten -release	Neues Release	Neuestes Release	Done!
-release	-stable	Neuestes Release	Downloade & erzeuge *-stable*
-release	-current	Aktuellster Snapshot	(optional) Downloade & erzeuge *-current*
Alten -current	-current	Aktuellster Snapshot	(optional) Downloade & erzeuge *-current*

Es ist empfohlen, dass du die Binary unter Verwendung der ‚Upgrade'-Option des Installationsmediums verwendest. Wenn das nicht möglich ist, kannst du auch die Binaries entpacken, so wie es hier beschrieben steht. Unabhängig davon musst du den gesamten Upgradeprozess durchführen, einschließlich dem Erzeugen möglicher Benutzer oder notwendige `/etc`-Verzeichnisänderungen.

5.3.3 - Den passenden Source-Code runterladen

OpenBSD-Source wird unter Verwendung vom CVS-‚content management system' verwaltet, und cvs(1) wird genutzt, um eine Kopie des gewünschten Sources auf deine lokale Maschine zum compilieren zu ziehen. Dies kann unter Verwendung eines AnonCVS-Servers geschehen (einer Maschine, die eine öffentlich zugängliche Kopie vom gesamten CVS-Repository hält, das vom OpenBSD-Projekt genutzt wird, oder von einem lokalen CVS-Repository, das du unter Verwendung von CVSup oder CVSync pflegst, welche über Packages verfügbar sind. CVSup kann ebenfalls in einem ‚checkout-Mode genutzt werden, aber das wird an dieser Stelle nicht behandelt. Wenn du mehrere Maschinen hast, auf denen du den Source-Code-Tree bewahren willst, kann es für dich durchaus nützlich sein, ein lokales CVS-Repository zu führen, erstellt und gepflegt unter Verwendung von CVSup oder CVSync.

Nach der Entscheidung, welchen AnonCVS-Server du gerne verwenden willst, musst du ein ‚checkout' für den Source-Tree durchführen, danach musst du dann den Tree durch das Ausführen von ‚updates' pflegen, um aktualisierte Dateien auf deinen lokalen Tree zu ziehen.

Das CVS(1)-Kommando hat viele Optionen, einige von ihnen sind ***notwendig***, um ein ‚checkout' und ‚update' für einen nutzbaren Tree durchzuführen. Andere Kommandos können dazu führen, dass dein Tree unbrauchbar wird. Diesen Anweisungen hier zu folgen und sie zu verstehen ist wichtig.

-*current* folgen

In diesem Fall nehmen wir ann, dass wir einen öffentlichen AnonCVS-Server, **anoncvs@anoncvs.be.openbsd.org:/cvs**, verwenden. Wir nehmen des weiteren an, dass du sh(1) als deine Kommandoshell verwendest, wenn du eine andere Shell nutzen solltest, musst du diese Kommandos dementsprechend anpassen.

Um ein ‚checkout' für einen *-current*-CVS-src-Tree durchzuführen, kannst du folgendes nutzen:

```
# cd /usr
# export CVSROOT=anoncvs@anoncvs.be.openbsd.org:/cvs
# cvs -d$CVSROOT checkout -P src
```

Sobald du einen Tree hast, kannst du ihn später jederzeit aktualisieren:

```
# cd /usr/src
# export CVSROOT=anoncvs@anoncvs.be.openbsd.org:/cvs
# cvs -d$CVSROOT up -Pd
```

-*Stable* folgen

Wenn du einen ‚checkout' für einen alternativen ‚branch' des Trees durchführen willst, wie zum Beispiel dem *-stable*-‚branch', musst du den ‚`-r`'-Modifizierer deinem Checkout nutzen:

```
# cd /usr/src
# export CVSROOT=anoncvs@anoncvs.be.openbsd.org:/cvs
# cvs -d$CVSROOT checkout -rOPENBSD_3_6 -P src
```

Dies wird die src-Datieen vom OPENBSD_3_6-‚branch' ziehen, welcher ebenfalls als ‚patch branch' oder ‚-stable' bekannt ist. Den Code würdest du ähnlich aktualisieren:

```
# cd /usr/src
# export CVSROOT=anoncvs@anoncvs.be.openbsd.org:/cvs
# cvs -d$CVSROOT up -rOPENBSD_3_6 -Pd
```

Tatsächlich ist CVS nett genug, eine Markierung (Tag) an die runtergeladene Dateisystem anzuhängen, so dass du dir den ‚`-rOPENBSD_3_6`'-Teil der Kommandozeile nicht merken musst, CVS wird sich immer daran erinnern, bis du ihn explizit zurücksetzt oder eine neue Markierung unter Verwendung der ‚`-A`'-Option mit ‚`update`' verwendest. Jedoch ist es vermutlich besser, zu viele Informationen in deine CVS-Kommandozeilen zu packen als zu wenig.

Obwohl nur der ‚`src`'-Tree soweit gezeigt wurde, wirst du die gleichen Schritte für ‚`XF4`' und ‚`ports`' durchführen. Da alle Teile von OpenBSD synchron gehalten werden müssen, sollten alle Trees, die du verwendest, zur gleichen Zeit einem ‚checkout' und Update unterzogen werden. Du kannst die ‚checkouts' in einer Zeile kombinieren (*-stable* gezeigt):

```
# cd /usr/src
# export CVSROOT=anoncvs@anoncvs.be.openbsd.org:/cvs
# cvs -d$CVSROOT checkout -rOPENBSD_3_6 -d src ports XF4
```

Jedoch müssen Updates Verzeichnis für Verzeichnis durchgeführt werden.

Zu diesem Zeitpunkt, ob du nun *-stable* oder *-current* folgst, solltest du einen einsetzbaren Source-Tree haben. Sei vorsichtig damit, welchen Tree du beziehst -- es ist einfach zu versuchen, *-current* zu kompilieren, obwohl du auf *-stable* aus bist.

Allgemeine CVS-Tipps

Wie zuvor schon beschrieben, gibt es einige Optionen, die notwendig sind, um einen gültigen `src`-Tree in OpenBSD zu erhalten. Die obige ‚`-P`'-Option ist eine solche: Es ‚prunes' (entfernt) Verzeichnisse, die leer sind. Über die Jahre wurden Verzeichnisse im OpenBSD-Source-Tree erstellt und gelöscht, und ab und zu werden die Namen von alten Verzeichnissen als Dateinamen genutzt. Ohne dieser ‚`-P`'-Option wird dein Tree NICHT erfolgreich kompilieren.

Größtenteils gilt das gleiche für die `-d`-Option beim ‚update'-Kommando -- es erstellt neue Verzeichnisse, die eventuell zum Tree seit deinem ersten ‚checkout' hinzugefügt worden sein.

Erfahrene CVS-Anwender wundern sich vielleicht, warum CVSROOT angegeben und in diesem Beispiel genutzt wurde, obwohl cvs(1) sich die Angaben über den Server vom Tree merkt, auf den ein ‚checkout' ausgeführt wurde. Das ist korrekt, aber trotzdem gibt es Momente, in denen man den standardmäßigen Anoncvs-Server überschreiben muss, viele Leute empfehlen daher *immer* das explizite Angeben vom Repository. Es ist an dieser Stelle ebenfalls erwähnenswert, dass, obwohl die CVSROOT-Umgebungsvariable direkt von cvs(1) genutzt werden kann, sie nur genutzt wird, wenn nichts anderes sie überschreibt (z.B. würde cvs(1) einen Fehler ohne dieser haben), wogegen das Angeben in der cvs(1)-Kommandozeile alle anderen Werte überschreibt.

Denke daran, dass in diesem Beispiel ein `-Pd` als Parameter für das `up`-Kommando genutzt wurde. Dies ist eine weitere dieser BENÖTIGTEN Optionen, welche es erlaubt, dass neue Verzeichnisse, die bisher nicht in deinem vorherigen ‚checkout' existieren, während dem Update-Prozess erstellt werden.

Es ist oft nützlich, eine `.cvsrc` in deinem Heimatverzeichnis zu verwenden, um Standardwerte für einige dieser Optionen anzugeben. Ein Beispiel einer solchen `.cvsrc`-Datei:

```
$ more ~/.cvsrc
cvs -q -danoncvs@anoncvs.be.openbsd.org:/cvs
diff -up
update -Pd
checkout -P
```

Diese Datei veranlasst cvs(1) dazu, den `anoncvs@anoncvs.be.openbsd.org:/cvs`-Server zu verwenden, normalerweise unnötige Ausgaben aller Operationen zu unterdrücken (‚`-q`' heißt ‚quiet' (ruhig)), dass ein ‚cvs up'-Kommando standardmäßig `-Pd` nutzt, ein ‚cvs diff' standardmäßig ‚unified diffs' wegen dem ‚`-u`' bereitstellt und dass ein ‚cvs checkout' die ‚`-P`'-Option verwenden wird. Obwohl diese Datei praktisch ist, wirst du Probleme haben, wenn du vergisst, dass diese Datei existiert oder du versuchst, Kommandos auf einer Maschine auszuführen, auf der diese Datei nicht existiert.

5.3.4 - Einen Kernel erzeugen

Wir nehmen an, dass du einen standardmäßigen (GENERIC oder GENERIC.MP) Kernel an dieser Stelle erzeugen möchtest. Normalerweise ist es genau das, was du machen möchtest. Ziehe *nicht* in Betracht, einen angepasten Kernel zu erzeugen, wenn du noch nicht über den standardmäßigen Erzeugungsprozess hinaus bist.

Es ist offensichtlich, dass der Kernel eine SEHR hardwareabhängige Komponente des Systems ist. Der Source für den Kernel ist in dem `/usr/src/sys`-Verzeichnis. Einige Teile des OpenBSD-Kernel-Codes werden auf allen Plattformen verwendet, andere sind sehr spezifisch für einen Prozessor oder eine Architektur. Wenn du in das `/usr/src/sys/arch/`-Verzeichnis guckst, wirst du eventuelle Dinge sehen, die etwas verwirrend wirken -- zum Beispiel gibt es dort `mac68k-`, `m68k-` und `mvme68k`-Verzeichnisse. In diesem Fall verwendet sowohl die `mvme68k-` als auch die `mac68k`-Systeme den gleichen Prozessor, aber die Maschinen, auf denen sie basieren, sind sehr unterschiedlich, und benötigen daher einen sehr unterschiedlichen Kernel (zu einem Computer-Design gehört mehr als nur der Prozessor!). Jedoch gibt es auch Teile des Kernels die allgemein sind, solche Teile werden in dem `m68k`-Verzeichnis aufbewahrt. Wenn du einfach einen Kernel erzeugst, musst du dir über Basis-Architektur-Verzeichnisse wie `m68k` keine Gedanken machen, du wirst ausnahmslos in den ‚compound'-Architektur-Verzeichnissen arbeiten, zum Beispiel `mvme68k`.

Kernel werden basierend auf den Kernel-Konfigurationsdateien erzeugt, welche sich in dem `/usr/src/sys/arch/`*<deine Plattform>*`/conf`-Verzeichnis befinden. Das Erzeugen des Kernels besteht aus dem Verwenden des `config(8)`-Programms, um eine Kernel-Compilier-Verzeichnis zu erstellen und einzurichten, welches dann `/usr/src/sys/arch/`*<deine Plattform>*`/compile/`*<KernelName>* genannt wird. Für dieses Beispiel nehmen wir an, dass du die i386-Plattform verwendest.

```
# cd /usr/src/sys/arch/i386/conf
# config GENERIC
# cd ../compile/GENERIC
# make clean && make depend && make
    [...jedemenge Ausgabe...]
# make install
```

Ersetze ‚i386' in der ersten Zeile mit deinem Maschinennamen. Das `machine(1)`-Kommando kann dir sagen, was dein momentaner Maschinenname ist, so dass eine offensichtliche Generalisierung das Nutzen des Kommandos ‚`cd /usr/src/sys/arch/`machine`/conf`' statt der ersten Zeile ist.

Starte deine Maschine nun neu, um den neuen Kernel zu aktivieren. Bedenke, dass der neue Kernel vor dem nächsten Schritt laufen sollte, wobei es eventuell egal sein kann, wenn du den oben angegeben Ratschlag, auf die am nähesten liegende Binary zu aktualisieren, befolgt hast. Manchmal jedoch ändern sich die APIs und der alte Kernel wird nicht in der Lage sein, neue Applikationen auszuführen, aber der neue Kernel wird normalerweise die alten unterstützen.

Variationen zu dem obigen Prozess: nur lesbarer Source-Tree

Manchmal möchtest du vielleicht sicherstellen, dass dein /usr/src/sys-Verzeichnis nicht modifiziert wird. Dies kann durch das Verwenden des folgenden Prozesses realisiert werden:

```
$ cd /irgendwo
$ cp /usr/src/sys/arch/i386/conf/GENERIC .
$ config -s /usr/src/sys -b . GENERIC
$ make clean && make depend && make
   ... jedemenge Ausgaben ...
```

Denke daran, dass du einen Kernel ohne root-Zugriff erzeugen kannst, aber du musst root sein, um den Kernel zu installieren.

5.3.5 - Das ‚userland' erzeugen

Es existiert ein bestimmter Prozess, der von OpenBSD verwendet wird. Prozesse, die unter anderen Betriebssystemen, mit denen du vertraut bist, genutzt werden, werden vermutlich nicht unter OpenBSD funktionieren, und du wirst ausgelacht werden, wenn du fragst, warum.

- Leere dein /usr/obj-Verzeichnis und erstelle die symbolischen Links neu:

  ```
  # rm -rf /usr/obj/*
  # cd /usr/src
  # make obj
  ```

  Bedenke, dass die Verwendung vom /usr/obj-Verzeichnis notwendig ist. Wenn dieser Schritt vor dem Erzeugen vom Rest des Trees fehlschlägt, ist es wahrscheinlich, dass der restliche Tree deinen src-Tree in schlechter Verfassung belassen wird.
- Stelle sicher, dass alle passenden Verzeichnisse erstellt werden.

  ```
  # cd /usr/src/etc && env DESTDIR=/ make distrib-dirs
  ```
- Erzeuge das System:

  ```
  # cd /usr/src
  # make build
  ```

  Dies compiliert und installiert die ‚userland'-Utilities in der passenden Reihenfolge. Dieser Schritt nimmt recht viel Zeit in Anspruch -- eine sehr schnelle Maschine kann ihn unter einer Stunde abschließen, eine sehr langsame Maschine benötigt vielleicht mehrere Tage. Wenn dieser Schritt abgeschlosen ist, hast du neu compilierte Binaries auf dein System installiert.
- **Wenn *-current* erzeugt wird:** Aktualisiere /dev und /etc mit den Änderungen, die in `current.html` aufgelistet sind. Wenn -stable nach einem ordentlichen Upgradeprozess oder einer Installation der ordentlich startenden Binary verfolgt wird, ist dieser Schritt weder notwendig noch erwünscht.

5.4 - Ein Release erstellen

Was ist ein ‚Release' und warum würde ich eines erstellen wollen?

Ein Release ist ein kompletter Satz an Dateien, die verwendet werden können, um OpenBSD auf einem anderen Rechner zu installieren. Wenn du nur einen Computer hast, auf dem OpenBSD läuft, hast du wirklich keinen Grund, ein Release zu erstellen, da der obige Erzeugungsprozess alles macht, was du brauchst. Eine Beispielanwendung für den Releaseprozess wäre das Erzeugen von *-stable* auf einer schnellen Maschine und dann das Erstellen von einem Release, das auf all deinen Maschinen in deinem Büro installiert werden kann.

Der Releaseprozess verwendet die Binaries, während dem obigen Erzeugungsprozess in dem Verzeichnis /usr/obj erstellt wurden, so dass du diesen Schritt erfolreich abschließen musst, und nichts das /usr/obj-Verzeichnis beeinflusst. Eine Situation, in der das ein Problem sein könnte, ist, wenn du eine memory disk für dein /usr/obj verwendest, um ein wenig Extraleistung während dem Erzeugungsprozess, daher wirst du sicherlich nicht dein System zwischen dem ‚Erzeugungs'- und dem ‚Release'-Schritt neustarten wollen!

Der Releaseprozess erfordert zwei Arbeitsverzeichnisse, welche wir DESTDIR und RELEASEDIR nennen werden. Alle Dateien, die Teil einer ‚sauberen' OpenBSD-Installation sind, werden an ihre richtige Stelle im DESTDIR kopiert. Sie werden dann getar(1)t und in das RELEASEDIR gelegt. Am Ende des Prozesses wird RELEASEDIR ein fertiges OpenBSD-Release enthalten. Der Releaseprozess wird ebenfalls /mnt verwenden, so dass dieser Ort für keine anderen Dinge während

dem Releaseprozess genutzt werden sollte. Als Beispiel werden wir nun DESTDIR mit dem Wert /usr/dest und RELEASEDIR mit /usr/rel verwenden.

Der Releaseprozess bezieht einige Utilities mit ein, welche nicht Teil des Basis-OpenBSD-Systems, crunch und crunchgen(), welche genutzt werden, um eine einzelne ausführbare Datei aus mehreren individuellen Binaries zu erstellen. Der Name, mit der diese einzelne Binary aufgerufen wird, entscheidet, welche Komponenten-Binary ausgeführt wird. Auf diese Weise werden individuelle Programme in den Ramdisk-Kernel gequetscht, der auf Bootdisketten und anderen Bootmedien vorliegt. *Diese Utilities müssen vor dem Beginn des Releaseprozesses erstellt werden.* Sie müssen nur ein einziges Mal erzeugt und installiert werden, aber Leute vergessen diesen Schritt häufig, und diese Programme werden schnell erzeugt, so dass einige Leute dazu tendieren, crunch und crunchgen einfach jedes Mal als Teil des Skriptes zu erzeugen, das sie zum Erstellen eines Releases nutzen.

Du musst root-Privilegien haben, um ein Release zu erstellen.

Ein Release erstellen

Zu aller erst, falls es auf dieser Maschine noch nicht geschehen ist, erzeuge crunch und crunchgen:

```
# cd /usr/src/distrib/crunch && make obj depend all install
```

Nun definieren wir unsere DESTDIR- und RELEASEDIR-Umgebungsvariablen:

```
# export DESTDIR=/usr/dest
# export RELEASEDIR=/usr/rel
```

Wir leeren das DESTDIR und erstellen die Verzeichnisse, wenn das notwendig ist:

```
# test -d ${DESTDIR} && mv ${DESTDIR} ${DESTDIR}.old && rm -rf ${DESTDIR}.old &
# mkdir -p ${DESTDIR} ${RELEASEDIR}
```

RELEASEDIR muss normalerweise nicht vor dem Beginn des Releaseprozesses leer sein, jedoch könnten alte Dateien rumliegen bleiben, wenn sich Releasedateien oder ihre Namen geändert haben. Eventuell möchtest du das Verzeichnis vor dem Beginn ebenfalls löschen.

Wir erstellen nun das eigentliche Release:

```
# cd /usr/src/etc
# make release
```

Es ist eine gute Idee, nachdem das Release erstellt wurde, das Release zu überprüfen, ob die Tar-Dateien mit dem übereinstimmen, was im DESTDIR-Verzeichnis liegt. Die Ausgabe von diesem Schritt sollte sehr gering sein.

```
# cd /usr/src/distrib/sets
# sh checkflist
```

Du hast nun ein vollständiges und überprüftes Dateisets in dem RELEASEDIR. Diese Dateien können nun verwendet werden, um OpenBSD auf anderen Maschinen zu installieren oder zu aktualisieren.

Die maßgeblichen Anweisungen, wie man ein Release erstellt, sind in release(8).

5.5 - X erzeugen

X verwendet einen anderen Erzeugungsprozess als der Rest vom OpenBSD-Trees, da dieser auf imake- und nicht auf dem standardmäßigen make(1)-Prozess basiert. Eine Konsequenz hieraus ist, dass es kein ‚obj'-Verzeichnis gibt, generierte Binaries werden unter den Source-Code gemixt, was zu Problemen führen kann (oder zumindest zu sehr viel Ausgabe) mit cvs(1). Eine Lösung für dieses Problem ist, lndir(1) zu verwenden, um ein ‚Schattenverzeichnis' (shadow directory) zu erstellen, das symbolische Verweise zum tatsächlichen Source-Verzeichnis für den XF4-Tree beinhaltet.

nur i386: XF86Setup, das verwendet wird, um XF3-Server auf der i386-Plattform (und zwar NUR der i386-Plattform) zu konfigurieren, benötigt das ‚tcl'- und das ‚tk'-Package, die beide vor dem Erzeugen von X installiert sein müssen (können im Ports-Tree unter /usr/ports/lang/tcl/8.4/ und /usr/ports/x11/tk/8.4/ gefunden werden. Das ‚tk'-Package wird als Abhängigkeit von ‚tcl' installiert). Wie sonst auch ist das Installieren eines Packages schneller als das Installieren dieser Applikationen vom Source aus. Das Fehlschlagen beim Erzeugen dieser Packages vor dem Erzeugen von X ist recht frustrierend, da das System für einzige Zeit laufen wird, bevor es wegen einem Fehler aufhört.

Um X unter Verwendung des ‚Schattenverzeichnisses' /usr/Xbld zu kompilieren, verwende die folgenden Schritte und um neue Binaries in die passenden Verzeichnisse zu kopieren, folge diesen Schritten:

```
# rm -rf /usr/Xbld
# mkdir -p /usr/Xbld
# cd /usr/Xbld
# lndir ../XF4
   [...jede Menge Ausgabe...]
# make build
   [...jede Menge Ausgabe...]
```

Ein X-Release erstellen

Dies ist dem Hauptsystem-Releaseprozess sehr ähnlich. Nach dem erfolgreichen Erzeugen von X wirst du DESTDIR und RELEASEDIR aus dem gleichen Grund wie weiter oben erzeugen. RELEASEDIR kann das gleiche Verzeichnis sein wie das Hauptsystem-RELEASEDIR, aber DESTDIR wird in diesem Prozess gelöscht und wieder erstellt. Wenn es vorsichtig angestellt wird, ist dies kein Problem, aber separate DESTDIRs zu verwenden, kann ‚sicherer' sein.

Für dieses Beispiel werden wir DESTDIR und RELEASEDIR jeweils mit den Werten /usr/Xbld/dest und /usr/Xbld/rel nutzen. Dies muss vor dem obigen Erzeugungsprozess gemacht werden.

```
# export DESTDIR=/usr/Xbld/dest
# export RELEASEDIR=/usr/Xbld/rel
# cd /usr/Xbld
# rm -rf dest
# mkdir dest rel
# make release
```

Falls du vor hast, sowohl einen ‚build' und ein Release von X zu erstellen, kannst du ein anderes make(1)-‚target', ‚b-r', nutzen, welches dann zuerst die ‚build'- und danach die Release-Schritte durchläuft. Das ‚b-r'-‚target' nimmt an, dass DESTDIR und RELEASEDIR die Unterverzeichnisse ‚rel' und ‚dest' sind, die unter deinem ‚build'-Unterverzeichnis liegen:

```
# rm -rf /usr/Xbld
# mkdir -p /usr/Xbld /usr/Xbld/dest /usr/Xbld/rel
# cd Xbld
# lndir ../XF4
   [...jede Menge AUsgabe...]
# make b-r
   [...jede Menge Ausgabe...]
```

5.6 - Warum brauche ich einen selbsterstellten Kernel?

Eigentlich brauchst du ihn vermutlich nicht.

Ein angepasster Kernel ist ein Kernel, der mit einer anderen Konfigurationsdatei als der bereitgestellten GENERIC Konfigurationsdatei erstellt wurde. Ein angepasster Kernel kann auf *-release, -stable oder -current* Code basieren, genauso wie der GENERIC es tun kann. Während das Kompilieren deines eigenen GENERIC Kernels vom OpenBSD Team unterstützt ist, ist es das Kompilieren eines selbst angepassten Kernels *nicht*.

Die standardmäßige OpenBSD Kernelkonfiguration (GENERIC) ist dafür ausgerichtet, für die meisten Leute einsetzbar zu sein. Mehrere Leute haben beim Versuch, das System zu optimieren, ihr System zerschossen, statt die Systemleistung zu verbessern. Es gibt Leute, die meinen, dass du deinen Kernel anpassen musst, um die optimale Leistung zu erhalten, doch das stimmt nicht für OpenBSD. Nur die fortgeschrittensten und erfahrensten Anwender mit den anspruchsvollsten Applikationen müssen sich Gedanken über einen angepassten Kernel oder ein angepasstes System machen.

Einige Gründe warum du vielleicht einen angepassten Kernel erzeugen möchtest:

- Du weißt tatsächlich was du tust und möchtest OpenBSD für einen Computer zurecht schneidern, der nur über sehr wenig RAM verfügt, indem du Gerätetreiber entfernst, die du nicht benötigst.
- Du weißt tatsächlich was du tust und möchtest standardmäßig aktivierte Optionen oder Optionen, die (aus gutem Grund) normalerweise nicht aktiviert sind, aktivieren.
- Du weißt tatsächlich was du tust und möchtest experimentelle Optionen aktivieren.
- Du weißt tatsächlich was du tust, eine spezielle Verwendung hast, die nicht von GENERIC bewerkstelligt werden könnte, und nicht nachfragen wirst, warum

es nicht funktioniert, wenn etwas schief läuft.

Einige Gründe, warum du keinen eigenen angepassten Kernel erzeugen solltest:

- Im Normalfall musst du es nicht.
- Du wirst kein schnelleres System erhalten.
- Du wirst wahrscheinlich ein weniger vertrauenswürdiges System erzeugen.
- Du wirst keine Unterstützung der Entwickler erhalten.
- Es wird von dir erwartet, dass du jegliches Problem mit einem GENERIC Kernel reproduzieren kannst, bevor Entwickler irgendeine Fehlermeldung ernst nehmen.
- Anwender und Entwickler werden dich auslachen, wenn du dein System zerschießt.
- Angepasste Kompiler Optionen sind normalerweise besser darin, Kompiler Probleme hervorzurufen, statt die Systemgeschwindigkeit zu erhöhen.

Das Entfernen von Gerätetreibern mag den Bootprozess von deinem System beschleunigen, aber wird den Wiederherstellungsprozess erschweren, wenn du ein Problem mit der Hardware hast, und wird meistens falsch durchgeführt. Das Entfernen der Geräte Treiber *wird nicht* dazu beitragen, dass dein System spürbar schneller wird, obwohl es einen kleineren Kernel erzeugt. ‚Debugging' Informationen und Fehlerüberprüfungsroutinen zu entfernen kann das System spürbar beschleunigen, doch wird es die Untersuchung eines Systems unmöglich machen, wenn etwas schief geht.

Es sei an dieser Stelle noch einmal erwähnt, dass Entwickler normalerweise Meldungen von Fehlern ignorieren, außer wenn sie ebenfalls mit einem GENERIC Kernel hervorgerufen werden können. Du wurdest gewarnt.

5.7 - Einen angepassten Kernel erzeugen

Es wird angenommen, dass du das vorherige gelesen hast und wirklich auf Schmerzen stehst. Es wird des weiteren angenommen, dass du ein Ziel hast, dass man weder durch das Nutzen einer Bootzeit-Konfiguration (UKC>) noch durch config(8)urieren eines GENERIC-Kernels erreicht werden kann. Wenn diesen beiden Annahmen falsch sind, solltest du beim Verwenden von GENERIC bleiben. Wirklich.

Die OpenBSD Kernelgenerierung wird über Konfigurationsdateien verwaltet, die sich standardmäßig im /usr/src/sys/arch/<*arch*>/conf/ Verzeichnis befinden. Alle Architekturen haben eine Datei, GENERIC, die verwendet wird, um den Standard OpenBSD Kernel für diese Architektur zu erzeugen. Dort können sich ebenfalls andere Konfigurationsdateien befinden, die Kernel erzeugen, die andere Ziele verfolgen, zum Beispiel minimalen RAM, ‚diskless workstations', etc.

Die Konfigurationsdatei wird von config(8) verarbeitet, das ein Kompilationsverzeichnis in ../compile erstellt und einrichtet; bei einer typischen Installation wäre es in /usr/src/sys/arch/<*arch*>/compile/. config(8) erstellt ebenfalls eine Makefile und weitere Dateien, die für eine erfolgreiche Erstellung des Kernels notwendig sind.

Kernelkonfigurationsoptionen bieten dir die Möglichkeit, zusätzliche Unterstützung zu deinem Kernel hinzuzufügen. Dies erlaubt dir nur die von dir gewünschten Geräte zu unterstützen. Es gibt eine Vielzahl an Möglichkeiten, deinen Kernel auf deine Wünsche abzustimmen. Hier werden wir nur auf einige der am häufigsten benutzten Möglichkeiten eingehen. Siehe die options(4) Manual Seite für eine komplette Liste der Optionen und, da diese sich von Zeit zu Zeit ändert, solltest du sicherstellen, dass du auch eine Manual Seite für die gleiche Version von OpenBSD verwendest, das du erzeugst. Du kannst auch die Beispiel-Konfigurationsdateien zu Rate ziehen, die für deine Architektur zur Verfügung stehen.

Ändere, entferne oder füge niemals Optionen hinzu, wenn du keinen triftigen Grund dazu hast! Editiere nicht die GENERIC-Konfigurationsdatei!! Die einzige Kernelkonfiguration, die vom OpenBSD-Team unterstützt wird, ist der GENERIC Kernel, die eine Kombination der Optionen in /usr/src/sys/arch/<*arch*>/conf/GENERIC und /usr/src/sys/conf/GENERIC ist, *so, wie sie vom OpenBSD Team bereit gestellt worden sind* (d.h. NICHT verändert). Das Melden eines Fehlers, der mit einem modifizierten Kernel zustande kam, resultiert meistens darin, dass dir gesagt wird, dass du versuchen sollst, das Problem mit einem GENERIC Kernel zu reproduzieren. Nicht alle Optionen sind kompatibel untereinander und viele Optionen sind notwendig, damit das System läuft. Es besteht keine Garantie, dass ein Kernel läuft, nur weil du ihn kompilieren konntest. Es besteht außerdem keine Garantie, dass ein Kernel, der ‚config(1)uriert' werden kann, auch erzeugt werden kann.

Hier kannst du die plattform-spezifischen Konfigurationsdateien sehen:

- alpha Kernel Konfgurationsdateien
- i386 Kernel Konfigurationsdateien
- macppc Kernel Konfigurationsdateien
- sparc Kernel Konfigurationsdateien
- sparc64 Kernel Konfigurationsdateien
- vax Kernel Konfigurationsdateien
- hppa Kernel Konfigurationsdateien
- Andere Architekturen

Wenn du diese Dateien genauer betrachtest, dann wird dir am Anfang eine Zeile wie diese auffallen:

```
include "../../../conf/GENERIC"
```

Dies bedeutet, dass ein Verweis auf eine andere Konfigurationsdatei vorhanden ist, eine, die plattform-unabhängige Optionen beinhaltet. Wenn du also deinen eigenen Kernel konfigurieren willst, stelle sicher, dass du auch /sys/conf/GENERIC beachtet hast. Dort **sind** Optionen enthalten, die **benötigt** werden.

Kernelkonfigurationsoptionen sollten in deiner Kernelkonfigurationsdatei im Format von:

```
option      Name
```

oder +

```
+option      Name=Wert
+
```

angeführt sein. Um beispielsweise die ‚DEBUG'-Option in den Kernel zu bringen, füge eine Zeile wie die folgende ein:

```
option      DEBUG
```

Die Optionen im OpenBSD Kernel werden in Kompilerpräprozessoroptionen übersetzt, daher würde eine Option wie DEBUG den Quelltext mit der Option -DDEBUG kompilieren, was einem #define DEBUG im Kernel entspricht.

Ab und zu möchtest du vielleicht Optionen deaktivieren, die bereits definiert wurden, typischerweise in der "src/sys/conf/GENERIC" Datei. Obwohl du eine Kopie dieser Datei verändern könntest, ist das Verwenden der *rmoption* Angabe eine bessere Wahl. Falls du zum Beispiel den in-kernel Debugger deaktivieren möchtest (*nicht empfohlen!*), würdest du eine Zeile wie diese:

```
rmoption DDB
```

in deine Kernelkonfigurationsdatei hinzufügen. option DDB ist in src/sys/conf/GENERIC definiert, aber die oben angegebene rmoption Zeile deaktiviert sie wieder.

Noch einmal, bitte siehe options(4) für weitere Informationen über die Spezifikationen dieser Optionen. Bedenke auch, dass viele dieser Optionen ihre eigenen Manual Seiten haben -- lese immer alles, was über eine Option verfügbar ist, bevor du sie zu deinem Kernel hinzufügst oder aus ihm entfernst.

Einen angepassten Kernel erzeugen

In diesem Fall werden wir einen Kernel erzeugen, der die boca(4)-ISA-Multiport serielle Karte unterstützt. Diese Karte ist nicht im GENERIC-Kernel, da er mit anderen Treibern in Konflikt gerät. Eine anderer häufiger Grund, einen angepassten Kernel zu erzeugen, wäre RAIDframe, das zu groß ist, um in einem Serienkernel zu sein. Es gibt zwei typische Wege, einen angepassten Kernel zu erzeugen: die GENERIC-Konfigurationsdatei unter anderem Namen zu kopieren und zu editieren oder eine ‚wrapper'-Datei zu erstellen, die den Standard-GENERIC-Kernel ‚einbindet' und du alle Optionen, die du benötigst, die nicht in GENERIC sind. In diesem Fall sieht unsere Wrapper-Datei beispielsweise so aus:

```
include "arch/i386/conf/GENERIC"

boca0  at       isa? port 0x100 irq 10     # BOCA 8-port serial cards
pccom* at       boca? slave ?
```

Die zwei Zeilen bezüglich der boca(4)-Karte werden von den auskommentierten Zeilen in GENERIC kopiert und der IRQ nach eigenen Bedürfnissen angepasst. Der Vorteil der Nutzung dieser ‚wrapper'-Datei ist, dass alle anderen Änderungen in der GENERIC automatisch aktualisiert werden, ohne dass irgendein anderer Source-Code aktualisiert werden müsste. Der Nachteil ist, dass man keine Devices entfernen kann (obwohl das normalerweise sowieso eine schlechte Idee ist).

Ein anderer Weg, einen angepassten Kernel zu generieren, ist, eine Kopie der standardmäßigen GENERIC zu machen, ihr einen anderen Namen zu geben und dann nach eigenen Bedürfnissen anzupassen. Der Nachteil dabei ist, dass spätere Updates für die GENERIC-Konfigurationsdatei in deine Kopie übertragen werden müssen oder aber du musst deine Konfigurationsdatei erneut erstellen.

In beiden Fällen verwende, nachdem du deine angepasste Kernelkonfigurationsdatei erstellt hast, config(1) und erstelle den Kernel, wie es oben beschrieben wird.

Vollständige Instruktionen zum Erstellen deines eigenen angepassten Kernels sind in der afterboot(8)-Manualseite.

5.8 - Konfiguration zur Bootzeit

Manchmal findet der Kernel beim Booten dein Gerät, aber eventuell den falschen IRQ. Und vielleicht brauchst du dieses Gerät sofort. Nun, ohne den Kernel neu zu bauen, kannst du mit der OpenBSD eigenen Bootzeit Konfiguration dieses Problem lösen. Dies wird aber dein Problem nur einmal lösen. Wenn du rebootest, dann musst du diese Prozedur wiederholen. Daher ist dies nur als vorübergehende Lösung gedacht und du solltest das Problem mittels config(8) beheben. Dein Kernel wird dennoch **option BOOT_CONFIG** benötigen, die GENERIC bereits beinhaltet.

Den Großteil dieses Dokumentes kannst du in der Manual Seite boot_config(8) finden.

Um in die Benutzerkernelkonfiguration (User Kernel Config) oder UKC zu gelangen, musst du die -c Option beim Hochfahren verwenden.

```
boot> boot hd0a:/bsd -c
```

Oder welchen Kernel du auch immer laden willst. So kommst du in die UKC. Hier kannst du Befehle ausführen, die kernelspezifische Geräte ändern oder deaktivieren.

Hier eine Liste der gängigen Befehle in der UKC.

- add **device** - Füge ein Gerät durch Kopieren eines anderen hinzu
- change **devno | device** - Ändere ein oder mehrere Geräte
- disable **devno | device** - Deaktiviere ein oder mehrere Geräte
- enable **devno | device** - Aktiviere ein oder mehrere Geräte
- find **devno | device** - Suche ein oder mehrere Geräte
- help - Kurze Zusammenfassung dieser Befehle
- list - Liste ALLE bekannten Geräte auf
- exit/quit - Setze das Starten fort
- show **[attr [val]]** - Zeige alle Geräte mit Eigenschaft (attr) und optional mit einem spezifierten Wert (val)

Wenn du einmal deinen Kernel konfiguriert hast, dann steige mit quit oder exit aus UKC aus und setze das Starten fort. Danach solltest du deine Änderung am Kernel-Image dauerhaft machen, indem du die Schritte gemäß Mittels config(8) deinen Kernel verändern ausführst.

5.9 - Mittels config(8) deinen Kernel verändern

Die **-e** und **-u** Optionen von config(8) können sehr hilfreich sein und Zeit sparen, die du mit dem Neukompilieren von Kerneln verschwenden würdest. Die **-e** Option erlaubt dir die UKC auf einem laufenden System zu benutzen. Die Änderungen werden dann beim nächsten Reboot wirksam. Die **-u** Option testet, ob irgendwelche Änderungen am laufenden Kernel während des Bootens gemacht wurden. D.h., ob du mittels **boot -c** die UKC beim Starten benutzt hast.

Das folgende Beispiel zeigt das Deaktivieren des ep* Gerätes im Kernel. Zur Sicherheit musst du die **-o** Option benutzen, die die Änderung in die angegebene Datei schreibt. Z.B.: **config -e -o bsd.new /bsd** wird die Änderungen in bsd.new schreiben. Das folgende Beispiel verwendet die **-o** Option nicht, daher werden die Änderungen einfach ignoriert und nicht in eine Kernelbinärdatei geschrieben. Für weitere Informationen über Fehler- und Warnmeldungen siehe die config(8) Manual Seite.

```
$ sudo config -e /bsd
OpenBSD 3.6 (GENERIC) #59: Fri Sep 17 12:32:57 MDT 2004
    deraadt@i386.openbsd.org:/usr/src/sys/arch/i386/compile/GENERIC
warning: no output file specified
Enter 'help' für information
ukc> ?
        help                            Command help list
        add         dev                 Add a device
        base        8|10|16             Base on large numbers
        change      devno|dev           Change device
        disable     attr val|devno|dev  Disable device
        enable      attr val|devno|dev  Enable device
        find        devno|dev           Find device
        list                            List configuration
        lines       count               # of lines per page
        show        [attr [val]]        Show attribute
        exit                            Exit, mitout saving changes
        quit                            Quit, saving current changes
        timezone    [mins [dst]]        Show/change timezone
        nmbclust    [number]            Show/change NMBCLUSTERS
        cachepct    [number]            Show/change BUFCACHEPERCENT
        nkmempg     [number]            Show/change NKMEMPAGES
        shmseg      [number]            Show/change SHMSEG
        shmmaxpgs   [number]            Show/change SHMMAXPGS
ukc> list
  0 audio* at
sb0|sb*|gus0|pas0|sp0|ess*|wss0|wss*|ym*|eap*|eso*|sv*|neo*|cmpci*|clcs*|clct*|auich*|autri*|auvia*|fms*|uaudio*|maestro*|esa*|yds*|emu*
flags 0x0
  1 midi* at sb0|sb*|opl*|opl*|opl*|opl*|ym*|mpu*|autri* flags 0x0
  2 nsphy* at
aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|ep*|ep*
phy -1 flags 0x0
  3 nsphyter* at
aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|ep*|ep*
phy -1 flags 0x0
  4 qsphy* at
aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|ep*|ep*
phy -1 flags 0x0
  5 inphy* at
aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|ep*|ep*
phy -1 flags 0x0
  6 iophy* at
aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|ep*|ep*
phy -1 flags 0x0
  7 eephy* at
aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|ep*|ep*
phy -1 flags 0x0
  8 exphy* at
aue*|xe*|ef*|gx*|stge*|bge*|nge*|sk*|ste*|sis*|sf*|wb*|tx*|tl*|vr*|ne0|ne1|ne2|ne*|ne*|ne*|dc*|dc*|rl*|fxp*|fxp*|xl*|xl*|ep0|ep0|ep0|ep*|ep*|ep*|ep*|ep*
phy -1 flags 0x0
[...snip...]
--- more ---
[snip]
ukc> disable ep
 67 ep0 disabled
 68 ep* disabled
 69 ep* disabled
155 ep0 disabled
156 ep0 disabled
157 ep* disabled
158 ep* disabled
210 ep* disabled
ukc> quit
not forced
```

Im obigen Beispiel werden alle ep* Geräte im Kernel deaktiviert und auch nicht abgefragt. In einigen Situationen, in denen du die UKC beim Booten mittels **boot -c** benutzt hast, willst du diese Änderungen dauerhaft niederschreiben. Dafür brauchst du die **-u** Option. Im folgenden Beispiel wurde die UKC gestartet und das wi(4) Gerät deaktiviert. Da diese Änderung mit boot -c NICHT dauerhaft ist, müssen die Änderungen erst geschrieben werden. Dieses Beispiel schreibt die Änderung in boot -c in eine neue Kernelbinärdatei namens bsd.new.

```
$ sudo config -e -u -o bsd.new /bsd
```

```
OpenBSD 3.6 (GENERIC) #59: Fri Sep 17 12:32:57 MDT 2004
    deraadt@i386.openbsd.org:/usr/src/sys/arch/i386/compile/GENERIC
Processing history...
105 wi* disabled
106 wi* disabled
Enter 'help' for information
ukc> quit
```

5.10 - Mehr ‚verbose' Nachrichten während dem Booten erhalten

Mehr ‚verbose' Nachrichten zu erhalten, kann sehr hilfreich sein, wenn man versucht, Probleme während des Bootens zu beheben. Wenn du ein Bootproblem hast, obwohl ein Diskettenboot keine hat, und mehr Informationen erhalten möchtest, starte einfach neu. Wenn du beim ‚boot&qt;'-Prompt angelangt bist, boot mit boot -c. Dies bring dich in den UKC>, wo du dann das machst:

```
UKC> verbose
autoconf verbose enabled
UKC> quit
```

Nun werden dir extrem viele ‚verbose' Nachrichten beim Booten gegeben.

5.11 - Häufige Probleme beim Kompilieren und Erzeugen des Systems

5.11.1 - Das Erzeugen bricht mit einem ‚Signal 11'-Fehler ab

OpenBSD und andere Programme vom Source aus erzeugen ist eine Aufgabe, die die Hardware stärker beansprucht als die meisten anderen, da intensive Verwendung der CPU, Festplatte und des Speichers vorliegt. Als Ergebnis, wenn du Hardware hast, die einen Fehler hat, ist der wahrscheinlichste Zeitpunkt, dass sich das Problem bemerkbar macht, beim Kompilieren. Signal 11 Fehler sind *typischerweise* durch Hardware Probleme verursacht, sehr häufig Speicherprobleme, können aber auch CPU-, Mainboard- oder Hitzeprobleme sein. Dein System mag ansonsten sogar sehr stabil sein, aber nicht in der Lage, Programme zu kompilieren.

Du wirst es vermutlich am besten finden, wenn du die Komponente, die die Fehler verursacht, reparierst oder austauschst, da Probleme sich auf andere Art und Weise in Zukunft zeigen könnten. Falls du Hardware hast, die du wirklich verwenden willst und sie dir sonst keine weiteren Probleme bereitet, installiere einfach einen Snapshot oder ein Release.

Siehe Sig11 FAQ für sehr viel mehr Informationen.

5.11.2 - ‚make build' schlägt mit 'cannot open output file snake: is a directory' fehl

Dies ist das Resultat von zwei separaten Fehlern:

- **Du hast deinen CVS-Tree nicht ordentlich heruntergeladen oder aktualisiert.** Wenn du eine CVS checkout Operation durchführst, musst du die "-P" Option verwenden; wenn du deinen Source Tree mit CVS aktualisierst, musst du die "-Pd" Option mit cvs(1) verwenden, so wie es oben beschrieben steht. Diese Optionen stellen sicher, dass neue Verzeichnisse im Tree erstellt und entfernt werden, jenachdem wie sich OpenBSD entwickelt.
- **Du hast vor dem ‚build' das obj Verzeichnis nicht ordnungsgemäß erstellt.** Den Tree ohne einem /usr/obj Verzeichnis zu Erzeugen ist nicht unterstützt.

Es ist wichtig, den Instruktionen sorgfältig zu folgen, wenn du deinen Tree herunterlädst und erzeugst.

5.11.3 - Mein System ohne IPv6 läuft nicht!

Stimmt. Bitte mach keine Modifikationen am Basis-System, von denen du nicht weißt, wie sie sich auswirken. Eine ‚kleine' Änderung im Kernel kann große Auswirkungen für den gesamten Rest des Systems haben. Bitte lese das hier erneut.

5.11.4 - Hoppla! Ich habe vergessen, zuerst das `/usr/obj`-Verzeichnis zu erstellen!

Durch das Ausführen von ‚make build' vor ‚make obj' wirst du mit Objektdateien da stehen, die in deiem /usr/src-Verzeichnis herumliegen. Das ist eine schlechte Sache. Wenn du vermeiden willst, deinen gesamten Src-Tree erneut zu beziehen, kannst du folgendes versuchen, um die Obj-Dateien zu entfernen:

```
# cd /usr/src
# find . -type l -name obj | xargs rm
# make cleandir
# rm -rf /usr/obj/*
# make obj
```

5.11.5 - Tipp: Lege `/usr/obj` auf seine eigene Partition

Wenn du häuft erzeugst, wirst du es vielleicht als schneller empfinden, wenn du /usr/obj auf seine eigene Partition legst. Der Nutzen ist einfach, es ist typischerweise schneller:

```
# umount /usr/obj
# newfs DeineObjPartition
# mount /usr/obj
```

durchzuführen als ‚rm -rf /usr/obj/*'.

5.11.6 - Wie verhindere ich das Erzeugen von bestimmten Teilen des Trees?

Manchmal möchtest du das Erzeugen von bestimmten Teilen des Trees verhindern, normalerweise, weil du einen Ersatz für eine mitgelieferte Applikation durch Packages hast oder weil du ein ‚kleineres' Release aus welchem Grund auch immer haben willst. Die Lösung hierfür ist, die SKIPDIR-Option von /etc/mk.conf zu verwenden.

Hinweis: es ist möglich ein kaputtes (broken) System auf diesem Wege zu erzeugen. Die Resultate aus diesen Optionen wird vom OpenBSD-Projekt nicht unterstützt.

5.11.7 - Wo kann ich mehr über den Erzeugungsprozess erfahren?

Hier sind einige andere Ressourcen:

- release(8)
- afterboot(8)
- mk.conf(5)
- /usr/src/Makefile
- ‚Patch Branches' (*-stable*)
- (für X) /usr/X11R6/README auf deinem installierten System

5.11.8 - Ich sehe keine Snapshots auf den FTP-Seiten. Wo sind sie geblieben?

Snapshots können entfernt werden, wenn sie zu alt werden (oder nicht länger relevant sind) oder wenn ein neues *-release* naht.

Wie führe ich ein ‚bootstrap' für eine neue Version vom Compiler (*gcc*) aus?

Du solltest wirklich einfach den aktuellsten Snapshot installieren.

OpenBSD unterstützt nun zwei Compiler tree-intern, gcc v3.3.5, das von den meisten Plattformen genutzt wird, aber ebenfalls gcc v2.95.3, das von einigen wenigen Plattformen genutzt wird, die bisher noch nicht konvertiert wurden, oder aber niemals konvertiert werden, da Unterstützung vom gcc3 fehlt oder aber gcc3 eine schlechte Leistung erbringt.

Die zwei Compiler befinden sich an zwei unterschiedlichen Teilen des Trees:

- gcc3: /usr/src/gnu/usr.bin/gcc
- gcc2: /usr/src/gnu/egcs/gcc

Weil das Upgraden von einem Compiler in etwa ein Huhn-oder-Ei-Problem ist, benötigen Änderungen der tree-internen Compiler ein wenig Extra-Aufmerksamkeit. Du musst den Compiler zweimal erzeugen -- der erste ‚build' erzeugt einen Compiler, der neuen Code erzeugt, aber mit Code arbeitet, der vom alten Compiler erzeugt wurde, der zweilte ‚build' macht ihn vollständig zum einen Compiler. Generell wirst du wohl die folgende Prozedur durchführen:

Falls deine Plattform gcc 2.95.3 verwendet:

```
# rm -r /usr/obj/gnu/egcs/gcc/*
```

```
   # cd /usr/src/gnu/egcs/gcc
     - or -
Falls deine Plattform gcc 3.3.5 verwendet:
   # rm -r /usr/obj/gnu/usr.bin/gcc/*
   # cd /usr/src/gnu/usr.bin/gcc

Gemeinsame ‚build'-Prozedur für v3.3.5 und v2.95.3
# make -f Makefile.bsd-wrapper clean
# make -f Makefile.bsd-wrapper obj
# make -f Makefile.bsd-wrapper depend
# make -f Makefile.bsd-wrapper
# make -f Makefile.bsd-wrapper install
# make -f Makefile.bsd-wrapper clean
# make -f Makefile.bsd-wrapper depend
# make -f Makefile.bsd-wrapper
# make -f Makefile.bsd-wrapper install
```

Und führe anschließend ein normales make build durch.

Was ist der beste Weg um /etc, /var und /dev zu aktualisieren?

Als Richtlinie modifiziert Software im OpenBSD-Tree keine Dateien automatisch in /etc. Das bedeutet, dass es *immer* am Administrator liegt, die benötigten Modifikationen dort durchzuführen. Upgrades sind keine Ausnahme. Um Dateien in diesen Verzeichnissen zu aktualisieren, stelle erst einmal fest, welche Änderungen an den Basis- (Distributions) Dateien stattgefunden haben, und führe diese Änderungen erneut durch.

Um zum Beispiel die Dateien im Tree zu sehen, die zu letzt geändert wurden, führe dies aus:

```
# cd /usr/src/etc
# ls -lt |more
```

Um alle Änderungen in /etc zwischen zwei bestimmten Versionen von OpenBSD zu sehen, kannst du CVS verwenden. Um zum Beispiel die Änderungen zwischen 3.5 und 3.6 zu sehen, führe dies aus:

```
# cd /usr/src/etc
# cvs diff -u -rOPENBSD_3_5 -rOPENBSD_3_6
```

To see the changes between 3.6 and *-current* ("HEAD"), use:

```
# cd /usr/src/etc
# cvs diff -u -rOPENBSD_3_6 -rHEAD
```

Das /dev/MAKEDEV-Skript wird nicht automatisch als Teil des ‚make build'-Prozesses aktualisiert, jedoch wird es als Teil eines Binary updates installiert. Als generelle Regel sei zu sagen, dass es eine gute Idee ist, dieses Skript zu kopieren (falls das notwendig ist) und von deinem Source-Tree aus aufzurufen, um ein Upgrade durchzuführen:

```
# cd /dev
# cp /usr/src/etc/etc.`machine`/MAKEDEV ./
# ./MAKEDEV all
```

Sobald du die Änderungen ausgemacht hast, füge sie deinem lokalen Tree erneut zu, jegliche lokale Konfiguration, die du gemacht hast, bewahrend.

Typische /etc-Änderungen, auf die zwischen zwei Releases geachtet werden muss, sind unter anderem:

- Erweiterungen zu /etc/protocols und /etc/services
- Neue sysctls (siehe /etc/sysctl.conf)
- Änderungen an den standardmäßigen ‚cron jobs'. Siehe /etc/daily, /etc/weekly, /etc/monthly und /etc/security
- Alle rc-Skripte, einschließlich netstart
- Device-Änderungen, siehe weiter oben
- Datei-Hierarchie-Änderungen in /etc/mtree, siehe unten
- Neu User (/etc/passwd) und Gruppen (/etc/group)

Diese Änderungen werden in upgrade36.html (um zum 3.6-Release zu gelangen) oder current.html (um zu *-current* zu gelangen).

5.11.11 - Gibt es einen einfachen Weg, um alle Datei-Hierarchien zu ändern?

Von Zeit zu Zeit werden Dateien oder Verzeichnisse zu der Datei hierarchy hinzugefügt oder aus ihr entfernt. Ebenfalls können sich Besitzer-Informationen für Teile des Dateisystems ändern. Ein einfacher Weg, um sicherzustellen, dass deine Datei-Hierarchie auf dem neuesten Stand ist, ist das Verwenden von dem mtree(8)-Utility.

Beziehe zuerst den aktuellsten Source, dann führe folgendes aus:

```
# cd /usr/src/etc/mtree
# install -c -o root -g wheel -m 600 special /etc/mtree
# install -c -o root -g wheel -m 444 4.4BSD.dist /etc/mtree
# mtree -qdef /etc/mtree/4.4BSD.dist -p / -u
```

Deine Datei-Hierarchie sollte nun auf dem neuesten Stand sein.

[FAQ Index] [Zum Kapitel 4 - OpenBSD 3.5 Installationsanleitung] [Zum Kapitel 6 - Netzwerk]

---

www@openbsd.org
$OpenBSD: faq5.html,v 1.56 2005/03/12 17:55:51 jufi Exp $

OpenBSD

[FAQ Index] [Zum Kapitel 5 - Das System aus dem Source-Code erzeugen] [Zum Kapitel 7 - Tastatur- und Bildschirm-Kontrollen]

6 - Netzwerk

Inhaltsverzeichnis

6.1 - Bevor wir weiter gehen

Für den Rest dieses Dokumentes sei gesagt, dass es hilfreich ist, das Kapitel Kernelkonfiguration und Einstellungen der FAQ gelesen und zumindest teilweise verstanden zu haben, weiterhin helfen die ifconfig(8) und netstat(1) Manual Seiten.

Wenn du ein Netzwerkadministrator bist, Routingprotokolle aufsetzt und dein OpenBSD Rechner dein Router wird, dann solltest du dein Wissen über IP Netzwerke mit Understanding IP addressing vertiefen. Dies ist wirklich ein exzellentes Dokument. "Understanding IP addressing" beinhaltet grundlegendes Wissen, auf dem man bei der Arbeit mit IP Netzwerken aufbauen kann, insbesondere wenn man es mit mehreren Netzwerken zu tun hat oder für sie verantwortlich ist.

Wenn du mit Anwendungen wie Web-, FTP- oder Mailservern arbeitest, dann könntest du viel vom Lesen der entsprechenden RFCs profitieren. Natürlich kannst du nicht alle lesen. Aber dennoch, lies jene, die dich interessieren oder die du bei deiner Arbeit brauchen könntest. Lies nach, wie alles funktionieren sollte. Die RFCs definieren mehrere (tausend) Standards für Protokolle im Internet und wie sie arbeiten sollten.

6.2 - Erste Netzwerkeinstellungen

6.2.1 - Identifizieren und Einstellen deiner Netzwerkkarten

Um beginnen zu können, musst du zunächst deine Netzwerkkarte identifizieren können. Bei OpenBSD werden Netzwerkkarten nach ihrem Typ, nicht nach Verbindungsart benannt. Du kannst sehen, ob deine Netzwerkkarte initialisiert wurde, entweder schon beim Booten oder auch später mit Hilfe des Befehls dmesg(8). Weiterhin kannst du mit dem Befehl ifconfig(8) deine Karte überprüfen. Als Beispiel hier die Ausgabe von dmesg für eine Intel Fast Ethernet Netzwerkkarte, die als Gerätenamen fxp hat.

```
fxp0 at pci0 dev 10 function 0 "Intel 82557" rev 0x0c: irq 5, address
00:02:b3:2b:10:f7
inphy0 at fxp0 phy 1: i82555 10/100 media interface, rev. 4
```

Wenn du deinen Geräte-Namen nicht kennst, sieh bitte in der Liste der unterstützten Hardware für deine Plattform nach.

Du wirst eine Liste vieler bekannter Karten und ihre OpenBSD Geräte-Namen finden (wie etwa fxp), zusammen mit einer Nummer, die vom Kernel zugewiesen wird, und du hast den sogenannten Interface-Namen (wie z.B. fxp0).

Du kannst herausfinden, ob deine Netzwerkkarte(n) erkannt wurde(n), indem du das ifconfig(8) Kommando benutzt. Das folgende Kommando zeigt uns alle Netzwerk-Interfaces im System an. Diese Beispielausgabe zeigt ein physikalisches Ethernet Interface, ein fxp(4).

```
$ ifconfig -a
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 33224
        inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x5
lo1: flags=8008<LOOPBACK,MULTICAST> mtu 33224
fxp0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        address: 00:04:ac:dd:39:6a
        media: Ethernet autoselect (100baseTX full-duplex)
        status: active
        inet 10.0.0.38 netmask 0xffffff00 broadcast 10.0.0.255
        inet6 fe80::204:acff:fedd:396a%fxp0 prefixlen 64 scopeid 0x1
pflog0: flags=0<> mtu 33224
pfsync0: flags=0<> mtu 2020
sl0: flags=c010<POINTOPOINT,LINK2,MULTICAST> mtu 296
sl1: flags=c010<POINTOPOINT,LINK2,MULTICAST> mtu 296
ppp0: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
ppp1: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
tun0: flags=10<POINTOPOINT> mtu 3000
tun1: flags=10<POINTOPOINT> mtu 3000
enc0: flags=0<> mtu 1536
bridge0: flags=0<> mtu 1500
bridge1: flags=0<> mtu 1500
vlan0: flags=0<> mtu 1500
        address: 00:00:00:00:00:00
vlan1: flags=0<> mtu 1500
        address: 00:00:00:00:00:00
gre0: flags=9010<POINTOPOINT,LINK0,MULTICAST> mtu 1450
carp0: flags=0<> mtu 1500
carp1: flags=0<> mtu 1500
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
gif1: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
gif2: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
gif3: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
```

Wie du hier sehen kannst, gibt uns ifconfig(8) eine Menge mehr Informationen, als wir zu diesem Zeitpunkt benötigen. Natürlich sehen wir trotzdem unser Interface. Im obigen Beispiel ist die Netzwerkkarte bereits konfiguriert. Das ist offensichtlich, da auf fxp0 bereits ein IP Netzwerk konfiguriert ist, sprich die Werte "inet 10.0.0.38 netmask 0xffffff00 broadcast 10.0.0.255". Außerdem sind die **UP** und **RUNNING** Flags gesetzt.

Schlussendlich fällt auf, dass standardmäßig eine Menge mehr Interfaces aktiviert sind. Dies sind virtuelle Interfaces, die verschiedene Funktionen haben. Informationen dazu findest du auf den folgenden Manual Seiten:

- lo - Loopback Interface
- pflog - Packet Filter Logging Interface
- sl - SLIP Netzwerk Interface
- ppp - Point to Point Protokoll
- tun - Tunnel Network Interface
- enc - Encapsulating Interface
- bridge - Ethernet Bridge Interface
- vlan - IEEE 802.1Q Encapsulation Interface
- gre - GRE/MobileIP Encapsulation Interface
- gif - Generic IPv4/IPv6 Tunnel Interface

- carp - Common Address Redundancy Protocol Interface

Falls du deine Netzwerkkarte noch nicht konfiguriert hast, ist der erste Schritt das Erstellen der */etc/hostname.xxx* Datei, wobei der Name deiner Karte den Platz von xxx einnnimmt. Aus der Information der obigen Beispiele würde der Name */etc/hostname.fxp0* lauten. Das Layout dieser Datei ist simpel:

```
address_family address netmask broadcast [weitere Optionen]
```

(Viel mehr Details zu dieser Datei findest du in der hostname.if(5) Manual Seite.)

Eine typische Interface-Konfigurationsdatei für eine IPv4 Adresse würde so aussehen:

```
$ cat /etc/hostname.fxp0
inet 10.0.0.38 255.255.255.0 NONE
```

Du solltest auch den ‚media type' für Ethernet angeben, wenn du z.B. den 100baseTX full-duplex Modus erzwingen willst.

```
inet 10.0.0.38 255.255.255.0 NONE media 100baseTX mediaopt full-duplex
```

(Auf keinen Fall solltest du das tun, wenn nicht beide Seiten der Verbindungen auf Voll-Duplex gestellt sind! Wenn du keine besonderen Anforderungen hast, kannst du diese media Einstellungen einfach ignorieren.)

Oder vielleicht willst du auch spezielle Flags für ein einzelnes Interface benutzen. Das Format der Datei ändert sich dabei nicht besonders!

```
$ cat /etc/hostname.vlan0
inet 172.21.0.31 255.255.255.0 NONE vlan 2 vlandev fxp1
```

Der nächste Schritt ist das Einstellen deines Standard-Gateways (default gateway). Dazu trage einfach die IP deines Gateways in die Datei */etc/mygate* ein. Dies erlaubt das Aktivieren deines Gateways beim Starten. Jetzt solltest du deine Nameserver eintragen und die Datei */etc/hosts* einrichten. Für die Nameserver benötigst du eine Datei namens */etc/resolv.conf*. Mehr über das Format dieser Datei findest du in der resolv.conf(5) Manual Seite. Für den Normalgebrauch ist hier ein Beispiel, in dem deine Nameserver 125.2.3.4 und 125.2.3.5 sind. Du gehörst zur Domain "example.com".

```
$ cat /etc/resolv.conf
search example.com
nameserver 125.2.3.4
nameserver 125.2.3.5
lookup file bind
```

Jetzt kannst du entweder rebooten oder das **/etc/netstart** Skript ausführen, indem du (als root) folgendes eingibst:

```
# sh /etc/netstart
writing to routing socket: File exists
add net 127: gateway 127.0.0.1: File exists
writing to routing socket: File exists
add net 224.0.0.0: gateway 127.0.0.1: File exists
```

Dabei werden ein paar Fehlermeldungen ausgegeben. Indem du dieses Skript ausführst, versuchst du ein paar Sachen zu konfigurieren, die bereits konfiguriert sind. Daher existieren bereits einige der Routen in der Kernel Routing Tabelle. Von hier ab sollte dein System laufen und online sein. Du kannst hier erneut mit ifconfig(8) prüfen, ob deine Interfaces richtig konfiguriert wurden. Deine Routen kannst du via netstat(1) oder route(8) überprüfen. Wenn du Probleme mit dem Routing hast, möchtest du vielleicht das -n Flag für route(8) benutzen, das die IP-Adressen ausgibt, statt einen DNS Lookup zu machen, und um den Hostnamen anzuzeigen. Hier ist ein Beispiel mit beiden Kommandos, um die Routing Tabelle anzeigen zu lassen:

```
$ netstat -rn
Routing tables

Internet:
Destination        Gateway            Flags     Refs     Use    Mtu  Interface
default            10.0.0.1           UGS          0      86      -  fxp0
127/8              127.0.0.1          UGRS         0       0      -  lo0
```

```
127.0.0.1          127.0.0.1          UH          0        0      -  lo0
10.0.0/24          link#1             UC          0        0      -  fxp0
10.0.0.1           aa:0:4:0:81:d      UHL         1        0      -  fxp0
10.0.0.38          127.0.0.1          UGHS        0        0      -  lo0
224/4              127.0.0.1          URS         0        0      -  lo0

Encap:
Source             Port  Destination        Port  Proto SA(Address/SPI/Proto)
```

```
$ route show
Routing tables

Internet:
Destination      Gateway            Flags
default          10.0.0.1           UG
127.0.0.0        LOCALHOST          UG
localhost        LOCALHOST          UH
10.0.0.0         link#1             U
10.0.0.1         aa:0:4:0:81:d      UH
10.0.0.38        LOCALHOST          UGH
BASE-ADDRESS.MCA LOCALHOST          U
```

6.2.2 - Einrichten deines OpenBSD Rechners als Gateway

Dies sind nur die grundlegenden Informationen, um deinen OpenBSD Rechner als Gateway (auch Router genannt) einzurichten. Wenn du OpenBSD als Router im Internet verwenden willst, solltest du auch die unten folgenden Packet Filter Instruktionen beachten, um potentiell schädliche IP Daten zu blockieren. Auch solltest du wegen der Knappheit an IPv4 Adressen die Informationen bezüglich Network Address Translation beachten, um deinen IP Adressbereich zu schonen.

Der GENERIC Kernel hat bereits die Fähigkeit für IP Forwarding, aber dies muss erst eingeschaltet werden. Du solltest dies mit sysctl(8) tun. Um diese Änderung permanent einzutragen, musst du die Datei /etc/sysctl.conf editieren. Füge einfach folgende Zeile in diese Konfigurationsdatei ein.

```
net.inet.ip.forwarding=1
```

Ohne Reboot kannst du dies auch direkt mit sysctl(8) durchführen. Beachte aber, dass diese Änderung nach einem Reboot weg ist und dass der folgende Befehl als root ausgeführt werden muss.

```
# sysctl net.inet.ip.forwarding=1
net.inet.ip.forwarding: 0  -> 1
```

Nun modifiziere die Routen der anderen Hosts. Es gibt viele verschiedene Möglichkeiten, OpenBSD als Router einzusetzen, z.B. mittels Software wie das OpenBSD-eigene OpenBGPD, routed(8), mrtd, zebra und quagga. OpenBSD hat Unterstützung in der Ports-Kollektion sowohl für zebra als auch für quagga und mrtd. OpenBGPD und routed werden als Teil des Basissystems installiert. OpenBSD unterstützt mehrere T1, HSSI, ATM, FDDI, Ethernet und serielle (PPP/SLIP) Schnittstellen.

6.2.3 - Einrichten von Aliases auf deiner Netzwerkkarte

OpenBSD hat einen einfachen Mechanismus, um IP-Aliase für deine Netzwerkkarten zu setzen. Dazu musst du einfach die Datei */etc/hostname.<if>* editieren. Sie wird beim Booten vom /etc/netstart(8) Skript gelesen, das ein Teil der rc startup Hierarchie ist. Für dieses Beispiel nehmen wir an, dass der User ein Interface **dc0** hat und sich im Netzwerk 192.168.0.0 befindet. Weitere wichtige Informationen:

- IP für dc0 ist 192.168.0.2
- NETMASK ist 255.255.255.0

Ein paar Bemerkungen zu Aliasen: In OpenBSD verwendet man nur den Adapternamen. Es gibt keine Unterschiede zwischen dem ersten und dem zweiten Alias. Daher muss man sie nicht - wie in einigen anderen Betriebssystemen - als dc0:0, dc0:1 bezeichnen. Wenn du dich auf einen speziellen IP Alias beziehst oder einen hinzufügst, dann nimm "`ifconfig int alias`" anstelle nur "`ifconfig int`" auf der Befehlszeile. Du kannst Aliase mit "`ifconfig int delete`" löschen.

Angenommen du verwendest mehrere IP Adressen im selben IP Subnetz mit Aliases, dann ist die Netzmaskeneinstellung für jeden Alias 255.255.255.255. Sie müssen nicht der Netzmaske der ersten IP der Netzwerkkarte folgen. In diesem Beispiel */etc/hostname.dc0* werden zwei Aliase zur Netzwerkkarte dc0 hinzugefügt, die als 192.168.0.2 mit Netzmaske 255.255.255.0 konfiguriert wurde.

```
# cat /etc/hostname.dc0
inet 192.168.0.2 255.255.255.0 media 100baseTX
inet alias 192.168.0.3 255.255.255.255
inet alias 192.168.0.4 255.255.255.255
```

Wenn du einmal diese Datei erstellt hast, benötigst du einen Reboot, um die Änderung automatisch durchzuführen. Du kannst aber auch die Aliase manuell mit ifconfig(8) hochbringen. Für den ersten Alias geht das so:

```
# ifconfig dc0 inet alias 192.168.0.3 netmask 255.255.255.255
```

Um die Aliases zu sehen:

```
$ ifconfig -A
dc0: flags=8863<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST>
        media: Ethernet manual
        inet 192.168.0.2  netmask 0xffffff00 broadcast 192.168.0.255
        inet 192.168.0.3 netmask 0xffffffff broadcast 192.168.0.3
```

6.3 - Wie kann ich mit OpenBSD filtern und eine Firewall aufsetzen?

Packet Filter (ab hier nur noch als PF bezeichnet) ist OpenBSDs System zum Filtern von TCP/IP Verkehr und zum Ausführen von Network Address Translation. PF ist außerdem in der Lage, TCP/IP-Verkehr zu normalisieren und zu konditionieren, eine Priorisierung von Paketen durchzuführen und kann verwendet werden, um eine mächtige und flexible Firewall zu erzeugen. Er wird im **PF Benutzerhandbuch** beschrieben.

6.4 - Dynamic Host Configuration Protokoll (DHCP)

Das Dynamic Host Configuration Protokoll ist ein Weg, um die Netzwerkkarten "automatisch" zu konfigurieren. OpenBSD kann als DHCP Server (der andere Maschine konfiguriert), als ein DHCP Client (der von einer anderen Maschine konfiguriert wird) und in einigen Fällen auch als beides eingesetzt werden.

6.4.1 DHCP Client

Um den DHCP Client dhclient(8) zu benutzen, der Teil von OpenBSD ist, editiere `/etc/hostname.xl0` (wenn deine Hauptethernetkarte xl0 ist. Deine kann ep0 oder fxp0 oder irgendeine andere sein!). Alles, was du in dieser Datei zu schreiben hast, ist '`dhcp`'.

```
# echo dhcp >/etc/dhcpd.interfaces
```

Dies wird OpenBSD veranlassen, den DHCP Client automatisch beim Booten zu starten. OpenBSD wird sich seine IP Adresse, sein Standardgateway und seine DNS Server vom DHCP Server besorgen.

Wenn du den DHCP Client von der Befehlszeile aus starten willst, stelle sicher, dass `/etc/dhclient.conf` existiert, dann versuche:

```
# dhclient fxp0
```

Wobei `fxp0` die Netzwerkkarte ist, auf der du dhcp empfangen willst.

Wie du auch immer dhclient startest, kannst du die `/etc/dhclient.conf` Datei immer so editieren, dass dein DNS **nicht** erneuert wird aufgrund der neuen DNS Informationen, indem du die 'request' Zeilen auskommentierst (Es gibt Beispiele in den Standardeinstellungen, aber du musst die Standardeinstellungen von dhclient überschreiben).

```
request subnet-mask, broadcast-address, time-offset, routers,
      domain-name, domain-name-servers, host-name, lpr-servers, ntp-servers;
```

und dann **entferne** domain-name-servers. Natürlich kannst du auch hostname oder andere Einstellungen entfernen.

6.4.2 DHCP Server

Wenn du OpenBSD als DHCP Server dhcpd(8) einsetzen willst, editiere `/etc/rc.conf.local` so, dass sie die Zeile `dhcpd_flags=""` beinhaltet. Und die Netzwerkkarten, auf denen dhcpd(8) **lauschen** soll, stehen in `/etc/dhcpd.interfaces`.

```
# echo xl1 xl2 xl3 >/etc/dhcpd.interfaces
```

Dann editiere `/etc/dhcpd.conf`. Die Optionen sind selbsterklärend.

```
option  domain-name "example.com";
option  domain-name-servers 192.168.1.3, 192.168.1.5;

subnet 192.168.1.0 netmask 255.255.255.0 {
        option routers 192.168.1.1;

        range 192.168.1.32 192.168.1.127;
}
```

Dies teilt deinen DHCP Clients mit, dass die an DNS Anfragen anzuhängende Domäne example.com ist (d.h., wenn der Benutzer 'telnet joe' schreibt, dann wird an joe.example.com gesendet). Es wird auf die DNS Server 192.168.1.3 und 192.168.1.5 verwiesen. Für Hosts, die sich im selben Netzwerk wie die Netzwerkkarte des OpenBSD Rechners befinden, welche im 192.168.1.0/24 Adressbereich liegt, wird der DHCP Server ihnen eine IP Adresse zwischen 192.168.1.32 und 192.168.1.127 und als Standardgateway 192.168.1.1 zuweisen.

Wenn du den dhcpd(8) von der Befehlszeile aus starten willst, nachdem du `/etc/dhcpd.conf` editiert hast, versuche:

```
# touch /var/db/dhcpd.leases
# dhcpd fxp0
```

Die `touch` Zeile ist notwendig, um eine leere `dhcpd.leases` Datei zu erzeugen, bevor dhcpd(8) starten kann. Die OpenBSD Startup Skripte erstellen diese Datei beim Hochfahren, wenn es notwendig ist, aber wenn du dhcpd(8) manuell startest, musst du sie zuerst erstellen.

Wenn du DHCP Dienste für einen Windows Rechner bereitstellst, dann willst du vielleicht auch eine 'WINS' Serveradresse liefern. Dafür füge einfach die folgenden Zeilen zu deiner /etc/dhcpd.conf:

```
option    netbios-name-servers    192.168.92.55;
```

(wobei `192.168.92.55` die IP deines Windows oder Samba Servers ist.) Siehe auch dhcp-options(5) für weitere Optionen, die deine DHCP Clients wünschen.

6.5 - PPP

Das Point-to-Point Protokoll wird verwendet, um eine Verbindung zu deinem ISP mit deinem Einwahl-Modem herzustellen. OpenBSD bietet dafür 2 Möglichkeiten:

- pppd(8) - der Kernel PPP Daemon.
- ppp(8) - der Userland PPP Daemon.

Sowohl ppp als auch pppd führen zwar die gleichen Funktionen auf, dieses jedoch auf unterschiedliche Wege. pppd arbeitet mit dem ppp(4)-Treiber des Kernels, während ppp mit tun(4) im Userland arbeitet. Dieses Dokument wird sich nur mit dem PPP-Daemon des Userlands beschäftigen, da es mit ihm einfacher ist, Fehlfunktionen zu korrigieren sowie mit ihm zu interagieren. Um zu beginnen, benötigen wir einige einfache Informationen über deinen ISP. Hier eine Liste hilfreicher Informationen, die du brauchen wirst.

- Die Einwahlnummer deines ISPs
- Deinen Nameserver
- Deinen Benutzernamen und dein Passwort
- Dein Gateway

Einige von diesen benötigst du nicht unbedingt, aber sie wären beim Aufsetzen des ppp hilfreich. Der Userland PPP Daemon benutzt die Datei /etc/ppp/ppp.conf als seine Konfigurationsdatei. Es gibt viele hilfreiche Dateien in */etc/ppp*, die verschiedene Einstellungen für verschiedene Situationen zeigen. Du solltest dir dieses Verzeichnis ansehen und es durchforsten.

Erste Einstellungen - für PPP(8)

Die ersten Einstellungen für den Userland PPP Daemon bestehen im Erstellen deiner */etc/ppp/ppp.conf* Datei. Diese Datei existiert nicht standardmäßig, aber du kannst einfach */etc/ppp/ppp.conf.sample* editieren, um deine eigene *ppp.conf* Datei zu erstellen. Hier werde ich mit den einfachsten und gebräuchlichsten Einstellungen beginnen. Hier eine kurze *ppp.conf* Datei, die einfach einige Standardwerte setzt:

```
default:
set log Phase Chat LCP IPCP CCP tun command
set device /dev/cua01
set speed 115200
set dial "ABORT BUSY ABORT NO\\sCARRIER TIMEOUT 5 \"\" AT OK-AT-OK ATE1Q0 OK
\\dATDT\\T TIMEOUT 40 CONNECT"
```

Der Absatz unter der `default:` Bezeichnung wird jedes Mal ausgeführt. Hier stehen alle wichtigen Informationen. Mit "set log" stellen wir unsere Loglevel ein. Um dies zu ändern, siehe ppp(8) für weitere Info. Unsere Schnittstelle wird mit "set device" eingestellt. Dies ist die Schnittstelle, mit der das Modem verbunden ist. In diesem Beispiel hängt das Modem auf COM Port 2. Daher wird COM Port 1 auf */dev/cua00* gesetzt. Mit "set speed" setzen wir die Geschwindigkeit unserer Einwahl-Verbindung und mit "set dail" setzen wir unsere Dialup Parameter, mit denen wir den Timeout, usw. setzen können. Diese Zeile sollte eigentlich ziemlich genau so bleiben, wie sie jetzt ist.

Nun können wir die spezifischen Informationen von unserem ISP eintragen. Wir tun dies, indem wir unter `default:` einen weiteren Absatz hinzufügen. Dieser kann benannt werden, wie du willst - am einfachsten nimmst du den Namen deines ISP. Hier werde ich `myisp:` als Verweis auf unseren ISP nehmen. Hier ist ein einfaches Beispiel, das alles beinhaltet, um uns zu verbinden:

```
myisp:
set phone 1234567
set login "ABORT NO\\sCARRIER TIMEOUT 5 ogin:--ogin: ppp word: ppp"
set timeout 120
set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.0 0.0.0.0
add default HISADDR
enable dns
```

Hier stehen alle wichtigen Informationen für unseren spezifischen ISP. Die erste Option "set phone" setzt die Einwahlnummer deines ISPs. "set login" setzt unsere login-Optionen. Hier haben wir den Timeout auf 5 gesetzt, was bedeutet, dass wir unseren login-Versuch nach 5 Sekunden abbrechen, wenn wir kein Trägersignal bekommen. Ansonsten wird er auf "login:" warten und dann deinen Benutzernamen und dein Passwort senden.

In diesem Beispiel ist unser Benutzername = ppp und das Passwort = ppp. Diese Werte müssen geändert werden. Die Zeile "set timeout" setzt den Idle Timeout für die gesamte Verbindungsdauer auf 120 Sekunden. Die "set ifaddr" Zeile ist ein bisschen schwieriger. Hier ist eine genauere Erklärung.

```
set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.0 0.0.0.0
```

Die obige Zeile folgt dem Format von "**set ifaddr [meineAdr[/nn] [seineAdr[/nn] [netzmaske [startAdr]]]]**". Daher ist die erste spezifizierte IP diejenige, die wir als unsere IP wollen. Wenn du eine statische IP Adresse hast, dann kannst du sie hier einsetzen. In unserem Beispiel benutzen wir /0, was besagt, dass keine Bits von dieser IP Adresse übereinstimmen müssen und der gesamte Ausdruck ersetzt werden kann. Die zweite IP behandelt die von uns erwartete IP unserer Gegenstelle. Wenn du sie weißt, dann kannst du sie hier angeben. Wiederum wissen wir nicht in unserer Zeile, welche IP dies wird, also lassen wir sie uns wieder mitteilen. Die dritte Option ist unsere Netzmaske, hier auf 255.255.255.0 gesetzt. Wenn startAdr spezifiziert ist, dann wird diese anstelle von meineAdr während der initialen IPCP Verhandlung; aber es wird nur eine Adresse aus dem meineAdr-Adressbereich akzeptiert. Dies ist nützlich, wenn Verhandlungen mit einigen PPP Implementierungen durchgeführt werden, die keine IP Nummer vergeben, es sei denn, ihr Peer fordert ``0.0.0.0" an.

Die nächste Option "add default HISADDR" setzt unsere Standardroute zu deren IP. Dies ist ‚klebrig', d.h., falls deren IP sich ändern sollte, dann wird unsere Route auch automatisch upgedatet. Mit "enable dns&quit; teilen wir unserem ISP mit, unsere Nameserveradresse zu authentifizieren. Tu dies NICHT, wenn du deinen eigenen lokalen DNS laufen hast, da PPP dies umgehen wird, indem es einige Zeilen in */etc/resolv.conf* schreibt.

Gegenüber den herkömmlichen Login-Methoden, verwenden viele IPS nun entweder CHAP- oder PAP-Authentifizierung. Wenn das der Fall ist, wird unsere Konfiguration etwas anders aussehen:

```
myisp:
set phone 1234567
set authname ppp
set authkey ppp
set login
set timeout 120
set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.0 0.0.0.0
add default HISADDR
enable dns
```

In dem oben genannten Beispiel geben wir unseren Benutzernamen (ppp) und das Passwort (ppp) unter jeweiliger Verwendung von authname und authkey an. Es ist nicht notwendig, anzugeben, ob CHAP- oder PAP-Authentifizierung genutzt wird - es wird automatisch ermittelt. "set login" gibt lediglich an, dass versucht wird, sich mit dem zuvor genannten Benutzernamen und dem Passwort anzumelden.

PPP(8) verwenden

Nun, da wir unsere *ppp.conf* Datei fertig eingerichtet haben, können wir beginnen, eine Verbindung zu unserem ISP aufzubauen. Hier einige Details über häufig verwendete Parameter mit ppp:

- `ppp -auto myisp` - Startet PPP, konfiguriert deine Schnittstellen und wird dich mit deinem ISP verbinden und dann in den Hintergrund verschwinden.
- `ppp -ddial myisp` - Ähnlich wie -auto, aber wenn deine Verbindung abbricht, wird PPP versuchen, sich erneut zu verbinden.

Mit dem Aufruf von */usr/sbin/ppp* ohne Optionen kommst du in den interaktiven Modus. Hier kannst du direkt mit dem Modem interagieren; das eignet sich hervorragend, um Probleme in deiner *ppp.conf* Datei zu debuggen. Wenn das oben genannte fehlschlägt, versuche, */usr/sbin/ppp* ohne Optionen aufzurufen - somit wird ppp im interaktiven Modus gestartet. Die Optionen können einzeln angegeben werden, um nach Fehlern oder anderen Problemen zu suchen. Unter Verwendung der zuvor genannten Einstellungen, wird ppp in /var/log/ppp.log schreiben. Diese Log enthält, wie die Manual-Seite, hilfreiche Informationen.

ppp(8)-Extras

In einigen Situationen möchstest du Befehle ausführen, wenn die Verbindung gerade errichtet oder beendet wurde. Für diese Fälle gibt es zwei Dateien, die du erstellen kannst: */etc/ppp/ppp.linkup* und */etc/ppp/ppp.linkdown*. Beispielkonfigurationen kannst du hier finden:

- ppp.linkup
- ppp.linkdown

ppp(8)-Varianten

Viele ISPs bieten nun xDSL-Dienste an, welche schneller als die herkömmlichen Einwähl-Methoden sind. Dies beinhaltet Varianten wie zum Beispiel ADSL und SDSL. Obwohl kein physikalisches Einwählen stattfindet, basiert die Verbindung weiterhin auf dem Point-to-Point-Protokoll. Beispiele beinhalten:

- PPPoE
- PPPoA
- PPTP

PPPoE/PPPoA

Das ‚Point to Point Protocol over Ethernet' (PPPoE) ist eine Methode, um PPP-Pakete in Ethernet-Frames zu versenden. Das ‚Point to Point Protocol over ATM' (PPPoA) läuft typischerweise in ATM-Netzwerken, wie sie in UK oder Belgien gefunden werden können.

Typischerweise bedeutet das, dass du eine Verbindung mit deinem ISP über eine normale Ethernetkarte und Ethernet-basierendes DSL-Modem herstellen kannst (im Gegensatzu zu einem Nur-USB-Modem).

Wenn du ein Modem hast, das PPPoE/PPPoA versteht, ist es möglich, das Modem so zu konfigurieren, dass es selbst die

Verbindung aufbaut. Wenn das Modem einen ‚bridge'-Modus hat, ist es alternativ möglich, dies zu aktivieren und so das Modem die Pakete einfach zu einer Maschine ‚durchleiten zu lassen', welche PPPoE-Software einsetzt (siehe unten).

Das Haupt-Softwareinterface für PPPoE/PPPoA unter OpenBSD ist pppoe(8), welches die Userland-Implementierung ist (auf fast die gleiche Art und Weise, wie wir ppp(8) weiter oben beschrieben haben). Eine Kernel-PPPoE-Implementierung, pppoe(4), wurde in OpenBSD -current eingebunden; jedoch wird diese für Nicht-current-Benutzer nicht bis OpenBSD 3.7 verfügbar sein.

PPTP

Das ‚Point to Point Tunneling Protocol' (PPTP) ist eine proprietäres Microsoft-Protokoll. Ein pptp-Client ist verfügbar, welcher mit pppd(8) kommuniziert und ist in der Lage, sich zu PPTP-basierenden Virtuellen Privaten Netzwerken (VPN) zu verbindungen, die von einigen Kabel- und xDSL-Anbietern verwendet werden. pptp selbst muss von den Ports oder Packages aus installiert werden. Weitere Anleitungen, wie man pptp einrichtet und verwendet, befinden sich in der Manualseite, welche mit dem pptp-Package installiert wird.

6.6 - Netzwerkparameter tunen

6.6.1 - Wie kann ich meinen Kernel optimieren, so dass er eine größe Anzahl an Neuversuchen und längeren timeouts für TCP Sitzungen hat?

Normalerweise möchtest du das tun, um Routing zuzulassen oder wegen Verbindungsproblemen. Natürlich müssen beide Seiten der Verbindung ähnliche Werte verwenden, damit es am effektivsten ist.

Um dies zu tunen, verwende `sysctl` und erhöhe die Werte von:

```
net.inet.tcp.keepinittime
net.inet.tcp.keepidle
net.inet.tcp.keepintvl
```

Mittels sysctl -a kannst du die derzeitigen Werte dieser (und vieler anderer) Parameter sehen. Um einen Wert zu verändern, verwende etwas wie `sysctl net.inet.tcp.keepidle=28800`.

6.6.2 - Wie kann ich ‚directed broadcasts' aktivieren?

Normalerweise willst du dies nicht tun. Dies erlaubt jemandem, Datenverkehr zu der broadcast Adresse deines verbundenen Netzwerkes zu schicken, wenn du deinen OpenBSD Rechner als Router verwendest.

Aber manchmal kann dies, in geschlossenen Netzwerken, nützlich sein, vor allem wenn man ältere Implementierungen des NetBIOS Protokolles verwendet. Wiederum mit sysctl. `sysctl net.inet.ip.directed-broadcast=1` aktiviert dies. Beachte aber Smurfangriffe, wenn du wissen willst, warum dies standardmäßig nicht aktiviert ist.

6.6.3 - Der Kernel soll bestimmte Ports nicht dynamisch allozieren

Auch dafür gibt es einen eigenen sysctl Befehl. Siehe sysctl(8):

```
Setze die Liste der reservierten TCP Ports, die nicht dynamisch vom
Kernel vergeben werden sollen. Das kann man benutzen, um Daemons davon
abzuhalten, einen speziellen Port zu benutzen, den ein anderes Programm
braucht, damit es funktionieren kann. Listen-Elemente können
mit Kommata und/oder Leerzeichen getrennt werden.

#  sysctl net.inet.tcp.baddynamic=749,750,751,760,761,871

Es ist ebenso möglich, Ports aus der aktuellen Liste hinzuzufügen
oder zu entfernen.

 #  sysctl net.inet.tcp.baddynamic=+748
 #  sysctl net.inet.tcp.baddynamic=-871
```

6.7 - Einfache NFS Anleitung

NFS, oder Network File System (Netzwerkdateisystem), wird verwendet, um ein Dateisystem über das Netzwerk zu verwenden. Du solltest vorher noch folgende Manual Seiten lesen, bevor du versuchst, einen eigenen NFS Server aufzusetzen:

- nfsd(8)
- mountd(8)
- exports(5)

Dieses Kapitel zeigt die Schritte, um ein einfaches NFS System aufzusetzen: Einen Server im LAN und Clients im LAN, die NFS verwenden. Es behandelt nicht, wie man NFS sicher macht. Wir nehmen an, dass du bereits Paketfilterung oder irgendeinen anderen Firewallschutz eingerichtet hast, damit von außerhalb nicht auf NFS zugegriffen werden kann. Wenn du Zugriff via NFS von außerhalb erlauben willst und sensible Daten dort gespeichert hast, dann empfehlen wir dir wärmstens den Gebrauch von IPsec. Ansonsten können andere Leute möglicherweise deinen NFS Datenverkehr sehen. Jemand könnte auch vortäuschen, die IP Adresse zu haben, der du Zugriff auf den NFS Server zulässt. Es gibt mehrere Angriffe, die möglich sind. Wenn IPsec richtig konfiguriert ist, dann schützt es gegen die Art von Angriffen.

Noch eine wichtige Anmerkung wegen Sicherheit. Füge niemals ein Dateisystem zu /etc/exports ohne eine Liste mit Rechnern, die explizit Zugriff haben sollen. Ohne einer solchen Liste, die ein bestimmtes Verzeichnis mounten können, kann jeder, der den Rechner erreichen kann, deine NFS exports mounten.

portmap(8) muss laufen, damit NFS funktionieren kann. Portmap(8) ist ab OpenBSD 3.2 standardmäßig abgeschaltet, so dass du die Zeile

```
portmap=YES
```

in rc.conf.local(8) einfügen und neustarten musst.

Der Server hat die IP **10.0.0.1**. Dieser Server soll nur NFS für Rechner innerhalb dieses Netzwerkes bereitstellen. Der erste Schritt ist deine */etc/exports* Datei zu erstellen. Diese Datei listet die Dateisysteme auf, die du über NFS freigeben willst, und definiert, wer auf sie zugreifen darf. Es gibt viele Optionen, die du in deiner */etc/exports* Datei haben kannst, und am besten ist, du liest die exports(5) Manual Seite. Für dieses Beispiel sieht */etc/exports* so aus:

```
#
# NFS exports Database
# See exports(5) for more information.  Be very careful, misconfiguration
# of this file can result in your filesystems being readable by the world.
/work -alldirs -ro -network 10.0.0 -mask 255.255.255.0
```

D.h., dass das lokale Dateisystem **/work** via NFS zugänglich gemacht wird. **-alldirs** bedeutet, dass Clients jedes Verzeichnis unter dem **/work** Mount Point mounten können. **-ro** bedeutet, dass nur Leseberechtigung gestattet wird. Die letzten zwei Argumente bedeuten, dass nur Clients innerhalb des 10.0.0.0 Netzwerkes mit einer Netzmaske von 255.255.255.0 dieses Dateisystem mounten dürfen. Dies ist wichtig für einige Server, die von verschiedenen Netzwerken zugänglich sind.

Ist einmal deine */etc/exports* Datei eingerichtet, kannst du weitergehen und deinen NFS Server aufsetzen. Du solltest zuerst sicherstellen, dass deine Kernelkonfiguration die Optionen NFSSERVER & NFSCLIENT enthält. (Der GENERIC Kernel beinhaltet diese Optionen.) Dann solltest du die Zeile **nfs_server=YES** in */etc/rc.conf.local* einfügen. Dies wird sowohl nfsd(8) und mountd(8) starten, wenn du rebootest. Nun kannst du fortschreiten und die Dienste selber starten. Diese Dienste müssen als root gestartet werden und du musst sicherstellen, dass portmap(8) auf deinem System läuft. Hier ein Beispiel von nfsd(8), der sowohl mit TCP als auch mit UDP bedient mittels 4 Diensten. Du solltest eine angemessenene Anzahl an NFS Serverdiensten einsetzen, um die maximale Anzahl von gleichzeitigen Clientanfragen, die du bedienen willst, zu bewerkstelligen.

```
# /sbin/nfsd -tun 4
```

Du musst nicht nur den nfsd(8) Server starten, sondern auch mountd(8). Dies ist der Dienst, der eigentlich die Mountanfragen auf NFS bedient. Um mountd(8) zu starten stelle sicher, dass eine leere mountdtab Datei existiert und starte den Daemon:

```
# echo -n >/var/db/mountdtab
# /sbin/mountd
```

Wenn du Änderungen an /etc/exports durchführst, während NFS bereits läuft, musst du mountd dies mitteilen, indem du den Dienst neustartest!

```
# kill -HUP `cat /var/run/mountd.pid`
```

NFS Status überprüfen

Um zu überprüfen, ob alle Dienste laufen und bei RPC registriert sind, verwende rpcinfo(8).

```
$ rpcinfo -p 10.0.0.1
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100005    1   udp    633  mountd
    100005    3   udp    633  mountd
    100005    1   tcp    916  mountd
    100005    3   tcp    916  mountd
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100003    2   tcp   2049  nfs
    100003    3   tcp   2049  nfs
```

Für den Normalgebrauch gibt es ein paar Hilfsprogramme, mit denen du den Status von NFS überprüfen kannst. Eines ist showmount(8) das dir anzeigt, was und wer gerade mountet. Dann gibt es auch noch nfsstat(8), das genauere Statistiken anzeigt. Für showmount(8), versuche `/usr/bin/showmount -a host`. Zum Beispiel:

```
$ /usr/bin/showmount -a 10.0.0.1
All mount points on 10.0.0.1:
10.0.0.37:/work
```

NFS Dateisysteme mounten

NFS Dateisysteme sollten mittels mount(8) geladen werden, oder genauer gesagt, mount_nfs(8). Um ein Dateisystem /work von Host 10.0.0.1 auf dem lokalen Dateisystem /mnt zu laden, tue folgendes (bedenke, dass du nicht IP Adressen verwenden musst, mount wird Hostnamen auflösen):

```
# mount -t nfs 10.0.0.1:/work /mnt
```

Damit dein System dies beim Hochfahren wieder tut, füge folgendes deiner */etc/fstab* hinzu:

```
10.0.0.1:/work /mnt nfs rw 0 0
```

Es ist wichtig, dass du `0 0` am Ende dieser Zeile verwendest, damit dein Rechner nicht versucht, das NFS Dateisystem beim Hochfahren mit fsck zu überprüfen!!!! Die anderen Sicherheitsoptionen wie noexec, nodev und nosuid, sollten auch immer - wenn anwendbar - verwendet werden. Wie zum Beispiel:

```
10.0.0.1:/work /mnt nfs rw,nodev,nosuid 0 0
```

Mit diesen Optionen können keine Geräte oder setuid Programme auf dem NFS Server Sicherheitsmaßnahmen auf dem NFS Client untergraben. Wenn du keine Programme auf diesem NFS Dateisystem auf dem NFS Client ausführen willst, füge noexec hinzu:

6.9 - Aufsetzen einer Bridge mit OpenBSD

Eine Bridge ist ein Link zwischen zwei oder mehr separaten Netzwerken. Anders als ein Router reisen Pakete durch die Bridge "unsichtbar" -- logisch erscheinen die beiden Netzwerksegmente als eines für Rechner auf beiden Seiten der Bridge. Die Bridge wird nur Pakete weiterleiten, die auch von einem Segment in das andere müssen, sie bieten also auch einen einfachen Weg den Verkehr in einem komplexen Netzwerk zu reduzieren und erlauben trotzdem den Zugriff jedes Rechners zu jedem anderen, falls nötig.

Denk daran, dass aufgrund dieser "unsichtbaren" Natur ein Interface in einer Bridge eine IP-Adresse haben kann, aber nicht muss. Wenn sie eine hat, hat die Karte effektiv zwei Betriebsmodi, nämlich eine als Teil der Bridge, die andere als

normale, stand-alone Netzwerk-Karte. Wenn keine der Karten eine IP-Adresse hat, wird die Bridge einfach Netz-Daten verschieben, aber nicht extern administrierbar oder wartbar sein (was auch ein Feature sein kann).

Ein Beispiel einer Bridge Anwendung

Eines meiner Computer Racks hat eine Anzahl alter Systeme, von denen keines eine eingebaute 10BASE-TX Netzwerkkarte hat. Während sie alle einen AUI oder AAUI Stecker haben, sind die Transceiver auf Koax beschränkt. Eine der Maschinen in diesem Rack ist ein OpenBSD-basierender Terminal-Server, der dauerhaft eingeschaltet und auch immer mit einem High-Speed-Netzwerk verbunden ist. Das Hinzufügen einer zweiten Netzwerkkarte mit einem Koax-Port erlaubt mir, diese Maschine als Bridge zum Koax-Netzwerk zu benutzen.

Dieses System hat jetzt zwei Netzwerkkarten (NICs), eine Intel EtherExpress/100 (`fxp0`) und eine 3c590-Combo Karte (`ep0`) für den Koax-Port. `fxp0` ist der Link in mein restliches Netzwerk und wird daher eine IP-Adresse haben, `ep0` macht nur Bridging und hat daher keine. Maschinen, die an das Koax-Segment angeschlossen sind, sollen genauso kommunizieren, als wenn sie im Rest meines Netzwerkes wären. Wie also bewerkstelligen wir das?

Die Datei `hostname.fxp0` enthält die Konfigurationsdaten für die `fxp0` Karte. Diese Maschine soll DHCP machen, also sieht die Datei etwa so aus:

```
$ cat /etc/hostname.fxp0
dhcp NONE NONE NONE NONE
```

Noch keinerlei Überraschungen.

Die `ep0` Karte ist ein wenig anders, wie du dir denken kannst:

```
$ cat /etc/hostname.ep0
up media 10base2
```

Hier sagen wir dem System, es möge das Interface mittels ifconfig(8) aktivieren und auf 10BASE-2 (Koax) setzen. Keine IP Addresse oder ähnliche Information muss für dieses Interface spezifiziert werden. Die Optionen, die von der `ep` Karte akzeptiert werden, sind detailliert in der Manual Seite aufgeführt.

Jetzt müsen wir die Bridge aufsetzen. Bridges werden durch die Existenz einer Datei namens `bridgename.bridge0`. initialisiert. Hier ist zum Beispiel eine Datei für meine Situation:

```
$ cat /etc/bridgename.bridge0
add fxp0
add ep0
up
```

Das sagt aus, es soll eine Bridge aus zwei NICs aufgesetzt und aktiviert werden, fxp0 und ep0. Es ist egal, in welcher Reihenfolge die Karten aufgeführt werden. Denke daran, die Bridge ist symmetrisch -- Pakete fließen ja in beide Richtungen.

Das war es! Reboote, und du wirst eine funktionierende Bridge haben.

Filtern auf der Bridge

Während es sicher auch eine Menge Anwendungen für eine solch einfache Bridge gibt, ist es doch wahrscheinlich, dass du etwas mit den ganzen Paketen TUN willst, während sie durch deine Bridge laufen. Wie zu erwarten, kann man Packet Filter dazu benutzen, den Traffic einzuschränken, der durch deine Bridge fließt.

Denke daran, dass wegen der Natur der Bridge die gleichen Daten über beide Interfaces fließen, aber du nur auf einem Interface filtern brauchst. Deine "Pass all" Statements würden dann wie folgt aussehen:

```
pass in  on ep0  any
pass out on ep0  any
pass in  on fxp0 any
pass out on fxp0 any
```

Sagen wir nun, ich wollte den Traffic filtern, der auf diese alten Maschinen trifft. Ich möchte, dass nur Web- und SSH-Verkehr zu ihnen durchkommt. In diesem Fall lassen wir jeglichen Verkehr durch das `ep0` Interface zu, sowohl rein als auch raus, aber filtern auf dem `fxp0` Interface, indem wir ‚keep state' für die Antwort-Daten benutzen:

```
# Pass all traffic through ep0
pass in quick on ep0 all
pass out quick on ep0 all

# Block fxp0 traffic
block in  on fxp0 all
block out on fxp0 all

pass in quick on fxp0 proto tcp from any to any port {22, 80} \
     flags S/SA keep state
```

Denke daran, dass dieses Regelwerk jeglichen Netzwerk-Verkehr mit Ausnahme von hereinkommendem HTTP- und SSH-Verkehr zur Bridge selbst und den Maschinen "dahinter" verhindert. Andere Resultate werden erzielt, wenn man auf dem anderen Interface filtert.

Um die Bridge zu überwachen und zu kontrollieren, benutze das brconfig(8) Kommando, mit dem man eine Bridge auch nach dem Booten erzeugen kann.

Tipps zum Bridging

- Es wird WÄRMSTENS empfohlen, nur auf einem Interface zu filtern. Wenn es auch möglich ist, auf beiden zu filtern, muss man das vorher jedoch sehr gut verstanden haben.
- Durch die Benutzung der *blocknonip* Option von brconfig(8) oder in bridgename.bridge0 kannst du jeglichen nicht-IP Traffic (wie etwa IPX oder NETBEUI) davon abhalten, sich um deine Filter herumzustehlen. Das kann in einigen Situationen sehr wichtig sein, aber du solltest wissen, dass Bridges für jeglichen Traffic funktionieren, nicht nur für IP.
- Für Bridging müssen die NICs im "Promiscuous mode" sein -- sie lauschen einfach am GESAMTEN Netzwerk-Verkehr, nicht nur an dem, der an das Interface gerichtet ist. Das hat einen höheren Load für CPU und Bus zur Folge, als man denkt. Einige NICs funktionieren leider nicht sauber in diesem Modus, der TI ThunderLAN Chip (tl(4)) ist leider so ein Beispiel, der nicht als Teil einer Bridge funktioniert.

6.10 - Wie boote ich mit PXE? (i386, amd64)

Das ‚Preboot Execution Environment', oder PXE, ist ein Weg, um einen Computer aus dem Netzwerk heraus zu booten, statt von Festplatte, Diskette oder CD-ROM. Diese Technologie wurde zuerst von Intel entwickelt, doch wird nun von den meisten führenden Netzwerkkarten- und Computer-Herstellern unterstützt. Bedenke, dass es viele verschiedene Netzwerkboot Protokolle gibt, PXE ist relativ neu. Traditionellerweise wird das PXE Booting unter Verwendung von ROMs auf dem NIC oder dem Mainboard ausgeführt, doch sind ebenfalls Boot Floppies von etlichen Quellen verfügbar, die ebenfalls das PXE Booting zulassen. Viele ROMs auf älteren NICs unterstützen zwar das Booten vom Netzwerk aus, allerdings NICHT PXE; OpenBSD/i386 oder am64 können mit diesen zur Zeit nicht über das Netzwerk gebootet werden.

Wie funktioniert das PXE Booting?

Zuallerst ist es klug zu verstehen, wie der OpenBSD Bootprozess auf i386 und am64 Plattformen funktioniert. Auf den Bootprozess folgend, sendet die PXE-fähige NIC einen DHCP Request über das Netzwerk. Der DHCP Server wird dem Adapter eine IP zuweisen und gibt ihm den Namen einer Datei, die von einem tftp(1) Server bezogen und ausgeführt werden muss. Diese Datei leitet dann den Rest des Bootprozesses ein. Für OpenBSD ist es die pxeboot Datei, die den Platz der standardmäßigen boot(8) Datei einnimmt. pxeboot(8) ist dann in der Lage, einen Kernel (wie zum Beispiel `bsd` oder `bsd.rd` vom gleichen tftp(1) Server zu laden und auszuführen.

Wie mache ich das?

Der erste und offensichtlichste Schritt ist, dass du einen PXE-bootfähigen Computer oder Netzwerkadapter haben musst. Einige Dokumente weisen darauf hin, dass alle modernen NICs und Computer PXE Ünterstützung hätten, aber das ist einfach nicht wahr -- viele Niedrigpreis Systeme liefern keine PXE ROMs mit oder verwenden ein älteres Netzwerkboot Protokoll. Du brauchst außerdem einen ordentlich konfigurierten DHCP und TFTP Server.

Davon ausgehend, dass eine OpenBSD Maschine die Quelle der Bootdateien ist, muss die dhcpd.conf Datei des DHCP Servers folgende Zeile beinhalten:

```
filename "pxeboot";
```

damit der DHCP Server diese Datei dem bootenden Arbeitsplatz anbietet. Zum Beispiel:

```
shared-network LOCAL-NET {
        option  domain-name "example.com";
        option  domain-name-servers 192.168.1.3, 192.168.1.5;

        subnet 192.168.1.0 netmask 255.255.255.0 {
                option routers 192.168.1.1;
                filename "pxeboot";
                range 192.168.1.32 192.168.1.127;
                default-lease-time 86400;
                max-lease-time 90000;
        }
}
```

Du musst außerdem den tftpd(8) Daemon aktivieren. Dies wird normalerweise durch inetd(8) realisiert. Die standardmäßige OpenBSD Installation hat eine Beispielzeile in `inetd.conf`, die wunderbar für dich funktionieren wird:

```
#tftp  dgram   udp  wait  root  /usr/libexec/tftpd   tftpd -s /tftpboot
```

von der lediglich das ‚#' Zeichen entfernt werden muss und sende inetd(8) ein -HUP Signal, um mitzuteilen, dass `/etc/inetd.conf` neu geladen werden soll. tftpd(8) bietet die Dateien von einem bestimmten Verzeichnis an, in dem Fall von dieser Zeile ist es das `/tftpboot` Verzeichnis, welches wir für dieses Beispiel verwenden werden. Offensichtlich ist, dass dieses Verzeichnis angelegt und eingerichtet werden muss. Typischerweise wirst du hier nur ein paar Dateien für das PXE Booting haben:

- pxeboot, der PXE Bootloader (der die gleichen Funktionen bereitstellt wie boot auf einem Platten-basierten System).
- bsd.rd, der Installationskernel oder `bsd`, ein angepasster Kernel.
- /etc/boot.conf, eine Boot Konfigurationsdatei.

Denke daran, dass `/etc/boot.conf` nur gebraucht wird, wenn der Kernel, den du booten möchtest, nicht `bsd` heißt oder andere pxeboot Standardwerte nicht so sind, wie du sie haben möchtest (zum Beispiel, wenn du eine serielle Konsole wünschst). Du kannst deinen tftpd(8) Server mit einem tftp(1) Client testen, indem du sicherstellst, dass du die benötigten Dateien herunterladen kannst.

Wenn deine DHCP und TFTP Server laufen, bist du bereit, es auszuprobieren. Du wirst PXE boot auf deinem System oder auf der Netzwerkkarte aktivieren müssen; konsultiere deine Systemdokumentation. Wenn du es gesetzt hast, solltest du etwas sehen, das diesem ähnlich ist:

```
Intel UNDI, PXE-2.0 (build 067)
Copyright (C) 1997,1998 Intel Corporation

For Realtek RTL 8139(X) PCI Fast Ethernet Controller v1.00 (990420)

DHCP MAC ADDR: 00 E0 C5 C8 CF E1
CLIENT IP: 192.168.1.76  MASK: 255.255.255.0  DHCP IP: 192.168.1.252
GATEWAY IP: 192.168.1.1
probing: pc0 com0 com1 apm pxe![2.1] mem[540k 28m a20=on]
disk: hd0*
net: mac 00:e0:c5:c8:cf:e1, ip 192.168.1.76, server 192.168.1.252
>> OpenBSD/i386 PXEBOOT 1.00
boot>
```

Zu diesem Zeitpunkt hast du den normalen OpenBSD Bootprompt. Wenn du hier einfach "`bsd.rd`" eintippst, wirst du die Datei `bsd.rd` von dem TFTP Server laden.

```
>> OpenBSD/i386 PXEBOOT 1.00
boot> bsd.rd
booting tftp:bsd.rd: 4375152+733120 [58+122112+105468]=0x516d04
entry point at 0x100120

Copyright (c) 1982, 1986, 1989, 1991, 1993
        The Regents of the University of California.  All rights reserved.
```

```
Copyright (c) 1995-2004 OpenBSD.  All rights reserved.  http://www.OpenBSD.org

OpenBSD 3.6 (RAMDISK_CD) #378: Fri Sep 17 13:04:04 MDT 2004
  ...
```

Der bsd.rd Installationskernel wird nun booten.

Kann ich andere Kerneltypen mit PXE booten, außer `bsd.rd`?

Ja, obwohl mit den Programmen, die zur Zeit in OpenBSD 3.6 enthalten sind, PXE Booting primär zum Installieren des OS gedacht ist.

6.11 - Das Common Address Redundancy Protokoll (CARP)

6.11.1 - Was ist CARP und wie funktioniert es?

CARP ist ein Werkzeug um beim Erreichen von System Redundanz zu helfen, indem mehrere Computer ein einzelnes, virtuelles Netzwerk Interface zwischen sich errichten, so dass, falls irgendein System ausfällt, ein anderes antworten kann, und/oder einen gewissen Grad an Load Sharing zwischen Systemen erlaubt. CARP ist eine Verbesserung vom Virtual Router Redundancy Protokoll (VRRP) Standard. Es wurde entwickelt, nachdem VRRP als nicht frei genug wegen einem möglicherweise-überlappendem Cisco Patent angesehen wurde. Für weitere Informationen über CARPs Ursprünge und den rechtlichen Problemen um VRRP, besuche bitte diese Seite.

Um gesetzliche Konflikte zu umgehen, entwarf Ryan McBride (mit Hilfe von Michael Shalayeff, Marco Pfatschbacher und Markus Friedl) CARP so, dass es fundamental anders war. Die Einbindung von Kryptographie ist eine der prominentesten Änderungen, aber weiterhin nur eine von vielen.

Wie es funktioniert: CARP ist ein Multicast Protokoll. Es gruppiert mehrere physikalische Computer unter einer oder mehrerer virtuellen Adressen zusammen. Von diesen ist ein System der Master und antwortet auf alle Pakete, die für diese Gruppe bestimmt sind, während die anderen Systeme als ‚hot spares' agieren. Unbedeutend wie die IP und MAC Adressen des lokalen Interfaces sind, werden Pakete, die zum CARP Interface gesendet worden sind, mit CARP Informationen zurückgesendet.

Zu konfigurierbaren Intervallen bekündet der Master seine Operation auf der IP Protokoll Nummer 112. Wenn der Master offline geht, beginnen die anderen Systeme in der CARP Gruppe mit dem bekünden. Der Host, der in der Lage ist am häufigsten zu bekünden, wird der neue Master. Wenn das Hauptsystem wieder online kommt, wird es standardmäßig ein Backup Host, obwohl, wenn es wünschenswerter ist, dass ein Host immer Master wird wenn das möglich ist (z.B. wenn ein Host eine schnelle Sun Fire V120 ist und die anderen vergleichbar langsame SPARCstation IPCs sind), kannst du sie so konfigurieren.

Während hoch redundante und Fehler-tolerante Hardware die Notwendigkeit von CARP verringert, vernichtet sie sie nicht. Es gibt keine Hardwarefehlertoleranz die in der Lage ist zu helfen, wenn jemand das Stromkabel herauszieht oder wenn dein Systemadministrator `reboot` im falschen Fenster eintippt. CARP macht es außerdem einfacher den Patch- und Rebootzyklus transparent den Anwendern gegenüber zu gestalten, und einfacher ein Software oder Hardware Upgrade zu testen -- wenn es nicht funktioniert, kannst du auf deine ‚spares' zurückgreifen, bis es behoben ist.

Es gibt jedoch Situationen, in denen CARP nicht helfen wird. CARPs Design setzt voraus, dass die Mitglieder einer Gruppe sich im selben physikalischen Subnet befinden und jedes Interface benötigt eine reale, statische IP Adresse zusätzlich zu einer statischen CARP IP Adresse. Ähnlich werden Dienste, die eine durchgehende Verbindung zum Server benötigen (so wie SSH oder IRC), nicht transparent auf andere Systeme weitergeleitet -- obwohl in diesem Fall CARP helfen kann, die Ausfallzeit zu minimieren. CARP wird von sich aus Daten zwischen Applikationen nicht synchronisieren, dies muss durch "alternative Kanäle" wie zum Beispiel pfsync(4) (für redundantes Filtern), manuelles Duplizieren von Daten zwischen Systemen mit rsync, oder was auch immer für deine Anwendungen geeignet ist, durchgeführt werden.

6.11.2 - Konfiguration

CARPs Kontrollen befinden sich an zwei Orten: sysctl(8) und ifconfig(8). Lass uns nun zuerst die sysctls betrachten.

Die erste sysctl, `net.inet.carp.allow`, definiert, ob der Host überhaupt CARP Pakete handhabt. Klarerweise ist dies notwendig, um CARP nutzen zu können. Diese sysctl ist standardmäßig aktiviert.

Die zweite, `net.inet.carp.arpbalance`, wird für das Load Balancing verwendet. Wenn diese Funktion aktiviert ist, wird CARP ein ‚source-hash' auf die Quell-IP der Anfrage durchführen. Dieser ‚hash' wird dann verwendet, um einen

virtuellen Host aus dem zur Verfügung stehenden Pool auszuwählen, damit dieser die Anfrage verarbeitet. Dies ist standardmäßig deaktiviert.

Die dritte, `net.inet.carp.log`, loggt CARP Fehler. Standardmäßig deaktiviert.

Vierte, `net.inet.carp.preempt` aktiviert natürliche Auswahl zwischen CARP Hosts. Der passendste für den Job (das heißt, wer in der Lage ist am schnellsten zu Bekünden) wird zum Master. Standardmäßig deaktiviert, das bedeutet, dass ein System, das nicht zum Master auserwählt wurde, nicht versuchen wird den Master Status (wieder) zu erhalten.

Alle diese sysctl Variablen sind in sysctl(3) dokumentiert.

Für den Rest von CARPs Konfiguration verlassen wir uns auf ifconfig(8). Zwei von den vier CARP-spezifischen Befehlen, `advbase` und `advskew`, befassen sich mit dem Intervall zwischen CARP Bekündungen. Die Formel (in Sekunden) ist `advskew` dividiert durch 255 und dann addiert zu `advbase`. `advbase` kann verwendet werden, um Netzwerkverkehr zu verringern oder eine längere Latenz zuzulassen, bevor ein Backup Host übernimmt; `advskew` lässt dir die Möglichkeit zu verwalten, welcher Host Master sein wird, ohne große ‚failover´ Verzögerungen (sollte das notwendig sein).

Als nächstes setzt `pass` ein Passwort und `vhid` setzt die virtuelle Hostidentifizierungsnummer der CARP Gruppe. Du musst jeder CARP Gruppe eine einzigartige Nummer verteilen, selbst wenn (für Load Balancing Zwecke) sie sich die gleiche IP Adresse teilen. CARP ist auf 255 Gruppen begrenzt.

Lass uns alle diese Einstellungen zusammen in eine Grundkonfiguration packen. Angenommen du setzt zwei identische Web Server auf, *rachael* (192.168.0.5) und *pris* (192.168.0.6), um ein älteres System zu ersetzen, das unter 192.168.0.7 verfügbar war. Die Befehle:

```
rachael# ifconfig carp0 create
rachael# ifconfig carp0 vhid 1 pass tyrell 192.168.0.7
```

erstellen das carp0 Interface und geben es eine vhid von 1, ein Passwort das *tyrell* lautet und die IP Adresse 192.168.0.7. Um es über die nächsten Reboots hinaus permanent zu machen, kannst du eine `/etc/hostnamecarp0` Datei anlegen, die wie folgt aussieht:

```
inet 192.168.0.7 255.255.255.0 192.168.0.255 vhid 1 pass tyrell
```

Mache das gleiche auf *pris*. Welches System von beiden das CARP Interface zu erst aufsetzt wird Master.

Bedenke, dass auf einer Maschine mit mehreren Interfaces das CARP Interface im gleichen Subnet des physikalischen Interfaces liegt.

Aber lass uns sagen du setzt nicht von Anfang an auf. *Rachael* war bereits unter der Adresse 192.168.0.7 vorhanden. Wie umgehst du das? Glücklicherweise hat CARP kein Problem mit einem System, das eine IP als physikalische Interface Adresse und in einer CARP Gruppe besitzt, so dass kein Grund besteht die vorherigen Befehle zu ändern. Trotz allem tendiert es dazu sauberer zu sein jeweils eine IP für jedes System zu haben -- es macht individuelle Überwachungen und Zugriffe viel einfacher.

Lass uns nun einen weiteren Schwierigkeitsgrad hinzufügen; wir möchten, dass *rachael* so lange wie möglich Master bleibt. Es gibt einige Gründe, warum wir das benötigen könnten: Hardware Unterschiede, einfache Vorurteile, "wenn das System nicht Master ist, wird es Probleme geben" oder zu wissen, wer der standardmäßige Master ist ohne mit Skripten die Ausgabe von ifconfig zu verarbeiten und per E-Mail zu versenden.

Auf *rachael* werden wir die sysctl verwenden, die wir weiter oben erstellt haben und editieren dann `/etc/sysctl.conf`, um sie permanent zu machen.

```
rachael# sysctl net.inet.carp.preempt=1
```

Wir werden ebenfalls die Konfiguration auf *pris* durchführen:

```
pris# ifconfig carp0 advskew 100
```

Dies verzögert die Bekündungen von *pris* ein wenig, was bedeutet, dass *rachael* Master sein wird, wenn der Host angeschlossen ist.

Bedenke, dass du "proto carp" mit folgender Zeile an alle beteiligten Interfaces übergeben musst, wenn du PF auf einem geCARPten Computer verwendest:

```
pass on fxp0 proto carp keep state
```

6.11.3 - Load Balancing

Siehe nun einige Monate nach vorne. Unsere Firma des vorherigen Beispiels ist so gewachsen, dass sie an dem Punkt angekommen ist, an dem ein einzelner Web Server die Last gerade so verarbeiten kann. Was nun? CARP ist die Rettung. Es ist Zeit, Load Balancing zu versuchen. Erstelle ein neues CARP Interface und eine Gruppe auf *rachael*:

```
rachael# ifconfig carp1 create
rachael# ifconfig carp1 vhid 2 advskew 100 pass bryant 192.168.0.7
```

Auf *pris* werden wir ebenfalls eine neue Gruppe und Interface anlegen und dann das "preempt" sysctl setzen:

```
pris# ifconfig create carp1
pris# ifconfig carp1 vhid 2 pass bryant 192.168.0.7
pris# sysctl net.inet.carp.preempt=1
```

Nun haben wir zwei CARP Gruppen mit der gleichen IP Adresse. Jede Gruppe zeigt auf einen anderen Host, was bedeutet, dass *rachael* Master der originalen Gruppe bleibt, aber *pris* wird die neue übernehmen.

Alles, was wir nun tun müssen, ist die Load Balancing sysctl auf beiden Systemen zu laden, die wir zuvor besprochen haben:

```
# sysctl net.inet.carp.arpbalance=1
```

Während diese Beispiele für einen zwei-Maschinen Cluster sind, gelten die gleichen Prinzipien auch für mehrere Systeme. Bitte bedenke, dass es trotzdem nicht erwartet wird, dass du perfekte 50/50 Distribution zwischen den beiden Maschinen erreichst -- CARP verwendet einen ‚hash' der ankommenden IP Adresse um zu ermitteln, welches System die Anfrage verarbeitet, statt durch Auslastung zu entscheiden.

6.11.4 - Weitere Informationen zu CARP

- carp(4)
- ifconfig(8)
- sysctl(8)
- sysctl(3)
- Firewall Failover with pfsync and CARP von Ryan McBride

6.12 - OpenNTPD verwenden

Genaue Zeit ist wichtig für viele Computer Applikationen. Trotzdem haben viele Leute bemerkt, dass ihre $5 Uhr eine genauere Uhrzeit halten kann als ihr $2000 Computer. Zusätzlich zum Wissen, welche Uhrzeit gerade ist, ist es ebenfalls häufig wichtig, Computer zu synchronisieren, so dass sie alle mit der gleichen Uhrzeit übereinstimmen. Für eine gewisse Zeit hat ntp.org eine Network Time Protokoll (RFC1305, RFC2030) Applikation entwickelt, verfügbar durch Ports, die verwendet werden kann, um die Uhren auf den Computern über das Internet zu synchronisieren. Trotzdem ist dies ein nicht-triviales Programm zum Einrichten, schwerer Code zum Überprüfen und hat eine große Speicheranforderung. Kurz gesagt spielt es eine wichtige Rolle für einige Leute, aber es ist weit entfernt von einer Lösung für jedermann.

OpenNTPD ist ein Versuch, einige dieser Probleme zu lösen, es einfacher-zu-administrieren zu machen und ein sicherer und simpler NTP kompatibler Weg zu sein, um eine genaue Uhrzeit auf deinem Computer zu haben. OpenBSDs ntpd(8) wird von einer einfach zu verstehenden Konfigurationsdatei gesteuert, `/etc/ntpd.conf`.

Aktiviere ntpd(8) einfach durch rc.conf.local und das Resultat wird sein, dass dein Computer sich selbst mit den pool.ntp.org Servern synchronisiert, einer Sammlung von öffentlich verfügbaren Zeit Servern. Wenn deine Uhr auf deinem Computer sehr falsch geht, solltest du sie zuerst so genau wie möglichen einstellen, möglicherweise unter Verwendung von rdate(8), da ntpd(8) deine Uhr nur SEHR langsam durch adjtime(2) angleicht -- es kann *mehrere Stunden* dauern (oder Tage oder sogar noch länger) um eine sehr ungenau eingestellte Uhr zu synchronisieren (*-current* wird unmittelbar nach dem Boot die Uhr setzen). Wenn deine Uhr erst einmal genau eingestellt ist, wird ntpd sie auf einem sehr hohen Genauigkeitsgrad halten.

6.12.1 - "Aber OpenNTPD ist nicht so genau wie der ntp.org Daemon!"

Das mag wahr sein. Es war nicht OpenNTPDs Entwurfsziel, es war vorgesehen, dass es frei, simpel, zuverlässig und sicher ist. Wenn du wirklich Mikrosekunden Präzision mehr als die Vorteile von OpenNTPD brauchst, tu dir keinen Zwang an und verwende ntp.orgs ntpd, da er weiterhin durch Ports und Packages verfügbar sein wird. Es existieren weder ein Plan noch das Verlangen, OpenNTPD mit allen vorstellbaren Funktionen vollzustopfen.

6.12.2 - "Jemand hat behauptet, dass OpenNTPD ‚schädlich' ist!"

Das wird wahrscheinlich Brad Knowles gewesen sein. Wenn genaue Zeit wichtig ist, haben einige Benutzer berichtet, dass die Ergebnisse von OpenNTPD besser sind als die von ntp.orgs ntpd. Wenn Sicherheit wichtig ist, ist OpenNTPDs Code sehr viel besser lesbar (und daher, kontrollierbar) und wurde unter Verwendung von OpenBSD Funktionsaufrufen wie strlcpy statt portableren Funktionen wie strcpy entwickelt und wurde von Anfang an sicher geschrieben, nicht "später sicher gemacht". Wenn es wertvoll ist, dass so viele Leute wie möglich Zeit Synchronisierung verwenden, macht es OpenNTPD einer großen Anzahl an Leuten einfacher, diese zu nutzen. Wenn das "schädlich" ist, stimmen wir dem voll und ganz zu.

Es gibt Anwendungsgebiete, bei denen ntp.orgs ntpd genauer ist, trotzdem geht man davon aus, dass für 95% der anderen Anwender OpenNTPD mehr als ausreichend sein wird.

Eine ausführlichere Antwort hierauf von den Entwicklern von OpenNTPD kann hier gelesen werden.

6.13 - Was sind meine Wireless Netzwerk Optionen?

OpenBSD hat für ein paar Wireless Chipsätze Unterstützung:

- awi(4) AMD 802.11 PCnet Mobile
- an(4) Aironet Communications 4500/4800
- wi(4) Prism2/2.5/3
- atw(4) ADMtek ADM8211

Karten, die auf diesen Chips basieren, können fast genauso wie andere Netzwerkkarten genutzt werden, um ein OpenBSD System an ein existierendes Wireless Netzwerk anzubinden (bitte siehe die Manual Seiten für präzise Details). Die Prism2 und Prism3-basierenden Karten können jedoch auch in dem "Host-Based Access Point" Modus genutzt werden, das ihnen erlaubt, in deinen Wireless Access Point für dein Netzwerk als Teil deiner Firewall gesetzt zu werden. Unglücklicherweise werden die Prism2/3-basierenden Netzwerkkarten nicht mehr von den meisten der "Massenmarkt" Herstellern produziert, da sie auf die günstigeren und neuen Chips wechselten, oft, ohne die Modellnummer gewechselt zu haben. Die meisten der Hersteller von neuen Chips haben entschieden, keine Dokumentation zu veröffentlichen, die für die Entwicklung von freien und offenen Treibern für diese Devices benötigt werden (Wenn du möchtest, kannst du die Hersteller kontaktieren und sie darum bitten, ihre Richtlinien zu ändern).

Zum Glück sind Prism2/3-basierende Karten weiterhin durch den Gebrauchthandel verfügbar und ein paar Hersteller verstehen den Wert eines ordentlich dokumentierten, Open-Source freundlichen Chipsatzes und setzen die professionelle Produktion von Prism-basierenden Karten fort. Ein Verkäufer dieser Karten ist Netgate.com. Wenn ein Hersteller nicht ausdrücklich ein aktuelles Produkt als Prism-basierend kennzeichnet, kann man normalerweise davon ausgehen, dass es nicht mit dem wi(4) Treiber kompatibel sein wird. Bedenke, dass die neuesten Prism-Serien Chips (wie zum Beispiel Prism-GT) nicht unterstützt werden.

Eine andere Möglichkeit, mit deiner OpenBSD-basierenden Firewall einen Wireless Access anzubieten, ist die Verwendung einer konventionellen NIC und einem externen Bridging Access Point. Dies hat den zusätzlichen Vorteil, dass du die Position der Antenne mit Leichtigkeit an die Stelle ändern kannst, an der sie am effektivsten ist, was nicht häufig direkt auf der Rückseite deiner Firewall ist.

[FAQ Index] [Zum Kapitel 5 - Das System aus dem Source-Code erzeugen] [Zum Kapitel 7 - Tastatur- und Bildschirm-Kontrollen]

---

www@openbsd.org

$OpenBSD: faq6.html,v 1.74 2005/02/04 20:39:14 jufi Exp $

OpenBSD

[FAQ Index] [Zu Sektion 6 - Netzwerk] [Zu Sektion 8 - Allgemeine Fragen]

7 - Tastatur- und Display-Kontrollen

Inhaltsverzeichnis

7.1 - Wie verändere ich die Belegung der Tastatur? *(wscons)*

Die Ports, die den wscons(4) Konsole Treiber benutzen: alpha, amd64, cats, hppa, i386, macppc, sparc, sparc64 und vax.

Bei wscons(4) Konsolen werden die meisten Optionen mit dem wsconsctl(8) Utility kontrolliert. Beispielsweise würde man die Tastaturbelegung mit Hilfe von wsconsctl(8) wie folgt ändern:

```
# wsconsctl -w keyboard.encoding=de
```

Im nächsten Beispiel werden wir die "Caps Lock" Taste neu belegen, und zwar so, dass sie nun "Steuerung L" ist:

```
# wsconsctl -w keyboard.map+="keysym Caps_Lock = Control_L"
```

7.2 - Wird die Maus auch unter der Konsole unterstützt?

Für die alpha, amd64 und i386 Plattformen bietet OpenBSD den wsmoused(8) an, einen Port des moused(8) von FreeBSD. Er kann bei jedem Booten automatisch aktiviert werden, indem du die passende Zeile in rc.conf(8) editierst.

7.3 - Wie lösche ich die Konsole, sobald sich ein User ausloggt?

Dazu musst du eine Zeile in /etc/gettytab(5) einfügen. Ändere die Sektion, die so aussieht:

```
P|Pc|Pc console:\
        :np:sp#9600:
```

Füge die Zeile "`:cl=\E[H\E[2J:`" am Ende hinzu, so dass das ganze schlussendlich so aussieht:

```
P|Pc|Pc console:\
        :np:sp#9600:\
        :cl=\E[H\E[2J:
```

7.4 - Auf den Konsolen 'Scrollback Puffer' zugreifen *(amd64, i386 und einige Alphas)*

OpenBSD bietet auf einigen Plattformen einen sogenannten "console scrollback buffer". Das macht es möglich, Informationen zu sehen, die bereits wieder oben aus dem Bildschirm gescrollt sind. Um sich auf- und abwärst zu bewegen, benutze einfach die Tastenkombinationen `[SHIFT]+[BILD AUF]` und `[SHIFT]+[BILD AB]`

Der Standard-Scrollback-Puffer und somit auch die Anzahl der Seiten, die du so betrachten kannst, beträgt 8. Das ist ein Feature des vga(4) Treibers, daher funktioniert das alles natürlich nicht ohne VGA Karte auf jeglichen Plattformen (viele Alpha Systeme haben TGA video).

7.5 - Wie wechsel ich zwischen den Konsolen?*(amd64, i386 und einige Alphas)*

Auf amd64, i386 und einigen Alphas mit vga(4) Karte bietet OpenBSD sechs virtuelle Terminal an, /dev/ttyC0 bis /dev/ttyC5. ttyC4 ist fü die Benutzung durch das X Window System reserviert, was noch fünf Text-Konsolen übrig lässt. Du kannst zwischen ihnen hin- und herspringen, indem du Tastaturkombinationen `[STRG]+[ALT]+[F1]`, `[STRG]+[ALT]+[F2]`, `[STRG]+[ALT]+[F3]`, `[STRG]+[ALT]+[F4]` und `[STRG]+[ALT]+[F6]` benutzt.

Die X Umgebung benutzt ttyC4, `[STRG]+[ALT]+[F5]`. Wenn du X benutzt, bringen dich die `[STRG]+[ALT]+[F`*n*`]` Tasten auf die Text-Konsolen, `[STRG]+[ALT]+[F5]` bringt dich wieder auf die grafische Oberfläche. Wenn du mehr als die standardmäßige Anzahl von Text-Konsolen haben willst, kannst du den wsconscfg(8) Befehl benutzen, um Bildschirme für ttyC6, ttyC7 und weitere zu erzeugen. Zum Beispiel:

```
wsconscfg -t 80x25 6
```

erzeugt ein virtuelles Terminal für ttyC6, was man mit `[STRG]+[ALT]+[F7]` erreicht. Vergiss nicht, diesen Befehl deiner `rc.local(8)` Datei hinzuzufügen, wenn der zusätzliche Bildschirm auch nach dem nächsten Reboot noch vorhanden sein soll.

Denk daran, dass du keinen "`login:`" Prompt auf der neu erzeugten virtuellen Konsole erhälst, bis du sie in `/etc/ttys(5)`, auf "on" setzt und entweder mit init(8) rebootest oder ein HUP Signal mittels kill(1) sendest.

7.6 - Wie bekomme ich eine Konsolen-Auflösung von 80x50? *(amd64, i386)*

amd64 und i386 User haben normalerweise eine Konsole mit 25 Zeilen und jede hat 80 Buchstaben. Viele VGA Grafikkarten sind aber in der Lage, eine höhere Textauflösung von 50 Zeilen mit 80 Buchstaben darzustellen.

Zunächst einmal kann man einen Font, der die gewünschte Auflösung unterstützt, mittels des wsfontload Befehls laden. Der Standard-80x25-Text-Bildschirm benutzt 8x16 pixel Fonts, um die vertikale Auflösung zu verdoppeln, müssen wir 8x8 pixel Fonts benutzen.

Danach müssen wir die virtuelle Konsole löschen und in der gewünschten Auflösung neu erzeugen, und zwar mit dem wsconscfg Befehl.

Das kann während des Bootens automatisch am Ende deiner `rc.local` Datei geschehen:

```
wsfontload -h 8 -e ibm /usr/share/misc/pcvtfonts/vt220l.808
wsconscfg -dF 5
wsconscfg -t 80x50 5
```

Wie bei jeder Änderung deiner Systemkonfiguration solltest du etwas Zeit mit den passenden Manual Seiten verbringen, um zu verstehen, was diese Befehle eigentlich tun.

Die erste Zeile dort oben lädt den 8x8 Font. Die zweite Zeile löscht den virtuellen Bildschirm Nummer 5 (den man mit `[STRG]-[ALT]-[F6]` erreichen würde). Die dritte Zeile erzeugt einen neuen Bildschirm 5 mit der Auflösung 50 Zeilen mit je 80 Buchstaben. Wenn du alles so gemacht hast, werden deine anderen virtuellen Bildschirme ganz normal im 80x25 Modus erscheinen, und der neue Bildschirm 5 mit 80x50 ist mit `[STRG]-[ALT]-[F6]` erreichbar.

Denke daran, dass `[STRG]-[ALT]-[F1]` hier Bildschirm 0 ist. Wenn du die anderen Schirme auch ändern willst, wiederhole einfach die Schritte "delete" und "add screen" für jeden Bildschirm, den du mit 80x50 betreiben willst.

Du solltest aber Bildschirm 4 (ttyC4, `[STRG]-[ALT]-[F5]`) unverändert lassen, da dieser von X als grafischer Schirm genutzt wird. Außerdem ist es nicht möglich, die Auflösung des ersten Konsolen-Device zu ändern (also ttyC0).

Natürlich können all diese Befehle auch als root an der Konsole eingegeben werden, oder besser mittels sudo(8).

Hinweis: Das funktioniert nicht mit allen Grafikkarten. Dummerweise unterstützen nicht alle Grafikkarten die von wscons benötigten Fonts für den 80x50 Text-Modus. In diesen Fällen solltest du über den Einsatz von X nachdenken.

7.7 - Wie kann ich eine serielle Konsole benutzen ?

Es gibt viele Gründe, warum du auf deinem OpenBSD System eine serielle Konsole benutzen willst:

- Aufnehmen der Ausgabe der Konsole (zur Dokumentation).
- Remote Management.
- Einfachere Wartung einer großen Anzahl von Maschinen
- Ein sauberes dmesg von Maschinen bekommen, von denen das sonst nicht so einfach ist.
- Eine saubere "trace" und "ps" Ausgabe bieten, wenn dein System gecrasht ist, so dass die Entwickler eine Chance haben, das Problem zu beheben.

OpenBSD unterstützt auf den meisten Plattformen eine serielle Konsole, die Details weichen aber stark ab.

Bedenke, dass serielles Arbeiten/Interfacing KEINE einfache Sache ist -- du wirst oft ungewöhnliche Kabel benötigen und Ports sind oft nicht zwischen den Maschinen standardisiert und manchmal noch nicht mal auf einer einzelnen Maschine konsistent. Ein komplettes Tutorial über serielles Arbeiten/Interfacing geht zwar weit über die Möglichkeiten dieses Artikels hinaus, aber wir geben dir trotzdem einen guten Rat: Nur weil der Stecker hineinpasst, heisst das noch lange nicht, dass es auch funktioniert.

/etc/ttys Änderung

Es gibt zwei Punkte, um eine funktionierende serielle Konsole unter OpenBSD zu erhalten. Erstens musst du OpenBSD haben, um deinen seriellen Port als Konsole für Status-Ausgaben und ‚single user' Modus benutzen zu können. Zweitens muss dein serieller Port eingeschaltet sein, um als interaktives Terminal benutzbar zu sein, so dass ein Benutzer sich einloggen kann, wenn die Maschine im ‚multi user' Betrieb läuft. Dieser Teil ist zwischen den verschiedenen Plattformen relativ gleich und wird hier besprochen.

Terminal Sitzungen werden von der Datei */etc/ttys* kontrolliert. Bevor OpenBSD dir auf der seriellen Konsole einen "`login:`" Prompt serviert, musst du das ganze in */etc/ttys* einschalten, normalerweise gibt es ja andere Verwendungen für die serielle Schnittstelle. Bei Plattformen, die normalerweise eine angeschlossene Tastatur und Bildschirm haben, ist die serielle Konsole standardmäßig abgeschaltet. Wir benutzen hier die i386-Plattform als Beispiel. In diesem Fall musst du die Zeile editieren, die wie folgt aussieht:

```
tty00   "/usr/libexec/getty std.9600"   unknown off
```

Ist zu ändern in:

```
tty00   "/usr/libexec/getty std.9600"   vt100   on secure
```

`tty00` ist hier der serielle Port, den wir als Konsole nutzen wollen. Das "`on`" aktiviert das *getty* für den seriellen Port, so dass ein "`login:`" Prompt präsentiert wird, das "`secure`" erlaubt ein root (uid 0) Login auf dieser Konsole (das kann man wollen, oder auch nicht) und das "`9600`" ist die Baud-Rate des Terminals. Denke daran, dass du die serielle Konsole auch ohne diese Schritte für Installationen benutzen kannst, da das System im ‚singel user' Modus läuft, und *getty* gar nicht erst für den Login benutzt wird.

Auf einigen Plattformen und einigen Konfigurationen ist es notwendig, das System im ‚single user' Modus zu booten, damit die Änderungen durchgeführt werden können.

amd64 und i386

Um den Boot-Prozess dazu zu bewegen, den seriellen Port als Konsole zu verwenden, erzeuge oder editiere deine */etc/boot.conf* Datei, damit sie folgende Zeile enthält:

```
set tty com0
```

damit du den ersten seriellen Port als Konsole verwendest. Die Standard-Baud-Rate ist 9600bps, das kann mittels der stty Option in */etc/boot.conf* angepasst werden. Diese Datei wird auf deinem Boot-Laufwerk abgelegt, was auch deine Installations-Floppy sein kann, oder das Kommando kann ebenso gut am `boot>` Prompt vom OpenBSD ‚second stage' Bootloader eingegeben werden, wenn es nur ein einziges Mal (oder zum ersten Mal) genutzt werden soll.

amd64 und i386 Hinweise:

- OpenBSD zählt die seriellen Ports beginnend mit *tty00*, DOS/Windows mit *COM1*. Denke also daran, dass *tty02 COM3* ist und nicht *COM2*
- Einige Systeme können sogar ohne eingesteckte Grafikkarte funktionieren, aber sicher nicht alle -- viele Systeme sehen das als Fehler an. Manche Systeme verweigern sogar ohne eingesteckte Tastatur den Dienst.
- Einige Systeme sind in der Lage alle BIOS Tastatur- und Bildschirm-Aktivitäten dank einer Konfigurationsoption auf einen seriellen Port zu lenken, so dass die Maschine vollständig über den seriellen Port verwaltet werden kann. Möglicherweise ist das bei dir anders -- wenn man dieses Feature benutzt, verhindert manchmal das BIOS, dass der Bootloader den seriellen Port sieht und somit kann der Kernel ihn auch nicht benutzen. Manches BIOS hat eine Option namens "Continue Console Redirection after POST" (Power On Self Test), dies sollte auf "OFF" gestellt sein, so dass der Bootloader und der Kernel ihre eigene Konsole benutzen können. Dummerweise ist dieses Feature nicht universell.
- PC kompatible Computer sind nicht entworfen worden, um von einer seriellen Konsole aus bedient zu werden, im Gegensatz zu einigen anderen Plattformen. Sogar die Systeme, die eine serielle Konsole unterstützen, haben dafür im Normalfall eine Option im BIOS - und sollte diese Information verloren gehen, wird das System wieder nach normal angeschlossener Tastatur und Monitor suchen. Du solltest immer einen Monitor und eine Tastatur für deinen amd64 und i386 griffbereit haben, um im Notfall darauf zurückgreifen zu können.
- Du wirst */etc/ttys* wie oben anpassen müssen.
- Nur der erste serielle Port (com0) wird als serielle Konsole auf amd64 und i386 unterstützt.

SPARC und UltraSPARC

Diese Maschinen sind so entworfen worden, dass sie vollständig über eine serielle Konsole gewartet werden können. Entferne einfach die Tastatur von der Maschine und das System läuft über die serielle Konsole.

SPARC und UltraSPARC Hinweise

- Die zwei seriellen Ports auf SPARC heißen *ttya*, *ttyb*, etc.
- Anders als auf anderen Plattformen ist es nicht notwendig, irgendwelche Änderungen an */etc/ttys* zu machen, um die serielle Konsole benutzen zu können.

- Die SPARC/UltraSPARC Systeme interpretieren ein BREAK Signal auf dem Konsolen-Port als das selbe wie ein STOP-A Kommando, und bringt das System zurück zum Forth Prompt, und hält jede Anwendung und das Betriebssystem an diesem Punkt an. Das ist recht nützlich, wenn man es braucht, aber unglücklicherweise senden einige serielle Terminals beim Power-Down und einige RS-232 Switche etwas, was der Computer als Break-Signal interpretiert und halten damit die Maschine an. Überprüfe das, bevor du damit in einer Produktionsumgebung arbeitest.
- Wenn du Tastatur und Monitor angeschlossen hast, kannst du trotzdem die serielle Konsole benutzen, indem du die folgenden Befehle am `ok` Prompt eingibst:

```
ok setenv input-device ttya
ok setenv output-device ttya
ok reset
```

  Wenn Tastatur und Monitor (ttyC0) in */etc/ttys* (siehe oben) aktiv sind, kannst du sie zusätzlich benutzen.

MacPPC

Die MacPPC Maschinen sind durch OpenFirmware für die Benutzung mittels serieller Konsole vorbereitet. Benutze die folgenden Befehle:

```
ok setenv output-device scca
ok setenv input-device scca
ok reset-all
```

Setze deine serielle Konsole auf 57600bps, 8N1.

MacPPC Hinweise

- Unglücklicherweise ist die serielle Konsole auf den meisten MacPPCs nicht direkt verfügbar. Während die meisten dieser Maschinen zwar serielle Hardware haben, ist sie nicht von außen zugänglich. Einige Firmen bieten jedoch Zusatz-Hardware für verschiedene Macintosh-Modelle, so dass dann der Port als serielle Konsole (oder anderweitig) nutzbar ist. Verwende die Suchmaschine deines Vertrauens und suche nach etwas wie "Macintosh internal serial port".
- Du wirst `tty00` in */etc/ttys* auf `on` ändern müssen und die Geschwindigkeit auf 57600 anstelle der standardmäßigen 9600 setzen müssen, wie oben beschrieben, und zwar im ‚single user' Modus, bevor du eine funktionierende serielle Konsole hast.

Mac68k

Die serielle Konsole wird im *Booter* Programm ausgewählt, unter dem "`Options`" Pull-Down Menü, dann "`Serial Ports`". Prüfe den "Serial Console" Knopf, dann wähle den Modem- oder Drucker-Port. Du wirst ein Macintosh Modem oder Drucker-Kabel an den seriellen Port des Macs anschließen müssen. Wenn das in Zukunft deine Standardeinstellung sein soll, musst du dem Booter-Programm sagen, dass es deine Optionen speichern soll.

Mac68k Hinweise

- Der Modem Port ist *tty00*, der Drucker Port ist *tty01*.
- Der Mac68k schaltet seinen seriellen Port nicht ein, bis man ihn danach fragt, also wird deine breakout Box keinerlei Signale anzeigen, bis der OpenBSD Boot Prozess gestartet ist.
- Du wirst den Port in (*tty00* oder *tty01*) wie oben beschrieben einschalten müssen.

7.8 - Wie schalte ich meinen Konsolen-Bildschirmschoner ein ? (wscons)

Wenn du möchtest, dass sich deine Konsole nach einer gewissen Zeit der Inaktivität von X abschaltet, kannst du die folgenden wscons(4) Variablen ändern:

- **`display.vblank`** auf `on` gesetzt, schaltet das vertical sync Signal ein, was einige Monitore dazu

veranlasst, in den "energy saver" Modus zu gehen. Es wird hinterher mehr Zeit beanspruchen, den Monitor wieder in den Normalmodus zu bringen, aber es reduziert den Energieverbrauch und die Hitzeentwicklung neuerer Monitore. Im Modus `off` wird zwar der Bildschirm schwarz, aber der Monitor empfängt weiter das horizontale und vertikale sync Signal und die Rückkehr zum Normalbetrieb geschieht augenblicklich.

- **`display.screen_off`** gibt die Verzögerungszeit in tausendstel Sekunden an, also wäre 60000 eine Verzögerung von einer Minute.
- **`display.kbdact`** gibt an, ob Aktivitäten auf der Tastatur wieder zur Rückkehr in den Normalmodus führen. Normalerweise will man das.
- **`display.outact`** gibt an, ob Ausgaben auf dem Bildschirm wieder zur Rückkehr in den Normalmodus führen.

Du kannst diese Variablen auf der Kommandozeile mittels des wsconsctl(8) Befehls setzen:

```
# wsconsctl -w display.screen_off=60000
display.screen_off -> 60000
```

oder sie dauerhaft setzen, indem du sie in der Datei */etc/wsconsctl.conf* einträgst, so dass sie beim nächsten Bootvorgang aktiviert werden:

```
display.vblank=on               # enable vertical sync blank
display.screen_off=600000       # set screen blank timeout to 10 minutes
display.kbdact=on               # Restore screen on keyboard input
display.outact=off              # Restore screen on display output
```

Der Bildschirmschoner wird entweder aktiv, wenn `display.kbdact` oder `display.outact` auf "`on`" gesetzt sind.

7.9 - ALLES WAS ICH TIPPE IST IN GROSSBUCHSTABEN!

Tatsächlich ist das ein Feature, kein Fehler.

Nahezu alle Unix Befehle und Benutzernamen werden in Kleinbuchstaben eingegeben. Trotzdem waren einige alte Terminals nur in der Lage, Großbuchstaben zu benutzen, was es für sie unmöglich machte, für Unix benutzt werden zu werden. Als Workaround hat getty(8), wenn du deinen Usernamen komplett in Großbuchstaben eingegeben hast, angenommen, dein Terminal sei "lowercase challenged", und hat einfach alles, was du getippt hast, als Kleinbuchstaben interpretiert, es allerdings alles in Großbuchstaben ausgegeben. Wenn du ein gemischtes oder nur-Großbuchstaben-Passwort hast, kannst du dich nicht einloggen.

Wenn du STRG-D am Login Prompt drückst, wird getty(8) beendet und init(8) wird einen neuen starten, der dann Groß- und Kleinbuchstaben sauber akzeptiert.

[FAQ Index] [Zum Kapitel 6 - Netzwerken] [Zum Kapitel 8 - Allgemeine Fragen]


www@openbsd.org

$OpenBSD: faq7.html,v 1.45 2005/02/12 20:36:53 jufi Exp $

OpenBSD

[FAQ Index] [Zum Kapitel 8 - Allgemeine Fragen] [Zum Kapitel 10 - Systemverwaltung]

9 - Zu OpenBSD migrieren

Inhaltsverzeichnis

Weitere Informationen für Linux Benutzer gibt es unter http://sites.inka.de/mips/unix/bsdlinux.html.

9.1 - Tipps für Nutzer von anderen Unix-ähnlichen Betriebssystemen

Während OpenBSD ein sehr traditionelles Unix-ähnliches Betriebssystem ist und daher sehr vertraut für diejenigen ist, die bisher schon andere Unix-ähnliche Betriebssysteme verwendet haben, gibt es einige wichtige Unterschiede. Neue Benutzer von OpenBSD müssen ihre eigene Erfahrung einschätzen: wenn deine Erfahrung mit Unix aus dem Experimentieren mit einigen Varianten von Linux besteht, wirst du OpenBSD als "merkwürdig" empfinden. Im Übrigen wirkt Linux recht merkwürdig auf jene, die mit OpenBSD angefangen haben. Du musst den Unterschied zwischen "Standard" und deiner Erfahrung erkennen.

Wenn du Unix von einigen der guten Bücher gelernt hast, die generell über Unix handeln, die "Unix Philosophie" verstanden hast und dann dein Wissen auf eine bestimmte Plattform erweitert hast, wirst du OpenBSD als ein sehr "wahres" und vertrautes Unix wahrnehmen. Wenn du Unix durch einen "Tippe dies ein um das zu machen" Prozess oder durch ein Buch wie zum Beispiel "Lerne PinkBeenie v8.3 in 31.4 Stunden" und dir selbst dann eingeredet hast, dass du "Unix kennst", wirst du äußerst wahrscheinlich OpenBSD als sehr anders ansehen.

Ein sehr wichtiger Unterschied zwischen OpenBSD und vielen anderen Betriebssystemen ist die Dokumentation. OpenBSD Entwickler sind sehr stolz auf die System Manual Seiten. Die Manual Seiten sind *die* maßgebliche Quelle der OpenBSD Dokumentation -- nicht diese FAQ, nicht von unabhängigen dritten Partien verwaltete Seiten, nicht "HOWTO"s, etc. Wenn Entwickler eine Änderung am System durchführen, wird von ihnen erwartet, dass sie die Manual Seiten neben ihren Änderungen aktualisieren, nicht "später" oder "wenn sie dafür Zeit finden" oder "wenn jemand sich beschwert". Eine Manual Seite existiert für so gut wie jedes Programm, Treiber, Konfigurationsdatei und so weiter aus dem Basis-System. Es wird erwartet, dass der Benutzer die Manual Seiten durchsucht, bevor er in den Mailinglisten nach Hilfe fragt.

Hier sind einige häufig aufgefallene Unterschiede zwischen OpenBSD und anderen Unix Varianten.

- OpenBSD ist ein recht pures "BSD-Style" Unix, das dem 4.4BSD Design sehr genau folgt. Linux und SCO Unix sind "System V" style Systeme. Einige Unix-ähnliche Systeme (dazu zählen auch einige Linux Distributionen) mixen einige SysV und BSD Charakteristiken. Ein typischer Platz, wo dies Verwirrung hervorruft, ist bei den Startup Skripten, OpenBSD verwendet den traditionellen BSD4.4-style rc(8) Style.
- OpenBSD ist ein vollständiges *System*, dessen Absicht es ist, synchron gehalten zu werden. Es ist nicht ein "Kernel mit Anwendungen", die separat voneinander aktualisiert werden können. Fehlschläge, dein System (Kernel, Benutzeranwendungen und Programme) synchron zu halten, resultieren in schlimmen

Dingen, die passieren werden.

- Da viele Applikationen nicht dafür entwickelt worden sind, direkt in einer OpenBSD Umgebung zu kompilieren und ausgeführt zu werden, hat OpenBSD einen Ports Tree, ein System in dem Benutzer auf einfachem Wege Code beziehen, diesen für OpenBSD patchen, Abhängigkeiten installieren, ihn kompilieren, installieren und auf einem standardisierten und verwaltbaren Weg deinstallieren können. Vor-kompilierte Packages werden vom OpenBSD Ports Team erstellt und zur Verfügung gestellt. Benutzer werden gebeten diese Packages zu benutzen, statt ihre eigenen zu kompilieren.
- OpenBSD verwendet CVS für Source Änderungen. OpenBSD leistete Pionierarbeit für anonymous CVS, welches es jedermann ermöglicht, den kompletten Source Tree für jegliche Version von OpenBSD (von 2.0 bis ‚current' und alle Revisionen von allen Dateien zwischen ihnen) jederzeit zu extrahieren und du kannst auf die aktuellsten Änderungen zugreifen, innerhalb von Stunden nach derren Einbindung. Es gibt auch ein sehr bequemes und einfach zu benutzendes Web Interface zum CVS.
- OpenBSD produziert alle sechs Monate nach einem vordefinierten Zeitplan ein offizielles Release, das auf CD und per FTP verfügbar ist. Snapshots für alle unterstützten Plattformen werden semi-regulär mit dem aktuellen Entwicklungscode erstellt. Es ist das Ziel, dass der Source Tree jederzeit vollständig erzeugbar und das resultierende System nutzbar ist. Der Tree ist gelegentlich kaputt, aber dies ist ein ungewöhnlicher Vorfall, der schnell behoben wird und nicht etwas ist, dessen Fortsetzen geduldet wird.
- OpenBSD beinhaltet starke Kryptographie, die nicht in Betriebssysteme eingefügt werden können, die in einigen Ländern entwickelt werden.
- OpenBSD wurde schweren und kontinuierlichen Sicherheitsüberprüfungen unterzogen, um die Qualität (und daher, Sicherheit) des Codes zu gewährleisten.
- OpenBSDs Kernel ist `/bsd`.
- Der Name der Festplatten ist gewöhnlicherweise `/dev/wd` (IDE) und `/dev/sd` (SCSI oder Geräte, die SCSI Platten emulieren).
- /sbin/ifconfig mit keinen Argumenten gibt unter Linux den Status aller Interfaces aus, aber unter OpenBSD (und vielen anderen Betriebssystemen) benötigst du die `-a` Option.
- /sbin/route mit keinen Argumenten gibt unter Linux den Status von allen aktiven Routen aus, unter OpenBSD (und vielen anderen Betriebssystemen) benötigst du den "*show*" Parameter oder führe ein "`netstat -r`" aus.
- OpenBSD unterstützt NICHT Journaling Dateisysteme wie ReiserFS, IBMs JFS oder SGIs XFS. Stattdessen verwenden wir die Soft Updates Funktion des bereits sehr robusten ‚Unix Fast File System' (FFS), um die Ziele der Geschwindigkeit und Stabilität zu erreichen.
- OpenBSD wird mit Packet Filter (PF), nicht ipfw, ipchains, netfilter, iptables oder ipf ausgeliefert. Das bedeutet, dass Network Address Translation (als IP-Masquerading unter Linux bekannt), ‚queuing' und das Filtern durch pfctl(8), pf(4) und pf.conf(5) ausgeführt werden. Siehe das PF Benutzerhandbuch für detaillierte Konfigurationsinformationen.
- Die Interface Adresse ist in /etc/hostname.<interfacename> gespeichert (zum Beispiel `/etc/hostname.dc0` für eine Netzwerkkarte, die die dc(4) Treiber verwendet). Es kann Hostnamen (aufgelöst in /etc/hosts) beinhalten, statt einer IP Adresse.
- Der Maschinen Name ist in /etc/myname.
- Das standardmäßige Gateway ist in /etc/mygate.
- OpenBSDs standardmäßige Benutzershell ist /bin/sh, welche pdksh ist, die Public Domain Korn Shell im Bourne Shell Modus. Andere eingefügte Shells sind csh (die standardmäßige Shell für *root*) und ksh. Shells wie zum Beispiel *bash* und *tcsh* können als Packages oder von Ports hinzugefügt werden. Benutzern, die *bash* gewohnt sind, wird geraten, ksh(1) auszuprobieren, bevor sie sich *bash* auf ihre System laden -- es macht das, was viele Leute von der *bash* erwarten.
- Passwortverwaltung unter OpenBSD ist anders als die Passwortverwaltung unter einigen anderen Unix-ähnlichen Betriebssystemen. Das tatsächliche Passwort wird in der Datei master.passwd(5) gespeichert, welche nur von *root* gelesen werden kann. Diese Datei sollte nur mit dem vipw Programm geändert werden.

- Devices werden nach dem Treiber benannt, nicht nach dem Typ. Zum Beispiel gibt es keine eth* Devices. Es wäre ne0 für eine NE2000 Ethernet Karte und xl0 für eine 3Com Etherlink XL oder einem Fast Etherlink XL Ethernet Device, etc. Alle diese Treiber haben Manual Seiten unter Sektion 4. Um also mehr Informationen über die Nachrichten zu erfahren, die dein 3c905 Treiber ausspuckt, kannst du "`man 4 xl`" ausführen.
- OpenBSD/i386 verwendet ein "zwei Schichten" Plattenpartitionierungssystem, bei dem die erste Schicht die fdisk, BIOS-sichtbare Partition ist, die den meisten IBM kompatiblen Computerbenutzern bekannt ist. Die zweite Schicht ist das Disklabel, ein traditionelles BSD Partitionierungssystem. OpenBSD unterstützt bis zu 15 Disklabel Partitionen auf einer Platte, die alle in einer fdisk Partition liegen. Dies erlaubt OpenBSD/i386 neben anderen Betriebssytemen zu existieren, zu denen auch andere Unix-ähnliche Betriebssysteme zählen. OpenBSD muss eine der vier "primären" Partitionen sein.
- Einige andere Betriebssysteme erwarten, dass du deinen Kernel für deine Maschine anpasst. Von OpenBSD Benutzern wird erwartet, dass sie einfach den standardmäßigen GENERIC Kernel verwenden, der von den Entwicklern bereitgestellt und getestet wurde. Benutzer, die versuchen, ihren Kernel "anzupassen" oder zu "optimieren", verursachen normalerweise mehr Probleme als sie lösen und werden von den Entwicklern nicht unterstützt.
- OpenBSD arbeitet hart daran, die Lizenz Bestimmungen und Sicherheit des Projektes aufrecht zu erhalten. Aus diesem Grund werden einige neue Versionen von bestimmten Softwareprodukten, die entweder der Lizenz oder den Sicherheitszielen des Projektes nicht entsprechen, noch nicht oder auch niemals in OpenBSD integriert. Sicherheit und freie Lizenzen werden niemals nach hinten gestellt, um die größte Versionsnummer zu haben.

9.2 - Dual Boot von Linux und OpenBSD

Ja! Es ist machbar!

Lies INSTALL.linux.

9.3 - Deine Linux (oder andere Sixth Edition-artige) Passwort-Datei in BSD-Style konvertieren.

Finde zuerst heraus, ob deine Linux Passwortdatei ‚geshadowed' wird oder nicht. Wenn ja, besorge dir John the Ripper und benutze das ‚unshadow' Werkzeug, das darin enthalten ist, um deine `passwd` und `shadow` Dateien in eine Sixth Edition-artige Datei zu verwandeln.

In deiner Linux Passwort-Datei, wir nennen sie `linux_passwd`, musst du nun ::0:0 zwischen den Feldern 4 und 7 einfügen. Awk kann das für dich erledigen.

```
# cat linux_passwd | awk -F : '{printf("%s:%s:%s:%s::0:0:%s:%s:%s\n", \
> $1,$2,$3,$4,$5,$6,$7); }' > new_passwd
```

An diesem Punkt angelangt, solltest du jetzt die `new_passwd` Datei ändern und root und andere Systemeinträge löschen, die bereits in deiner OpenBSD Passwortdatei enthalten sind, oder die es in OpenBSD gar nicht gibt (und zwar alle davon). Stelle auch sicher, dass es keine doppelten Usernamen oder User-IDs zwischen `new_passwd` und dem `/etc/passwd` auf deiner OpenBSD-Kiste gibt. Der einfachste Weg ist, einfach mit einer frischen `/etc/passwd` anzufangen.

```
# cat new_passwd >> /etc/master.passwd
# pwd_mkdb -p /etc/master.passwd
```

Der letzte Schritt, `pwd_mkdb`, ist notwendig, um die /etc/spwd.db und /etc/pwd.db Dateien neu zu erzeugen. Es erzeugt auch eine Sixth Edition-artige Passwortdatei (ohne verschlüsselte Passwörter) unter /etc/passwd für die Programme, die darauf zugreifen. OpenBSD benutzt eine stärkere Verschlüsselung für Passwörter, nämlich ‚blowfish', die man wohl kaum auf Systemen mit vollen Sixth Edition-artigen Passwortdateien finden wird. Um

zu dieser stärkeren Verschlüsselung zu wechseln, müssen die User einfach ‚passwd' benutzen (bzw. tippen) und ihr Passwort ändern. Das neu eingegebene Passwort wird mit deiner Standardeinstelung verschlüsselt (normalerweise ‚blowfish', es sei denn, du hättest /etc/login.conf verändert). Oder du machst es als *root* mit `passwd Benutzername`.

9.4 - Linux Binaries unter OpenBSD ausführen

OpenBSD/i386 ist in der Lage, Linux Binaries auszuführen, wenn der Kernel mit der COMPAT_LINUX Option kompiliert wurde und die Laufzeit sysctl kern.emul.linux ebenfalls gesetzt ist. Wenn du den GENERIC Kernel verwendest (den du verwenden solltest), ist COMPAT_LINUX bereits aktiviert und du musst nur noch folgendes tun:

```
# sysctl kern.emul.linux=1
```

Damit dies automatisch jedes Mal ausgeführt wird, wenn der Computer bootet, entferne das # (Kommentar) Zeichen am Anfang der Zeile

```
#kern.emul.linux=1      # enable running Linux binaries
```

in /etc/sysctl.conf, so dass die Zeile wie folgt aussieht

```
kern.emul.linux=1       # enable running Linux binaries
```

und starte dein System neu, damit die Änderung wirksam wird.

Um Linux Binaries zum Laufen zu bringen, die nicht statisch gelinkt sind (und die meisten sind es nicht), solltest du den Anweisungen auf der compat_linux(8) Manual Seite folgen.

Ein einfacher Weg, die meisten nützlichen Linux Bibliotheken zu kriegen, ist, den `redhat/base` Port aus der Ports Collection zu installieren. Um mehr über die Ports Collection zu erfahren, lies einfach FAQ 8 - Ports. Wenn du den Ports Tree schon installiert hast, benutze diese Kommandos, um die Linux Bibliotheken zu installieren:

```
# cd /usr/ports/emulators/redhat/base
# make install
```

9.5 - Auf deine Linux Dateien von OpenBSD aus zugreifen

OpenBSD unterstützt das EXT2FS Dateisystem. Benutze

```
# disklabel Platte
```

(wobei ***Platte*** der Device Name deiner Platte ist, z.B. **wd0**), um zu überprüfen, was OpenBSD denkt, was deine Linux Partition ist. Trotzdem, verwende **nicht** disklabel (8) oder fdisk (8), um irgendwelche Änderungen am Disklabel durchzuführen.

Für weitere Informationen über die Benutzung von disklabel lies FAQ 14 - Disklabel.

[FAQ Index] [Zum Kapitel 8 - Allgemeine Fragen] [Zum Kapitel 10 - Systemverwaltung]

---

www@openbsd.org

$OpenBSD: faq9.html,v 1.36 2005/01/18 17:17:47 jufi Exp $

OpenBSD

[FAQ Index] [Zum Kapitel 10 - Systemverwaltung] [Zum Kapitel 12 - Plattform-spezifische Fragen]

11 - Leistungs Tuning

Inhaltsverzeichnis

11.1 - Festplatten E/A

Festplatten E/A Geschwindigkeit ist ein wichtiger Faktor in der Gesamtgeschwindigkeit deines Computers. Sie wird umso wichtiger, wenn dein Computer eine Multi-User-Umgebung beheimatet (User aller Arten, von solchen, die sich einloggen, bis zu denen die Serverdienste nutzen). Datenspeicher brauchen ständige Aufmerksamkeit. Insbesondere, wenn deine Partition überläuft oder deine Platten versagen. OpenBSD kennt verschiedene Optionen, um die Geschwindigkeit deiner Festplattenoperationen zu erhöhen und Fehlertoleranz zu bieten.

Inhaltsverzeichnis

11.1.1 - CCD

Die erste Option ist die Benutzung des ccd(4), des Concatenated Disk Driver. Dieser erlaubt dir, mehrere Partitionen in eine virtuelle Platte zu verwandeln (und damit kannst dafür sorgen, dass mehrere Festplatten wie eine einzige aussehen). Dieses Konzept ist ähnlich wie das von LVM (logical volume management), das in mehreren kommerziellen Unix-Arten zu finden ist.

Wenn du GENERIC benutzt, ist ccd bereits eingeschaltet (in `/usr/src/sys/conf/GENERIC`). Wenn du einen veränderten Kernel benutzt, musst du ihn vielleicht wieder in deine Kernel-Konfiguration einfügen. Wie auch immer, auf jeden Fall muss sich eine Zeile wie die folgende in deiner Konfigurationsdatei befinden:

```
pseudo-device   ccd     4       # concatenated disk devices
```

Das obige Beispiel gibt dir bis zu 4 ccd Devices (virtuelle Platten). Jetzt musst du festlegen, welche Partitionen auf deinen realen Festplatten du in den ccd einbinden willst. Benutze disklabel, um diese Partitionen als ‚ccd'-Typ zu markieren. Auf einigen Architekturen erlaubt dir disklabel das vielleicht nicht. In diesem Fall markiere sie einfach als ‚ffs'.

Wenn du ccd dazu benutzt, um mittels striping Performance zu gewinnen, solltest du wissen, dass du keine optimale Leistung bekommst, bis du das gleiche Festplatten-Modell mit den gleichen disklabel Einstellungen benutzt.

Editiere /etc/ccd.conf, bis sie etwa so aussieht: (Mehr Informationen über das Konfigurieren von ccd findest du unter ccdconfig(8))

```
# Configuration file for concatenated disk devices
#
# ccd   ileave  flags   component devices
ccd0   16      none    /dev/sd2e /dev/sd3e
```

Um die Änderungen wirksam zu machen, führe das hier aus:

```
# ccdconfig -C
```

Solange /etc/ccd.conf existiert, wird sich ccd automatisch beim Booten konfigurieren. Jetzt hast du eine neue Festplatte, ccd0, eine Kombination von /dev/sd2e und /dev/sd3e. Benutze disklabel einfach wie gewöhnlich, um die Partition oder Partitionen zu erzeugen, die du benutzen willst. Nutze erneut die ‚c' Partition nicht, um darauf irgendetwas zu speichern. Stelle sicher, dass deine benutzten Partitionen mindestens einen Zylinder vom Anfang der Disk weg ist.

11.1.2 - RAID

Eine weitere Lösung ist raid(4), wofür du raidctl(8) nutzen musst, um deine RAID Geräte zu kontrollieren. OpenBSDs RAID basiert auf Greg Osters NetBSD Port der CMU RAIDframe Software. OpenBSD hat Unterstützung für die RAID-Level 0, 1, 4 und 5. Für raid muss, wie auch bei ccd, Unterstützung im KERNEL sein. Diese Treiber-Unterstützung für RAID ist im Gegensatz zu ccd allerdings nicht im GENERIC-Kernel enthalten, also muss sie extra in deinen Kernel einkompiliert werden (RAID-Unterstützung vergrößert deinen i386 Kernel um gute 500k).

```
pseudo-device   raid   4       # RAIDframe disk device
```

Ein RAID aufzusetzen ist mit einigen Betriebssystemen verwirrend und schmerzhaft, um es sanft auszudrücken. Nicht jedoch mit RAIDframe. Lies die raid(4) und raidctl(8) Manual Seiten für die kompletten Details. Es gibt dafür viele Optionen und mögliche Konfigurationen, und ein detaillierter Überblick sprengt den Rahmen dieses Dokumentes.

11.1.3 - Soft Updates

Ein weiteres Tool zum Erhöhen der Systemgeschwindigkeit sind Soft Updates. Eine der langsamsten Operationen im traditionellen BSD Dateisystem ist das Updaten der Metainfos (was unter anderem immer dann geschieht, wenn du Dateien oder Verzeichnisse erzeugst oder löschst). Soft Updates versucht die Metainfo im RAM upzudaten, statt jedes einzelne Metainfo-Update auf die Platte zu schreiben. Ein weiterer Nebeneffekt ist, dass die Metainfos auf der Festplatte immer auf dem aktuellen Stand sind. Das heißt, ein Systemcrash sollte kein fsck(8) beim folgende Booten benötigen, sondern eine einfache Hintergrund-Version von fsck, die Änderungen an den Metainfos im RAM macht (a la softupdates). Das heißt, dass Reboots viel schneller sind, da nicht mehr auf fsck gewartet werden muss! (OpenBSD hat dieses Feature leider noch nicht.) Mehr über Soft Updates findest du im Soft Updates FAQ Eintrag.

11.1.4 - Größe des namei() Cache

Hinweis: Vorher hat die options(4) Manual Seite empfohlen, die `NVNODE=integer` Kernel Option zu setzen. Das wird nicht mehr empfohlen; du solltest stattdessen das sysctl(8) Kommando benutzen.

Die name-to-inode Übersetzung (a.k.a., namei() Cache) kontrolliert die Geschwindigkeit der pathname zu inode(5) Übersetzung. Ein sinnvoller Weg zum Herausfinden der passenden Größe des Cache wäre, eine große Anzahl von namei() ‚cache misses', die man mit einem Tool wie systat(1) messen könnte, vorausgesetzt, eine Untersuchung des momentanen berechneten Wertes mittels sysctl(8), (das diesen Parameter "`kern.maxvnodes`" nennt) und diesen Wert zu vergrößern, bis sich entweder die namei() Cache ‚hit rate' verbessert, oder es bewiesen ist, dass das System nicht wesentlich von einer Erhöhung der Größe des namei() Cache profitiert. Nachdem der Wert festgestellt wurde, kannst du ihn für die nächsten Systemstarts mit sysctl.conf(5) setzen.

11.2 - Hardware Auswahl

(Hinweis - diese Sektion dreht sich fast ausschließlich um die i386 oder PC Architektur. Andere Architekturen geben dir sozusagen keine so große Auswahl!)

Die Leistung deiner Anwendungen hängt stark von deinem Betriebssystem und den Fähigkeiten ab, die es bereitstellt. Das mag ein Grund dafür sein, dass du OpenBSD benutzt. Die Leistung deiner Anwendungen hängt aber auch stark von deiner Hardware ab. Für viele Leute ist das Preis-Leistungs-Verhältnis eines brandneuen PCs mit einem Intel Pentium IV oder AMD Athlon Prozessors viel besser als das Preis-Leistungs-Verhältnis einer Sun UltraSPARC 60! Der Preis von OpenBSD ist natürlich unschlagbar.

Wenn du einen neuen PC kaufen willst, ob nun in einem Komplettangebot, oder Einzelteil für Einzelteil, solltest du sicherstellen, dass du unbedingt nur zuverlässige Teile bekommst. In der Welt der PCs ist das leichter gesagt als getan. **Schlechte oder sonstige unzuverlässige oder unpassende Teile können dazu führen, dass OpenBSD schlecht läuft und oft abstürzt**. Der beste Rat, den wir geben können, ist, vorsichtig zu sein und Marken und Teile zu kaufen, die von jemandem empfohlen werden, dem du trauen kannst. Wenn du nur auf den Preis eines PCs achtest, wirst du wahrscheinlich auch an Qualität verlieren!

Es gibt ein paar Dinge, die dir helfen können, die maximale Leistung aus deiner Hardware zu holen:

- **Benutze mehrere Festplatten.** Statt nur eine große Platte zu kaufen, kaufe mehrere kleine Platten. Wenn das auch mehr kostet, wird es doch die Last auf mehrere Spindeln verteilen und somit die Zeitspanne verringern, die ein Datenzugriff benötigt. Außerdem kannst du mit mehreren Platten auch mehr Zuverlässigkeit und schnelleren Datenzugriff mit RAID bekommen.
- **Benutze SCSI, wenn du hohe Festplatten-EA-Geschwindigkeit brauchst.** IDE Festplatten laufen normalerweise mit 5400 RPM bis 7200 RPM. Selbst bei hochwertigen IDE Platten ist es manchmal zu viel verlangt, wenn man mehr als 15 bis 20 MB pro Sekunde an Datendurchsatz von einer einzelnen Platte verlangt. Mit hochwertigen SCSI-Platten (10k RPM oder 15k RPM) kannst du mehr Durchsatz bekommen. Im Gegensatz dazu ist es eine Verschwendung von Geld, wenn du mittlere oder langsame SCSI-Platten benutzt, da dann IDE die gleiche oder bessere Leistung bringt.

  Wenn du einen Server und mehr als ein Laufwerk brauchst, solltest du über SCSI nachdenken. IDE beschränkt dich auf zwei Platten pro Controller. Gleichzeitige Zugriffe auf diese Platten haben vermutlich einen negativen Effekt auf die E/A Leistung dieser Platten. Mit Wide SCSI kannst du 15 Platten pro Controller anschließen und es hat bessere Unterstützung für gleichzeitigen Zugriff als IDE.
- **Benutze SDRAM statt DRAM.** Diese Option trifft fast nur auf PCs zu. Bei den meisten anderen Architekturen hast du keinerlei Auswahl welche Art von RAM du benutzen kannst. Bei den meisten PCs schon. Mit SDRAM bekommst du eine bessere Leistung als mit DRAMs (SIMMs). Wenn dein System RDRAM unterstützt oder vielleicht DDR oder eine andere neue Art von RAM bist du sogar noch besser dran.
- **Benutze ECC oder Parity RAM.** Parity fügt einen Mechanismus hinzu, der prüft, ob die Daten im RAM noch in Ordnung sind. ECC baut das noch dahingehend aus, dass es versucht, Fehler bei einzelnen Bits automatisch zu korrigieren. Diese Option gibt es wieder fast nur bei PCs. Die meisten anderen Architekturen brauchen einfach ECC oder Parity RAM. Einige nicht-PC-Computer booten nicht einmal mit nicht-Parity-RAM. Wenn du kein ECC/Parity RAM benutzt, kann es zu Daten-Korrumpierung und anderen Abnormitäten kommen. Einige Herstelle von "billigem PC RAM" stellen nicht einmal eine ECC-Variante her! Das hilft dir, sie zu vermeiden! PC Hersteller verkaufen oftmals mehrere Produktlinien, in "Server" und "Workstations" aufgeteilt. Die Server haben Parity (und jetzt ECC) seit vielen Jahren beinhaltet. Unix-Workstation-Hersteller benutzen Parity (und nun ECC) seit vielen Jahren in all ihren Produktlinien.
- **Vermeide ISA-Karten.** Während die meisten Leute ISA-Karten schon deshalb meiden, weil sie veraltet und zudem noch schwer zu konfigurieren sind, gibt es aber trotzdem noch eine ganze Menge davon. Wenn du den ISA Bus für deine Festplatte oder Netzwerkkarte benutzt (oder noch schlimmer, für beides) denke daran, dass der ISA Bus vermutlich ein Flaschenhals ist. Wenn du Geschwindigkeit brauchst, benutze PCI. Natürlich gibt es noch zahllose ISA-Karten, die einfach gut funktionieren. Unglücklicherweise sind das meist Soundkarten oder solche für serielle Ports.
- **Vermeide billige PCI Netzwerkkarten.** OpenBSD unterstützt eine ganze Menge von billigen PCI Netzwerkkarten. Diese Karten funktionieren prima in einfachen Heim-Systemen, oder solchen mit wenig oder moderater Netzwerk-Last im Geschäfts- oder Forschungs-Bereich. Aber, wenn du hohen Durchsatz brauchst, und wenig Belastung deines Servers, bist du mit einer Qualitäts-Netzwerkkarte besser dran. Unglücklicherweise sind einige Serien von teuren Marken-Herstellern nicht besser als die billigen Karten. Gigabit Karten sind wegen dem besseren Buffering meist leistungsfähiger als 10Mbps/100Mbps Adapter,

selbst wenn sie mit langsameren Netzwerkgeschwindigkeiten genutzt werden.

11.3 - Wieso benutzen wir keine ‚async mounts'?

Frage: "Ich gebe einfach ein "mount -u -o async /" ein, was ein Paket, was ich brauche, benutzbar macht. (das darauf besteht alle paar Momente ein paar hundert Dateien zu ändern.) Wieso wird ‚async mounting' abgelehnt und ist nicht standardmäßig aktiviert (wie in manchen anderen Unixen)? Wäre das nicht ein einfacherer, und daher auch ein sichererer Weg, die Leistung mancher Applikation zu erhöhen?"

Antwort: "Asynchrone mounts sind tatsächlich schneller als synchrone mounts, aber sie sind unsicherer. Was passiert im Falle eine Stromausfalls? Oder bei einem Hardwareproblem? Die Suche nach Geschwindigkeit darf nicht auf Kosten von Stabilität und Zuverlässigkeit des Systems gehen. Siehe auch die Manual Seite von mount(8)."

```
async   All I/O to the file system should be done asynchronously.
        This is a dangerous flag to set since it does not guaran-
        tee to keep a consistent file system structure on the
        disk.  You should not use this flag unless you are pre-
        pared to recreate the file system should your system
        crash.  The most common use of this flag is to speed up
        restore(8) where it can give a factor of two speed in-
        crease.
```

Auf der anderen Seite, wenn du sowieso nur mit temporären Daten umgehst, die du nach einem Crash wieder rekonstruieren kannst, kannst du mehr Geschwindigkeit erhalten, indem du eine separate Partition nur für diese Daten benutzt, die asynchron gemountet ist. Tue das aber *nur, wenn* dir der Verlust aller Daten in der Partition nach irgendeinem Problem nichts ausmacht. Daher sind mfs(8) Partitionen asynchron gemountet, weil sie ja nach jedem Reboot sowieso gelöscht und neu erzeugt werden.

11.4 - Deine Monitor-Auflösung unter XFree86 tunen

Hinweis: Die meisten Benutzer müssen sich NICHT um die Erstellung einer ModeLine in modernen Versionen von X sorgen. TROTZDEM, manchmal ist es in ungewöhnlichen Situationen notwendig.

Es ist durchaus mit vielen multi-Sync-Monitoren möglich, einen X Server in einer akzeptablen Auflösung zum Laufen zu kriegen. Mit den Standard-Konfigurations-Werkzeugen xf86config oder XF86Setup ist es aber recht schwierig, ein gutes Ergebnis zu erhalten. Einer der schmerzvolleren Punkte ist es, deinen Monitor zur gewünschten Auflösung zu bewegen, und dann eine vertikale Scan-Rate von mindestens 72-75 Hz zu bekommen, eine Rate, bei der das Bildschirmgeflacker wesentlich geringer sichtbar für menschliche Augen ist. Was passiert aber, wenn du die vertikale Scan-Rate sehr niedrig einstellst? So könntest du den Bildschirm zum Beispiel ohne Flackern auf Video filmen, aber auch dazu sind die Methoden mit den Standard-Werkzeugen von XFree86 eher nicht-intuitiv.

Schlussendlich ist es bei den Auflösungen, (800x600, 1024x768, 1152x900,1280x1024), die die meisten Leute heute mit preiswerten VGA-Monitoren benutzen (zumindest mit neueren Modellen) bestens möglich, vertikale Wiederholungsraten von 85 Hz und mehr zu bekommen, um ein wirklich klares und ansehnliches Bild zu erhalten. Der XFree86 X Server hat einen Mechanismus, der dir erlaubt, im Detail den Grafik-Modus zu beschreiben, den du benutzen willst, dies nennt sich ModeLine. Eine ModeLine hat vier Sektionen, eine einzelne Nummer für die Pixel Clock, vier Nummern für horizontales Timing, vier Nummern für vertikales Timing und eine optionale Sektion mit einer Liste von Flags für weitere Charakteristika wie etwa den Modus (z.B. Interlace, DoubleScan und weitere... mehr Details gibt es in der XF86Config(5) Manual Seite)

Das Erzeugen einer ModeLine ist eine schwarze Kunst. Glücklicherweise gibt es mehrere Skripte, die das für dich erledigen können. Eines davon ist der Colas XFree86 ModeLine Generator. Ein weiteres ist The XFree86 Modeline Generator, der bei SourceForge gehostet wird, und es gibt weitere bei Freshmeat. Bevor du diese ModeLine-Generatoren benutzen kannst, musst du die vertikalen und horizontalen sync Grenzen für deinen Monitor herausfinden. Diese Angaben finden sich oftmals im Handbuch oder auf der Webseite des Monitor-Herstellers. Wenn du sie dort nicht finden kannst, suche einfach im Web nach deinem Modell und Hersteller, viele Leute waren so freundlich, Listen mit den entsprechenden Angaben zu erstellen.

Sagen wir zum Beispiel, du hättest einen Dell D1226H Monitor. Du hast auf Dells Website herausgefunden, dass er einen Bereich von 30-95 kHz horizontal und 50-160 Hz vertikal hat. Besuche die ModeLine Generator Page, und

gib diese Informationen ein. Als nächstes musst du die minimale vertical scan rate eingeben, die du haben willst. Jede Rate ab 72 Hz und größer sollte im Allgemeinen wenig flackern. Je mehr, desto besser wird das Bild.

Mit all diesen Informationen wird das Skript eine ModeLine für jede mögliche 4x3 Auflösung generieren, die dein Monitor unterstützen kann. Wenn jemand die Dell Spezifikationen von oben und eine minimale vertikale Rate von 75 Hz eingibt, gibt das Skript etwas ähnliches wie das folgende aus:

```
ModeLine "320x240" 20.07 320 336 416 448 240 242 254 280 #160Hz
ModeLine "328x246" 20.86 328 344 424 456 246 248 260 286 #160Hz
...
ModeLine "816x612" 107.39 816 856 1056 1136 612 614 626 652 #145Hz
ModeLine "824x618" 108.39 824 864 1064 1144 618 620 632 658 #144Hz
ModeLine "832x624" 109.38 832 872 1072 1152 624 626 638 664 #143Hz
...
ModeLine "840x630" 109.58 840 880 1080 1160 630 632 644 670 #141Hz
ModeLine "848x636" 110.54 848 888 1088 1168 636 638 650 676 #140Hz
...
ModeLine "1048x786" 136.02 1048 1096 1336 1432 786 788 800 826 #115Hz
ModeLine "1056x792" 136.58 1056 1104 1344 1440 792 794 806 832 #114Hz
ModeLine "1064x798" 137.11 1064 1112 1352 1448 798 800 812 838 #113Hz
...
ModeLine "1432x1074" 184.07 1432 1496 1816 1944 1074 1076 1088 1114 #85Hz
ModeLine "1576x1182" 199.86 1576 1648 2008 2152 1182 1184 1196 1222 #76Hz
ModeLine "1584x1188" 198.93 1584 1656 2016 2160 1188 1190 1202 1228 #75Hz
```

Dieser Monitor gibt nun vor, 1600x1200 @ 75 Hz machen zu können, aber das Skript sagt nicht, dass das innerhalb der 75 Hz sei. Wenn du also exakt 1600x1200 haben willst, geh ein wenig mit deiner minimalen vertikalen Rate herunter. (Hier z.B. kannst du bis 70 Hz heruntergehen)

```
ModeLine "1592x1194" 197.97 1592 1664 2024 2168 1194 1196 1208 1234 #74Hz
ModeLine "1600x1200" 199.67 1600 1672 2032 2176 1200 1202 1214 1240 #74Hz
ModeLine "1608x1206" 198.65 1608 1680 2040 2184 1206 1208 1220 1246 #73Hz
ModeLine "1616x1212" 197.59 1616 1688 2048 2192 1212 1214 1226 1252 #72Hz
ModeLine "1624x1218" 199.26 1624 1696 2056 2200 1218 1220 1232 1258 #72Hz
ModeLine "1632x1224" 198.15 1632 1704 2064 2208 1224 1226 1238 1264 #71Hz
ModeLine "1640x1230" 199.81 1640 1712 2072 2216 1230 1232 1244 1270 #71Hz
ModeLine "1648x1236" 198.64 1648 1720 2080 2224 1236 1238 1250 1276 #70Hz
```

Hier sehen wir, dass der Monitor tatsächlich 1600x1200 @ 74 Hz macht, wenn die Bandbreite (dot clock) auf 200MHz begrenzt ist. Setze die Bandbreite gemäß der Grenzen, die vom Monitor definiert werden.

Nachdem du einmal die ModeLines hast, schreibe sie in deine /etc/X11/XF86Config Datei. Kommentiere die alten ModeLines aus, so dass du sie noch benutzen kannst, falls die neuen nicht funktionieren. Als nächstes wähle aus, mit welcher Auflösung du nun arbeiten willst. Als erstes musst du nun herausfinden, ob X im "accelerated mode" läuft, oder nicht (das tut es mit den meisten Grafik-Karten), so dass du auch weißt, welche `"Screen"` Sektion der XF86Config-Datei du modifizieren musst. Alternativ kannst du natürlich einfach alle Screen-Sektionen modifizieren.

```
Section "Screen"
   Driver          "Accel"
   Device          "Primary Card"
   Monitor         "Primary Monitor"
   DefaultColorDepth 32
   SubSection "Display"
      Depth        32
      Modes        "1280x1024" "1024x768"
   EndSubSection
```

Die erste Auflösung nach dem "Modes" Stichwort ist die Auflösung, in der X startet. Mit dem Drücken von STRG-ALT-NUMMERN_BLOCK MINUS oder STRG-ALT-NUMMERN_BLOCK PLUS kannst du zwischen den

hier aufgeführten Auflösungen hin- und herschalten. Gemäß der Angaben oben wird X versuchen im 32-Bit-Modus zu starten (wegen der DefaultColorDepth Direktive, ohne sie würde X im 8-Bit-Modus starten). Die erste Auflösung, die versucht wird, ist 1280x1024 (es wird einfach der Reihenfolge in der 'Modes'-Zeile gefolgt). Denke daran, dass "1280x1024" einfach ein Label für die Werte in der ModeLine ist.

Du solltest wissen, dass das ModeLine Generator-Skript Optionen hat, um seine Timings für ältere oder kleinere Monitore etwas zu lockern, und dass es die Möglichkeit hat, ModeLines für spezielle Monitore anzubieten. Abhängig davon, was für eine Hardware du hast, ist sie vielleicht nur schwer mit den Standard-Optionen zu betreiben. Wenn das Bild zu groß, zu breit oder zu klein ist oder nicht genügend horizontal oder vertikal gekippt ist und die Monitor-Kontrollen zur Kompensierung nicht ausreichen, kann man mittels xvidtune(1) die ModeLine besser dem Monitor anpassen.

In den meisten modernen Monitoren gibt es kein festes Limit der Bandbreite, daher ist sie auch oftmals nicht in den Spezifikationen aufgeführt. Aber je mehr du in der Bandbreite nach oben gehst, desto verschwommener wird das Bild. Du könntest also zum Testen die Bandbreite deiner Grafikkarte (auch "dotclock" genannt) eingeben (so kannst du deinen Monitor nicht beschädigen) und Schritt-für-Schritt in Bandbreite heruntergehen, bis du ein schönes, klares Bild hast.

Wenn dir das unnötig kompliziert erscheint, liegt das daran, dass es genau das ist. XFree86 4.0 kümmert sich darum und macht diesen Prozess bedeutend einfacher, da es viele eingebaute Modi hat und außerdem in der Lage ist, Angaben aus vielen "plug and play" DDC und DDC2 Monitoren auszulesen.

Du kannst das "Colas XFree86 ModeLine Generator script" hier herunterladen: http://koala.ilog.fr/ftp/pub/Klone/. Du brauchst den Klone Interpreter und musst ihn kompilieren. Er ist als `lang/klone` in den Ports. Die Skripte existieren im "scripts" Verzeichnis der Klone Distribution. (Der Port installiert sie nach /usr/local/lib/klone/scripts.)

Es sind zwei Versionen des Skriptes dabei, die erste ist eine CGI Version, die identisch zu der obigen Webseite ist. Die zweite ist eine nicht-CGI Version, die deine komplette XF86Config-Datei nimmt, die Monitor-Spezifikationen dekodiert, die du in xf86config/XF86Setup eingegeben hast (Hast du eigentlich die echten Spezifikationen für deinen Monitor eingegeben, oder die generischen benutzt?) und passt die existierenden ModeLines an.

[FAQ Index] [Zum Kapitel 10 - Systemverwaltung] [Zum Kapitel 12 - Plattformspezifische Fragen]


www@openbsd.org

$OpenBSD: faq11.html,v 1.10 2005/02/12 20:36:53 jufi Exp $

OpenBSD

[FAQ Index] [Zum Kapitel 11 - Performance Tuning] [Zum Kapitel 14 - Platten Einrichtung]

12 - Plattform-spezifische Fragen

Inhaltsverzeichnis

12.1 - Generelle Hardware Anmerkungen

12.1.1 - PCI Devices

- PCI Geräte sind größtenteils selbst-konfigurierend -- der Computer und das Betriebssystem werden den Karten die benötigten Ressourcen zuweisen.
- Interrupts können auf dem PCI Bus geteilt werden. Nicht nur, dass sie geteilt werden können, das System wird sogar bessere Leistung haben, wenn die IRQs geteilt werden, insbesondere auf i386 Systemen.
- Es gibt einige verschiedene PCI Bus Standards. Du wirst ab und zu eine PCI2.2 Spezifikation Karte finden, die einfach nicht in einem PCI2.1 Spezifikation System laufen will. Außerdem, viele Karten mit On-Board Brücken (zum Beispiel Multi-Port Netzwerkkarten) werden in alten Systemen nicht vernünftig laufen.
- Der PCI Bus unterstützt zwei Spannungslevel, 3.3v und 5v. Karten, die mit 3.3v Spannung funktionieren, haben eine zweite Kerbe in ihrem PCI Connector. Die meisten PCI Karten verwenden 5v Spannung, die von den meisten Computern genutzt wird. Die Soekris Single-Board Computer (Net45x1 und Net4801) sind häufig-angetroffene Computer, die nur 3.3v Spannung unterstützen.

12.1.2 - ISA Devices

- ISA Geräte können keine Ressourcen teilen und müssen normalerweise manuell so eingerichtet werden, dass die Einstellungen keine Konflikte mit anderen Geräten im System verursachen.
- Einige ISA Geräte sind "Plug and Play" (isapnp(4)) -- wenn du dennoch irgendwelche Probleme mit diesen Geräten hast, stelle ihre Konfiguration in deiner dmesg(8) sicher, ISAPnP funktioniert nicht immer so wie gewollt.
- Wenn du eine Wahl hast, ist den meisten Leuten generell am besten damit geraten, ISA Karten im Vorzug für PCI Karten zu vermeiden. ISA Karten sind schwerer zu konfigurieren und haben einen größeren negativen Effekt auf die Leistung des Systems.

12.1.3 - Mein Device wird "erkannt", aber sagt "`not configured`" in dmesg

Kurz gesagt bedeutet das, dass dein Device nicht von deinem Kernel unterstützt wird, den du zur Zeit verwendest, so dass du nicht in der Lage sein wirst, es zu benutzen.

PCI und viele andere Gerätetypen bieten Identifizierungsinformationen an, so dass das OS Devices korrekt erkennen und unterstützen kann. Erkennung hinzuzufügen ist einfach, Unterstützung häufig nicht. Hier ist ein Teil einer dmesg mit zwei Beispielen von "not configured" Devices:

```
...
vendor "Intel", unknown product 0x5201 (class network subclass ethernet,
rev 0x03) at pci2 dev 9 function 1 not configured
...
"Intel EE Pro 100" rev 0x03 at pci2 dev 10 function 0 not configured
...
```

Das erste Device (eine Netzwerkkarte) hatte ihren Vendor Code identifiziert und der generelle Typ der Karte wurde ermittelt, aber nicht das genaue Modell der Karte. Das zweite Beispiel war eine andere Netzwerkkarte, diese hat ein Entwickler gesehen und hat sie in die Identifikationsdatei eingetragen, die genutzt wird, um die Karte zu identifizieren. In beiden Fällen jedoch werden die Karten nicht funktionieren, da beide als "`not configured`" angezeigt werden, was bedeutet, dass kein Treiber für diese Karten zuständig ist.

Was kann ich mit einem "`not configured`" Device anfangen?

- Wenn das Device oder die Karte, die du siehst, keine ist, die du benötigst, kannst du die "`not configured`" Devices einfach ignorieren, da sie deinem System nicht schaden werden. Einige "besondere Zwecke" Devices sind bewusst unkonfiguriert geblieben, so dass das BIOS des Systems sie handhaben kann.

- In einigen Fällen ist es einfach eine Variation eines bereits unterstützten Devices, in dessen Fall es relativ einfach für einen Entwickler ist, Unterstützung für diese neue Karte hinzuzufügen. In anderen Fällen könnte es ein gar nicht unterstützter Chipsatz oder eine nicht unterstützte Implentierung sein (wie in dem Beispiel oben). In diesem Fall muss ein neuer Treiber geschrieben werden, was vielleicht nicht einmal möglich ist, wenn das Device nicht vollständig dokumentiert wurde. Du kannst natürlich gerne selbst einen Treiber für das Device schreiben.
- Wenn du einen Installationskernel verwendest, könnte das Device von dem Installationsmedium aus, das du verwendest, nicht unterstützt werden, aber eventuell durchaus von einer anderen Bootdisk. Dies ist häufig bei Anwendern mit einigen populären SCSI Karten der Fall, die die Fußnoten der i386 Plattform Seite falsch lesen und alle Bootfloppies ausprobieren, auf denen ihre SCSI Karte nicht unterstützt wird, statt die eine zu verwenden, mit der sie laufen würde.
- Wenn du einen modifizierten Kernel verwendest, könntest du eventuell die Unterstützung für ein Device entfernt haben, das du nun brauchst. Devices von einem Kernel zu entfernen ist normalerweise eine schlechte Idee. Dies ist der Grund warum.
- Bevor du ein "`not configured`" Device meldest, stelle sicher, dass du zuerst den aktuellsten Snapshot ausprobiert hast, da die Unterstützung vielleicht bereits hinzugefügt worden ist und überprüfe die Mailinglisten Archive, um zu sehen, ob dieses Thema bereits besprochen wurde. Denke jedoch daran, dass, wenn du eine ältere Version von OpenBSD verwendest, du normalerweise ein Upgrade ausführen musst, um die Vorzüge eines neu geschriebenen Treibers erhalten zu können.

12.1.4 - Ich habe eine Karte, die als "unterstützt" aufgelistet ist, aber sie funktioniert nicht!

Unglücklicherweise verwenden viele Hersteller eine Produkt-Modellnummer, um auf eine Markposition zu deuten, statt auf die technische Natur eines Produkts. Aus diesem Grund kann es sein, dass du ein Produkt mit dem gleichen Namen oder der gleichen Modellnummer eines Produktes kaufst, die auf den Plattform Seiten aufgelistet wird, aber am Ende mit einem völlig anderen Produkt da stehst, das eventuell nicht mit OpenBSD funktioniert. Zum Beispiel basierten viele frühe Wireless Netzwerkkarten auf dem Prism2 Chipsatz, unter Verwendung des (wi(4)) Treibers, aber später, als die kostengünstigeren Chips verfügbar wurden, änderten viele Hersteller ihr Produkt, so dass diese Chips, für die keine Open Source Treiber existieren, verwendet wurden, aber änderten niemals die Modellnummern. Wireless Netzwerkkarten sind unglücklicherweise weit davon entfernt, das einzige Beispiel hierfür zu sein.

12.2 - DEC Alpha

[bisher nichts]

12.3 - AMD 64

[bisher nichts]

12.4 - CATS ARM Entwicklungsboard

[bisher nichts]

12.5 - HP300

[bisher nichts]

12.6 - HPPA

[bisher nichts]

12.7 - i386

12.7.1 - ISA NICs

Da OpenBSD auch auf älterer Hardware gut läuft, enden Benutzer häufig dabei, ISA NICs mit OpenBSD Systemen zu verwenden. ISA Hardware benötigt sehr viel mehr Konfiguration und Verständnis als es mit PCI Hardware der Fall ist. Du kannst normalerweise nicht einfach die Karte in den Computer stecken und erwarten, dass sie auf magische Weise funktioniert. In vielen Maschinen musst du die Ressourcen, die die Karte verwendet, im BIOS des Systems reservieren, wenn dein ISA Device nicht in einem "Plug 'n' Play" (PNP) Modus ist.

3Com 3C509B ep(4)

Dies ist eine exzellent funktionierende ISA NIC, die vom ep(4) Treiber unterstützt wird. Die ‚B' Version kann von der nicht-B Version anhand der Beschriftung auf der Karte und anhand des größeren "main" Chips auf der Karte (ungefähr 2,5cm auf einer Seite für die ‚B' Version gegenüber 2cm auf einer Seite für die ältere Version) unterschieden werden und wird bessere Leistung auf einem "loaded" oder "dual" Netzwerkkarten System liefern. Der 3C509B wird in einem PNP Modus konfiguriert ausgeliefert, der unglücklicherweise nicht mit Standards konform ist und Probleme in OpenBSDs isapnp(4) Unterstützung hervorruft. Die Karte wird zuerst als ein nicht-PNP Device aufgefasst und dann, wenn später die PNP Unterstützung wieder on-line kommt, in einer extra NIC resultieren, die in der dmesg angezeigt wird. Dies mag gut funktionieren oder es führt zu anderen Problemen. Es ist dringend empfohlen, dass der PNP Modus der 3C509B Karten ausgeschaltet und manuell auf nicht-konflikt-verursachende Einstellungen verändert wird, die unter Verwendung der 3Com DOS-basierenden Konfigurationsanwendungen vor der Konfiguration eingerichtet werden können.

Der ep(4) Treiber wird die Karten mit irgendeiner Hardware Kombination aufnehmen, die keinen Konflikt mit anderen Devices im System verursacht.

Wenn du mehrere 3C509 Karten in deinem System hast, wird empfohlen, dass du die Rückseite der Karte mit der MAC Adresse kennzeichnest und die dmesg verwendest, um zu ermitteln, welche welche ist.

Bedenke, dass die 3C509, die 3C905 und die 3C590 sehr oft verwechselt werden. Die 3C509 ist eine 10Mbps ISA Karte, die 3C905 und 3C590 sind PCI Karten.

NE2000

Die Original NE2000 NIC wurde in der Mitte der 1980er von Novell entwickelt. Seitdem haben viele Hersteller Karten produziert, die sehr ähnlich sind, die normalerweise NE2000-kompatible oder ‚clones' genannt wurden. Die Leistungen dieser ‚clone' Karten variieren stark. Während einige ältere NE2000-kompatible Karten sehr gut liefen, liefen viele der zur Zeit erhältlichen schlecht. NE2000-kompatible sind durch den ne(4) Treiber unter OpenBSD unterstützt.

OpenBSD wird einige ISAPNP-fähige NE2000-kompatible Karten gut handhaben, wenn der ISAPNP Modus angeschaltet ist. Andere Karten müssen entweder mit Jumpern oder DOS-basierenden Konfigurationsanwendungen eingerichtet werden. Unglücklicherweise, da die Original NE2000 Karten weder eine Softwarekonfiguration noch ISAPNP Unterstützung hatten, existieren keine Standards hierfür -- du benötigst die Anwendung, die ursprünglich mit der spezfischen Netzwerkkarte mit ausgeliefert worden ist. Diese kann oft schwer zu erhalten sein.

Der ne(4) Treiber unterstützt drei Konfigurationen von der ISA NE2000 Karte in dem GENERIC OpenBSD Kernel:

```
ne0:  port 0x240 irq 9
ne1:  port 0x300 irq 10
ne2:  port 0x280 irq 9
```

Wenn diese Einstellungen nicht hinnehmbar sind, kannst du sie mit Hilfe der User Kernel Configuration (UKC) oder anhand des Kompilieren eines angepassten Kernels einstellen.

Bedenke, dass die ne(4) Treiber recht "plump" sind -- nur der E/A Port wird untersucht und wenn einer der oben angegeben E/A Adressen entdeckt wird, wird der dazugehörige IRQ *angenommen*. dmesg(8) wird nicht den

tatsächlichen IRQ der Karte im Falle der ISA ne(4) Treiber wiedergeben. Wenn dies nicht der aktuelle IRQ ist, auf den deine Karte eingestellt wurde, wird es nicht funktionieren.

Bedenke, dass es nicht-ISA Karten gibt, die den ne(4) Treiber verwenden -- PCI und PCMCIA ne(4) Karten existieren. Diese Hinweise treffen nicht auf diese zu, diese Devices werden automatisch konfiguriert.

12.7.2 - OpenBSD will nicht auf meinem 80386/80386SX/80486SX System laufen

80386sx

Der 80386sx hat einen maximalen Adressbereich von 16M, der sehr stark am unteren Ende von OpenBSD/i386s Unterstützung liegt. Die meisten 80386sx Systeme können nicht mehr als 8M an RAM unterstützen, was sie in die "Nur für Experten" Kategorie platziert, da einige nicht-triviale Schritte und ein zweiter Computer benötigt sind, um sie zum Laufen zu kriegen. Siehe ebenfalls die nächste Sektion:

80386

OpenBSD wird auf einem 80386 oder 80386sx System laufen, WENN es über einen 80387 oder 80387sx Hardware Math-Coprozessor (Floating Point Unit, oder FPU) verfügt. Leider waren diese FPUs nicht sehr üblich, so dass viele 80386 Systeme diese nicht haben. OpenBSD wird nicht ohne FPU auf der i386 Plattform laufen. Noch einmal, sei dir bewusst, dass dies sehr wenig Prozessorkraft für ein krypto-intensives Betriebssystem wie OpenBSD ist. Du wirst vermutlich nicht glücklich mit der Leistung einer solchen Maschine für den täglichen Gebrauch sein.

80486SX

Der 80486SX Chip war eine "günstige" Version des 80486, dem die Hardware Floating Point Unterstützung (wie dem 80386) fehlte, die OpenBSD benötigt. Zum Glück sind volle 80486DX Chips recht häufig verfügbar und für die meisten Systeme ein einfaches Upgrade.

12.7.3 - Meine dmesg zeigt mehrere Devices, die sich den gleichen Interrupt (IRQ) teilen!

Dies ist völlig hinnehmbar und in der Tat für PCI Geräte sogar gewünscht. Dies ist ein Design Feature des PCI Busses. Einige Leute werden sagen, dass das Teilen der Interrupt Requests (IRQs) schlecht ist, wobei sie die Situation entweder mit dem ISA Bus (bei dem das Teilen der IRQs nicht erlaubt ist) oder es mit früherer Erfahrung mit kaputter Hardware oder Software verwechseln.

ISA Geräte können nicht IRQs teilen. Wenn du ISA Geräte findest, die IRQs teilen, musst du dieses Problem beheben.

12.7.4 - Wie verwende ich meine USB Tastatur?

OpenBSD/i386 wird üblicherweise eine USB Tastatur und die Maus ohne Schwierigkeiten nach der Installation verwenden. Trotzdem kann die Installation von OpenBSD auf einem System mit einer USB Tastatur schwierig sein, da der Installationskernel nicht alle zusätzlichen USB Treiber hat, die für die volle Unterstützung der USB Tastatur benötigt werden. Da Maschinen variieren, musst du eventuell experimentieren:

- Einige Systeme haben eine BIOS Option für "Legacy USB support" oder ähnlich genannt. Dies erlaubt dem BIOS eine herkömmliche PS/2 Tastatur zu emulieren. Wenn dies nicht aktiviert ist, wirst du *nicht* in der Lage sein, Kommandos am `boot>` Prompt einzugeben. Jedoch werden einige Systeme mit dieser Funktion besser laufen, andere, wenn diese Funktion ausgeschaltet ist. Einige werden die Installation mit einer Einstellung besser ausführen können und im normalen Betrieb die andere verwenden. Einige Maschinen haben diese Option nicht.
- Wenn du "Legacy USB support" verwendest, kann es sein, dass du feststellst, dass es besser läuft, wenn du die USB Unterstützung des Kernel in der User Kernel Configuration (UKC) durch Eingabe von "`disable uhci`" vor dem Abschließen des Bootens ausstellst.
- PS/2 Tastatur/Maus-zu-USB Adapter können eventuell nicht funktionieren.

- Du kannst *eventuell* feststellen, dass das System stabiler ist, wenn die Maus nicht angeschlossen ist, bis das System vollständig installiert ist.
- Im schlechtesten Fall kann es sein, dass du es einfacher finden wirst, OpenBSD auf einem anderen System zu installieren und dann die Festplatte in deine "legacy-freie" Maschine schiebst.

Wenn du dann OpenBSD installiert hast:

- Unter Verwendung von UKC pckbd0 auszuschalten kann in weniger "Kernel Nachrichten Lärm" (normalerweise `pckdc: cmd failed`) enden, wobei du keine PS/2 Maus mehr verwenden kannst, wenn du das tust.
- Die USB Tastatur und Maus werden genauso wie PS/2 Tastatur und Maus von X gehandhabt. Verwende "wscons" für deinen Maustreiber.
- Zur Zeit existiert kein Weg um eine serielle Konsole auf einem "legacy-freien" (keine serielle, parallele oder PS/2 Schnittstelle) System zu betreiben. Wenn du Schwierigkeiten mit deinem System hast, musst du die Nachrichten mit Stift und Papier aufschreiben.

Unglücklicherweise existieren eine Menge Wege, wie USB auf PCs zur Zeit unterstützt werden, so dass du mit deinem System experimentieren musst, um deine USB Tastatur ordentlich zum Laufen zu kriegen. Bitte kontaktiere faq@openbsd.org mit weiteren gut-dokumentierten USB Tastatur Tipps, die du finden kannst.

12.7.5 - Meine Tastatur/Maus hängt sich ständig auf (oder spielt verrückt)!

Dies wird meistens beobachtet, wenn man eine "Switch Box" verwendet, um mehrere Computer an eine Tastatur, einen Monitor und eine Maus anzuschließen. Du kannst mit unterschiedlichen Marken und Design Switch Boxen experimentieren, aber OpenBSD scheint sensibler auf das Wechseln der Maus zu reagieren als andere Betriebssysteme. Das Problem ist normalerweise nur das Wechseln der Maus. Wenn du keine Maus verwendest, ist die Lösung einfach: schließe kein Mauskabel an den Computer an. Wenn du eine Maus verwendest, wäre eine Lösung, "eine Maus pro Computer" zu verwenden und nur die Tastatur und den Monitor zu Wechseln. Wenn du nur Konsolenzugriff auf die Maschine haben möchtest, solltest du stattdessen eine serielle Konsole in Betracht ziehen.

12.7.6 - Werden WinModems unterstützt?

WinModems sind günstige Modems, die darauf angewiesen sind, dass der Prozessor einen Großteil der Signalverarbeitung durchführt, die normalerweise in der Hardware eines "echten" Modems gemacht werden. Aus der Tatsache heraus, dass es eine große Anzahl an inkompatiblen und typischerweise nicht dokumentierten WinModem Chips gibt, existiert keine Unterstützung für WinModems in OpenBSD und es ist unwahrscheinlich, dass sich das ändern wird.

12.8 - Mac68k

Hinweis: seit dem OpenBSD 3.6 Release wurden viele wichtige Verbesserungen an der OpenBSD/mac68k-Plattform durchgeführt. Wenn du irgendeines dieser folgenden Probleme hast (mit Außnahme des Problems mit dem Zeitverlust), verwende bitte einen aktuellen Snapshot statt -release

12.8.1 - Mein Mac68k System scheint nicht zu funktionieren

Unglücklicherweise haben wir eine kleine Anzahl an unterschiedlichen Mac68 Maschinen in den Händen der OpenBSD Entwickler und daher ist zur Zeit nur von ein paar wenigen bekannt, dass sie funktionieren.
(*in -current wird angenommen, dass alle verfügbaren 68040-basierenden (nicht 68LC040) mac68k-Systeme zumindest fast vollständig funktionieren*)

12.8.2 - Wieso verliert mein Mac68k soviel Zeit?

Dies wird durch einen Hardwarefehler verursacht. OpenBSD verwendet Clock Interrupts, um die aktuelle Zeit beizubehalten, aber diese Interrupts haben die niedrigste Priorität in der Mac68k Architektur. Hierdurch werden unter hoher Last (wie zum Beispiel Platten- oder Netzwerk-Aktivitäten) Clock Interrupts verloren gehen und die Unix Uhr wird nicht so weiterlaufen, wie sie sollte. MacPPC Systeme haben dieses Problem nicht.

Mac OS umgeht dieses Problem, indem es immer die Hardware Uhrzeit ausliest. OpenBSD liest die Hardware Uhrzeit nur zur Bootzeit aus und ignoriert sie danach. Du kannst während des Shutdowns eventuell feststellen, dass der Kernel nicht überzeugt genug davon ist, die Unix Zeit zurück in die Hardware Uhr zu schreiben, weil dieses Zeitproblem bekannt genug ist.

Eine einfache Lösung ist, rdate(8) auf einer regulären Basis auszuführen, indem du einen crontab Eintrag hierfür hast. Ein anderer guter Ort, rdate(8) auszuführen, ist in deiner */etc/ppp/ppp.linkup* Datei, wenn du permanent verbunden und ein PPP Benutzer bist. Eine andere Lösung wäre, ntpd(8) zu verwenden, so, wie es in der OpenNTPD-FAQ beschrieben steht. Man sollte jedoch beachten, dass keiner dieser Schritte dazu führen wird, dass ein Mac68k eine vernünftige Zeitsynchronisation vorweist, lediglich das Ausmaß der Fehler wird begrenzt. Fehler von zehn Minuten oder länger können unter hoher Last durchaus eintreten.

Siehe auch: http://www.macbsd.com/macbsd/macbsd-docs/faq/faq-3.html#ss3.17

12.8.3 - Mein Mac68k System will nicht mit zwei Platten funktionieren

Ja, leider ist dies ein bekanntes Problem. Dies äußert sich normalerweise in Abstürzen, kurz nachdem das System multi-user geht und der Absturz nicht offensichtlich mit dem Plattensystem zusammenhängt. Eine Lösung ist, nur ein Laufwerk mit deinem Mac68k System zu verwenden.
(*in -current behoben*)

12.8.4 - Der Installer stürzt während der Installation ab

Der Mac68k Installer, der unter Mac OS läuft, wird Dateien nicht in eine "große" Partition installieren. Wenn irgendeine Partition, auf die du installieren willst, größer als 500M ist, kannst du von folgendem seltsamen Fehler ausgehen:

```
Error on SCSIRead(), #5
pos = 0, i = 22, fs = /
alloccgblk: can't find blk in cyl
```

Die Lösung hierfür ist, erst ein Minimalsystem in eine "kleine" root Partition zu installieren, dann OpenBSD zu booten und Dinge nach eigenem Geschmack wieder neu zuzuweisen.

So, lass uns sagen, du möchtest eine 200M / Partition. Erstelle eine root Partition, erstelle die anderen Partitionen, die du möchtest und newfs sie alle mit Mac OS Anwendungen. Installiere etc36.tgz und base36.tgz in diese / Partition.

Mounte deine anderen Partitionen auf einen temporären Mount Point und kopiere die Verzeichnisse, die du auf ihnen haben willst, wie hier vorgeführt.

Nun modifiziere /etc/fstab, starte neu und entpacke den Rest der *.tgz Dateien wie hier beschrieben.

Diesem Prozess folgend kann jede Größe für die nicht-root Partitionen erreicht werden.
(*in -current behoben*)

12.9 - MacPPC

12.9.1 - Wieso ist mein bm(4) Treiber so langsam?

Die bm Treiber, die den BMAC Chip unterstützen, der auf einigen MacPPC Systemen verwendet wird (dazu zählen auch frühe iMacs), haben ein Problem, wenn sie bei 100Mbps laufen. Es wird dringend empfohlen, dass du die Treiber auf 10Mbps setzt, indem du eine `"media 10baseT"` Option in deiner `/etc/hostname.bm0` Datei verwendest oder sie auf andere Weise auf 10Mbps bei deinem Hub oder Switch zwingst.

12.10 - MVME68k

[bisher nichts]

12.11 - MVME88k

[bisher nichts]

12.12 - SPARC

[bisher nichts]

12.13 - UltraSPARC (sparc64)

12.13.1 - Meine UltraSPARC will nicht vom Floppy Image booten

Nur die Ultra 1/1e und Ultra 2 können *irgendein* OS von Floppy Diskette booten. Verwende stattdessen CD-ROM, Miniroot oder Netzwerk Boot, um deine Installation durchzuführen.

12.12 - DEC VAX

[bisher nichts]

[FAQ Index] [Zum Kapitel 11 - Performance Tuning] [Zum Kapitel 14 - Platten Einrichtung]


www@openbsd.org
$OpenBSD: faq12.html,v 1.22 2005/01/31 13:15:54 saad Exp $

OpenBSD

[FAQ Index] [Zum Kapitel 12 - Plattform-spezifische Fragen]

14 - Platten Einrichtung

Inhaltsverzeichnis

14.1 - Benutzung von OpenBSDs disklabel(8)

Inhaltsverzeichnis

Was ist disklabel(8)?

Lese zunächst die disklabel(8) Manual Seite.

Disklabels werden erzeugt, um ein effizientes Interface zwischen deiner Festplatte und den Festplattentreibern, die im Kernel enthalten sind, zu erzeugen. Labels enthalten bestimmte Informationen über deine Festplatte, wie z.B. die Plattengeometrie und Informationen über deine Dateisysteme. Dies wird dann vom ‚bootstrap' Programm benutzt, um die Festplatte zu laden und um zu wissen, wo auf der Platte die Dateisysteme sind. Labels werden auch zusammen mit den Dateisystemen benutzt, um eine effizientere Umgebung zu erzeugen. Tiefergehende Informationen über disklabel gibt es in der disklabel(5) Manual Seite.

Zusätzlich führt die Benutzung von disklabel zur Überwindung der Architekturgrenzen beim Partitionieren von Festplatten. Auf i386 kann man z.B. nur 4 primäre Partitionen haben. (Partitionen, so wie sie andere Betriebssysteme wie Windows NT oder DOS sehen.) Mit disklabel(8) benutzt du eine dieser ‚primären' Partitionen, die dann *alle* deine OpenBSD Partitionen enthält (z.B. ‚swap', ‚/', ‚/usr' und ‚/var'). Und du hast noch 3 weitere für andere Betriebssysteme über!

disklabel(8) während der OpenBSD Installation

Einer der Hauptteile der OpenBSD Installation ist das erstmalige Erzeugen der Labels. Das kommt (für i386 Benutzer) direkt nach der Benutzung von fdisk(8). Während der Installation benutzt du disklabel, um deine separaten Labels zu erzeugen, die deine separaten Mount Points enthalten. Während der Installation kannst du mittels disklabel(8) deine Mount Points setzen, aber das ist eigentlich nicht nötig, da du deine Änderungen später sowieso bestätigen musst. Aber es macht deine Installation schon etwas geradliniger.

Da das während der Installation geschieht, hast du noch keine funktionierenden Labels und sie müssen erst erzeugt werden. Das erste Label, das du erzeugst, ist das Label ‚a'. Das SOLLTE das Label sein, auf dem dann / gemountet wird. Die empfohlenen Partitionen und ihren Größen kannst du dir unter FAQ4, Wieviel Platz brauche ich für eine OpenBSD Installation? ansehen. Für Server wird empfohlen, zumindest diese Labels separat zu halten. Für Desktop User reicht vermutlich ein einzelner Mountpoint /. Wenn du deine root-Partition (‚a' Label) erzeugst, denk dran, dass du in jedem Fall noch ETWAS Platz für dein Swap Label benötigst. Jetzt kennst du die Grundlagen und daher geben wir hier jetzt mal ein Beispiel für das Benutzen von disklabel. In diesem ersten Beispiel wird angenommen, dass OpenBSD das einzige Betriebssystem auf diesem Computer ist und eine vollständige Installation gemacht wird.

```
Wenn die Festplatte mit anderen Betriebssystemen geteilt wird, sollten diese
Betriebssysteme einen BIOS Partitionseintrag haben, der den kompletten Platz
umfasst, den sie beanspruchen. Aus Sicherheitsgründen solltest du auch
sicherstellen, dass alle OpenBSD Dateisysteme innerhalb des Offsets und der
Größe sind, die in der ,A6' BIOS Partitionstabelle angegeben sind.
(Standardmäßig wird der disklabel Editor versuchen, das zu erzwingen). Wenn
du dir nicht sicher bist, wie man mehrere Partitionen sauber benutzt
(also wie man /, /usr, /tmp, /var, /usr/local und andere Dinge voneinander
trennt), belasse es jetzt erstmal bei einem root und einem Swap
Label.

# using MBR partition 3: type A6 off 63 (0x3f) size 4991553 (0x4c2a41)

Treating sectors 63-16386300 as the OpenBSD portion of the disk.
You can use the 'b' command to change this.

Initial label editor (enter '?' for help at any prompt)
> d a
> a a
offset: [63] <Enter>
size: [16386237] 64M
Rounding to nearest cylinder: 131040
FS type: [4.2BSD] <Enter>
mount point: [none] /
fragment size: [1024] <Enter>
block size: [8192] <Enter>
cpg: [16] <Enter>
> a b
offset: [131103] <Enter>
size: [16255197] 64M
Rounding to nearest cylinder: 131040
FS type: [swap] <Enter>
```

An diesem Punkt hast du eine 64MB root Partition erzeugt, die mit / gemountet wird und eine 64Meg Swap Partition. In diesem Fall startet der Offset bei Sektor 63. So willst du es haben. Wenn es bei der Größe angekommen ist, wird dir Disklabel die Größen in Sektoren angeben, du musst aber die Größen nicht ebenfalls im gleichen Format eingeben. Wie im Beispiel oben kannst du Größen zum Beispiel so eingeben: *64 Megabytes = 64M* und *2 Gigabytes = 2G*. Disklabel wird dann einfach auf den naheliegendsten Zylinder runden. Im obigen Beispiel kann man auch sehen, dass disklabel annimmt, dass Label ‚b' Swapbereich sein wird. Das ist eine korrekte Annahme, da im GENERIC Kernel Swap auf Label ‚b' festgelegt ist, und daher solltest du dieser Richtlinie ebenfalls folgen und ‚b' als Swapbereich benutzen.

Das nächste Beispiel wird dich durch die Erzeugung zweier weiterer Labels führen. Im Übrigen kann das keine komplette Installation sein, da die Größe dieser beiden Partitionen nicht ausreicht, um OpenBSD komplett zu installieren. Das Erzeugen dieser ganzen Partitionen dient nur zur Wiederholung und Vertiefung.

```
> a d
offset: [262143] <Enter>
size: [16124157] 64M
Rounding to nearest cylinder: 131040
FS type: [4.2BSD] <Enter>
mount point: [none] /tmp
fragment size: [1024] <Enter>
block size: [8192] <Enter>
cpg: [16] <Enter>
> a e
offset: [393183] <Enter>
size: [15993117] 64M
Rounding to nearest cylinder: 131040
FS type: [4.2BSD] <Enter>
mount point: [none] /var
fragment size: [1024] <Enter>
block size: [8192] <Enter>
cpg: [16] <Enter>
```

Im obigen Beispiel fallen dir vermutlich zwei Dinge auf. Zum einen, dass der Offset, der als nächstes an der Reihe ist, automatisch für dich errechnet wird. Wenn du eine solche Installation machst, musst du dich mit dem Ändern der Offsets nicht herumschlagen. Ein weiterer Unterschied, der dir vielleicht auffällt, ist, dass Label ‚c' übersprungen wurde. Das ist aber Absicht, und zwar deshalb, weil Label ‚c' die ganze Festplatte repräsentiert. Aus diesem Grund solltest du Label ‚c' vollkommen in Ruhe lassen.

Sind deine Label erst einmal alle erzeugt, ist alles, was noch nötig ist, die Label auf die Festplatte zu schreiben und einfach mit dem Installationsprozess fortzufahren. Um alles zu schreiben und disklabel zu beenden (und mit der Installation weiterzumachen), tippe folgendes:

```
> w
> q
```

Gebräuchliche Verwendungen von disklabel(8)

Wenn dein System erst einmal installiert ist, solltest du disklabel nicht mehr allzuoft benutzen müssen. Aber du kannst es gebrauchen, wenn du z.B. Festplatten hinzufügen willst, welche entfernen willst oder auch einfach umstrukturieren möchtest. Eines der ersten Dinge, die du dann machst, ist, dir den momentanen gültigen Disklabel anzusehen. Und das geht so:

```
# disklabel wd0 >----- Oder was du dir auch immer für eine Platte ansehen willst

# using MBR partition 3: type A6 off 64 (0x40) size 16777152 (0xffffc0)
# /dev/rwd0c:
type: ESDI
disk:
label: TOSHIBA MK2720FC
flags:
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
cylinders: 2633
total sectors: 2654064
rpm: 3600
interleave: 1
trackskew: 0
cylinderskew: 0
headswitch: 0           # milliseconds
track-to-track seek: 0  # milliseconds
drivedata: 0

16 partitions:
#        size   offset    fstype   [fsize bsize   cpg]
  a:  2071440    65583    4.2BSD     1024  8192    16   # (Cyl.   65*- 2120)
  b:    65520       63      swap                        # (Cyl.    0*- 65)
  c:  2654064        0    unused        0     0         # (Cyl.    0 - 2632)
  j:   512001  2137023    4.2BSD     1024  8192    16   # (Cyl. 2120*- 2627*)
```

Der obige Befehl zeigt dir einfach den existierenden Disklabel und stellt sicher, dass du nichts kaputt machst oder durcheinanderbringst. (Was wir alle von Zeit zu Zeit mal brauchen.) Um aber Veränderungen durchzuführen, musst du die -E Option wie folgt mit angeben:

```
# disklabel -E wd0
```

Das wird dich an einen Prompt bringen, und zwar den selben, den du schon während der OpenBSD Installation benutzt hast. Das wahrscheinlich wichtigste Kommando an diesem Prompt ist ‚?'. Das erzeugt nämlich eine Liste mit möglichen Optionen für Disklabel. Mit Hilfe von ‚M' kannst du dir sogar die gesamte disklabel(8) Manual Seite ansehen. Von diesem Prompt aus wirst du dein gesamtes Hinzufügen, Löschen und Ändern der Partitionen vornehmen. Zusätzliche Informationen gibt es in der disklabel(8) Manual Seite.

14.2 - Benutzung von OpenBSDs fdisk(8)

Um sicher zu sein, lese zuerst die fdisk(8) Manual Seite.

Fdisk ist ein Programm, das bei der Wartung deiner Partitionen helfen soll. Dieses Programm wird auch bei der Installation benutzt, um deine OpenBSD Partition einzurichten (diese Partition kann mehrere Labels enthalten, jedes mit Dateisystemen/Swap/etc.). Es kann den Platz auf deiner Festplatte aufteilen und eine Partition als aktiv markieren. Dieses Programm wird für gewöhnlich im ‚single user' Modus benutzt (boot -s). Fdisk setzt auch den MBR auf deinen verschiedenen Festplatten.

Für Installationszwecke braucht man meistens nur **EINE** OpenBSD Partition und benutzt dann disklabel, um Swap und Dateisysteme darauf zu installieren.

Um dir nur deine Partitionstabelle mit fdisk anzugucken, verwende:

```
# fdisk fd0
```

Was dann eine ähnliche Ausgabe wie diese hier erzeugt:

```
Disk: sd0       geometry: 553/255/63 [8883945 Sectors]
Offset: 0       Signature: 0xAA55
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
*0: A6    3   0  1 -  552 254 63 [       48195:     8835750 ] OpenBSD
 1: 12    0   1  1 -    2 254 63 [          63:       48132 ] Compaq Diag.
 2: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
```

```
 3: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
```

In diesem Beispiel betrachten wir die Ausgabe des ersten SCSI Laufwerks. Wir können die OpenBSD Partition (A6) und ihre Größe sehen. Der * sagt uns, dass die OpenBSD Partition eine bootfähige Partition ist.

Im vorherigen Beispiel haben wir uns die Informationen nur angesehen. Was aber, wenn wir unsere Partitionstabelle verändern wollen? Nun, dazu müssen wir zunächst das **-e** Flag benutzen. Das bringt uns dann zu einer Kommandozeile, die uns mit fdisk interagieren lässt.

```
# fdisk -e wd0
Enter 'help' for information
fdisk: 1> help
        help            Command help list
        manual          Show entire OpenBSD man page for fdisk
        reinit          Re-initialize loaded MBR (to defaults)
        setpid          Set the identifier of a given table entry
        disk            Edit current drive stats
        edit            Edit given table entry
        flag            Flag given table entry as bootable
        update          Update machine code in loaded MBR
        select          Select extended partition table entry MBR
        print           Print loaded MBR partition table
        write           Write loaded MBR to disk
        exit            Exit edit of current MBR, without saving changes
        quit            Quit edit of current MBR, saving current changes
        abort           Abort program without saving current changes
fdisk: 1>
```

Es ist absolut sicher, in fdisk ein wenig rumzuwandern und zu probieren, solange man **N** auf die Frage antwortet, ob die Änderungen abgespeichert werden sollen und *NICHT* das **write** Kommando benutzt.

Hier ist eine Übersicht über die Kommandos, die man nach der Eingabe des **-e** Flags benutzen kann.

- **help** Zeige eine Liste der Kommandos an, die fdisk im interaktiven ‚edit' Modus versteht.
- **reinit** Initialisiere die momentane im Speicher befindliche Kopie des Bootblocks.
- **disk** Zeige die momentane Plattengeometrie an, die fdisk herausgefunden hat. Du bekommst eine Möglichkeit sie zu ändern, wenn du willst.
- **setpid** Ändere eine Partitionsidentifizierung des angegebenen Partitionstabellen Eintrages. Dieses Kommando ist insbesondere nützlich, um eine existierende Partition OpenBSD wieder zugänglich zu machen.
- **edit** Ändere eine ausgewählte Plattengeometrie in der Kopie des momentanen Bootblocks. Das geschieht entweder im BIOS ‚geometry' Modus oder in Sektor-Offsets und Größen.
- **flag** Den jetzigen Partitionstabelleneintrag bootfähig machen. Nur ein Eintrag zur Zeit kann bootbar sein. Wenn du von einer erweiterten Partition booten willst, musst du auch den entsprechenden Eintrag als bootfähig markieren.
- **update** Bringe den Maschinencode in der Speicherkopie des momentanen Bootblocks auf aktuellen Stand.
- **select** Wähle und lade den Bootblock, auf den der Eintrag der erweiterten Partitionstabelle im momentanen Bootblock zeigt.
- **print** Gebe die momentan im RAM befindlichen und gewählte Kopie des Bootblocks und seinen MBR auf dem Bildschirm aus.
- **write** Schreibe die RAM-Version des Bootblocks auf die Platte. Du wirst um eine Bestätigung gebeten.
- **exit** Verlasse den momentanen Level von fdisk, kehre entweder zur vorher gewählten Kopie eines Bootblocks im RAM zurück oder verlasse das Programm, wenn es keinen gibt.
- **quit** Verlasse den momentanen Level von fdisk, kehre entweder zur vorher gewählten Kopie eines Bootblocks im RAM zurück oder verlasse das Programm, wenn es keinen gibt. Anders als exit schreibt diese Variante den modifizierten Block auf die Platte.
- **abort** Verlasse das Programm ohne Änderungen zu speichern.

14.3 - Hinzufügen von weiteren Festplatten unter OpenBSD

Nun, nachdem du deine Festplatte **SAUBER** eingebaut hast, musst du fdisk(8) (*nur i386*) und auch disklabel(8) verwenden, um deine Festplatte unter OpenBSD benutzen zu können.

Die i386 Leute starten mit fdisk. Andere Architekturen können das einfach ignorieren. In dem Beispiel weiter unten werden wir dem System ein drittes SCSI Laufwerk hinzufügen.

```
# fdisk -i sd2
```

Das wird die "echte" Partitionstabelle der Festplatte für eine ausschließliche Benutzung von OpenBSD initialisieren. Als nächstes musst du ein Disklabel dafür erzeugen. Das wird wohl etwas verwirrend wirken.

```
# disklabel -e sd2

(der Bildschirm wird leer, dein $EDITOR erscheint)
type: SCSI
...bla...
```

```
sectors/track: 63
total sectors: 6185088
...bla...
16 partitions:
#        size   offset    fstype   [fsize bsize   cpg]
  c:  6185088        0    unused        0     0         # (Cyl.    0 - 6135)
  d:  1405080       63    4.2BSD     1024  8192    16   # (Cyl.    0*- 1393*)
  e:  4779945  1405143    4.2BSD     1024  8192    16   # (Cyl. 1393*- 6135)
```

Zunächst einmal ignoriere die ‚c' Partition, sie ist immer da und Programme wie disklabel benötigen sie, um zu funktionieren! ‚fstype' für OpenBSD ist 4.2BSD. ‚total sectors' ist die gesamte Größe der Festplatte. Nehmen wir an, es handelt sich um eine 3 Gigabyte Festplatte. Drei Gigabytes in der Sprache der Festplattenhersteller sind 3000 Megabytes. Dividiere also 6185088/3000 (benutze bc(1)). Du erhältst 2061. Um jetzt Partitionsgrößen für a, d, e, f, g, ... zu erstellen, rechne einfach X*2061, um X Megabytes Platz auf dieser Partition zu erhalten. Der Offset für deine erste Partition sollte derselbe sein, wie unter "sectors/track" vorher in disklabels Ausgabe angegeben. Bei uns ist es 63. Der Offset für jede Partition ist hinterher eine Kombination aus der Größe und dem Offset jeder anderen Partition (mit Ausnahme der C Partition, da sie keine Rolle in dieser Gleichung spielt).

Wenn du aber nur eine Partition auf deiner Festplatte brauchst, zum Beispiel, wenn du das ganze Ding nur zum Ablegen von Webseiten oder einem Heimatverzeichnis oder etwas anderem nutzen willst, nimm einfach die gesamte Größe der Platte und ziehe die Sektoren pro Spur davon ab. 6185088-63 = 6185025. Deine Partition ist

```
    d:  6185025       63    4.2BSD     1024  8192    16
```

Wenn dir das alles unnötig komplex erscheint, kannst du disklabel -E benutzten, um den selben Patitionierungsmodus zu erhalten, den du auf deiner Installationsdisk hattest! Dort kannst du "96M" benutzen, um "96 Megabytes" anzugeben. (Oder, wenn deine Festplatte groß genug ist, 96G für 96 Gigabytes!) Unglücklicherweise benutzt der -E Modus die BIOS Plattengeometrie und nicht die reale, und oft sind die beiden nicht deckungsgleich. Um dieses Problem zu umgehen, tippe ‚g d' für ‚geometry disk'. (Andere Möglichkeiten sind ‚g b' für ‚Geometry BIOS' und ‚g u' für ‚geometry user' oder einfach das, was das Label gesagt hat, bevor disklabel irgendwelche Änderungen gemacht hat.)

Das war eine Menge. Aber du bist noch nicht fertig. Zuletzt musst du noch das Dateisystem auf der Festplatte mittels newfs(8) erzeugen.

```
# newfs sd2a
```

Oder wie auch immer deine Festplatte nach dem OpenBSD Plattennummerierungs-Schema heißen mag. (Siehe einfach in der Ausgabe von dmesg(8) nach, um zu sehen, wie die Platte von OpenBSD benannt wurde.)

Nun überleg dir, wohin du deine gerade neu geschaffene Partition mounten willst. Sagen wir einfach mal /u. Daher erzeuge zunächst einmal /u. Dann mounte sie.

```
# mount /dev/sd2a /u
```

Zuletzt musst du sie noch zu /etc/fstab(5) hinzufügen.

```
/dev/sd2a /u ffs rw 1 1
```

Was aber, wenn du ein existierendes Verzeichnis, wie zum Beispiel /usr/local auslagern willst? Mounte die neue Platte unter /mnt und benutze `cpio -pdum`, um /usr/local in das /mnt Verzeichnis zu kopieren. Passe die /etc/fstab(5) Datei so an, dass die /usr/local Partition nun /dev/wd1a ist (deine frisch formatierte Partition). Beispiel:

```
/dev/sd2a /usr/local ffs rw 1 1
```

Reboote in den ‚single user' Modus mit **boot -s**, verschiebe das existierende /usr/local nach /usr/local-backup (oder lösche es gleich, wenn du mutig bist) und erzeuge ein leeres Verzeichnis /usr/local. Dann starte das System neu und voila, die Dateien sind da!

14.4 - Wie man in eine Datei swappt

(Hinweis: wenn du in eine Datei swappen willst, weil du immer "virtual memory exhausted" Fehler bekommst, solltest du lieber versuchen, deine ‚per-process limits' zu erhöhen und zwar mit cshs unlimit(1) oder auch mit shs ulimit(1).)

In eine Datei zu swappen benötigt keinen angepassten Kernel, obwohl das weiterhin gemacht werden könnte, zeigt dir diese FAQ, wie man den Swapbereich auf beide Arten hinzufügen kann.

In eine Datei swappen.

In eine Datei zu swappen ist der einfachste und schnellste Weg, um zusätzlichen Swap zu bekommen. Die Datei darf aber nicht auf einem Dateisystem mit SoftUpdates liegen (was ja standardmäßig deaktiviert ist). Für den Anfang findest du erstmal heraus, wieviel Swap du momentan hast und wieviel du davon benutzt, und das geht einfach mit dem swapctl(8) Werkzeug. Zum Beispiel mit diesem Kommando:

```
$ swapctl -l
Device      512-blocks     Used    Avail Capacity  Priority
swap_device      65520        8    65512     0%    0
```

Das zeigt alle Geräte, die momentan für das swappen benutzt werden, und ihre momentane Statistik an. Im obigen Beispiel gibt es nur ein

Gerät namens "swap_device". Das ist der vordefinierte Bereich auf der Platte, der für das Swappen benutzt wird. (Wird im Übrigen als Partition ‚b' bei Disklabels angezeigt) Wie du auch sehen kannst, wird das Gerät zur Zeit nicht sonderlich belastet oder vielmehr benutzt. Aber für den Zweck dieses Dokumentes tun wir einfach so, als wenn noch weitere 32MB benötigt werden würden.

Der erste Schritt, um eine Datei als Swapbereich zu nutzen, ist, die Datei zu erzeugen. Am besten macht man das mit Hilfe von dd(1). Hier ist ein Beispiel, das die 32M große Datei */var/swap* erzeugt.

```
$ sudo dd if=/dev/zero of=/var/swap bs=1k count=32768
32768+0 records in
32768+0 records out
33554432 bytes transferred in 20 secs (1677721 bytes/sec)
```

Nachdem das erledigt ist, können wir jetzt das Swappen auf dieses Device richten. Benutze einfach das folgende Kommando, um das Swappen auf dieses Device zu lenken

```
$ sudo chmod 600 /var/swap
$ sudo swapctl -a /var/swap
```

Jetzt müssen wir noch prüfen, ob sie auch korrekt zu unserer Liste der Swap-Devices hinzugefügt wurde.

```
$ swapctl -l
Device      512-blocks     Used    Avail Capacity  Priority
swap_device      65520        8    65512     0%    0
/var/swap        65536        0    65536     0%    0
Total           131056        8   131048     0%
```

Jetzt, da die Datei erzeugt wurde und in sie hinein geswappt wird, musst du noch eine Zeile in deine */etc/fstab* Datei hineinschreiben, so dass die Datei beim nächsten Booten auch benutzt wird. Wenn diese Zeile nicht hinzugefügt wird, wird dieses Swap-Device eben nicht konfiguriert.

```
$ cat /etc/fstab
/dev/wd0a / ffs rw 1 1
/var/swap /var/swap swap sw 0 0
```

Swappen über ein vnode Device

Dies ist eine dauerhaftere Lösung, um mehr Swapbereich zu erhalten. Um in eine Datei zu swappen, erzeuge zunächst einen Kernel mit vnd0c als Swap. Wenn du wd0a als root Dateisystem hast, und wd0b als bisherigen Swap, benutze diese Zeile in deiner Kernel Konfigurationsdatei (wenn du dir nicht sicher bist, siehe dir das Kapitel "Einen neuen Kernel kompilieren" in dieser FAQ an):

```
config          bsd     root on wd0a swap on wd0b and vnd0c dumps on wd0b
```

Nachdem das erledigt ist, muss die Datei erzeugt werden, in die geswappt werden soll. Du solltest dies mit dem selben Kommando wie in den vorherigen Beispielen machen.

```
$ sudo dd if=/dev/zero of=/var/swap bs=1k count=32768
32768+0 records in
32768+0 records out
33554432 bytes transferred in 20 secs (1677721 bytes/sec)
```

Da deine Datei jetzt an ihrem Platz ist, musst du die Datei in deine */etc/fstab* eintragen. Hier ist eine Beispielzeile, mit der man dieses Device beim Booten als Swap benutzt.

```
$ cat /etc/fstab
/dev/wd0a / ffs rw 1 1
/dev/vnd0c none swap sw 0 0
```

An diesem Punkt muss dein Computer neu gebootet werden, so dass die Änderungen am Kernel Effekt haben. Nachdem das passiert ist, ist es an der Zeit, das Gerät als Swap zu konfigurieren. Dazu wirst du vnconfig(8) benutzen.

```
$ sudo vnconfig -c -v vnd0 /var/swap
vnd0: 33554432 bytes on /var/swap
```

Als letzten Schritt musst du den Swap auf diesem Gerät noch einschalten. Wir machen das genau wie in dem Beispiel oben mit swapctl(8). Und zuletzt prüfen wir wieder, ob es auch korrekt in unsere Liste der Swap Devices eingetragen wurde.

```
$ sudo swapctl -a /dev/vnd0c
$ swapctl -l
Device      512-blocks     Used    Avail Capacity  Priority
swap_device      65520        8    65512     0%    0
/dev/vnd0c       65536        0    65536     0%    0
Total           131056        8   131048     0%
```

14.5 - Soft Updates

Soft Updates basieren auf einer Idee, die von Greg Ganger und Yale Patt vorgeschlagen wurde, und wurden für FreeBSD von Kirk McKusick entwickelt. SoftUpdates erzwingen eine gewisse Reihenfolge der Buffer Cache Operationen, was die Anforderungen für das Entfernen des FFS Codes ermöglicht, der für das synchrone Schreiben von Verzeichniseinträgen zuständig ist. Daher konnte ein großer Geschwindigkeitsanwachs in der Leistung der Schreibzugriffe auf Platten festgestellt werden.

Das Potential eines Hintergrund fsck(8) unter Verwendung von Soft Updates ist in OpenBSD noch nicht realisiert, so dass fsck(8) weiterhin nach einem unsauberen Shutdown benötigt wird. Dies kann sich in zukünftigen Versionen ändern.

Um Softupdates aktivieren zu können, muss in deinen Kernel folgende Option

option FFS_SOFTUPDATES

einkompiliert sein, dies ist bereits in GENERIC eingetragen.

Die Aktivierung von Soft Updates muss mit einer mount-zeit Option ausgeführt werden. Wenn eine Partition mit mount(8) gemountet wird, kannst du angeben, dass du Soft Updates auf dieser Partition aktivieren möchtest. Unten ist ein Beispiel */etc/fstab(5)* Eintrag, der eine Partition *sd0a* hat, die wir mit Soft Updates gemountet haben möchten.

```
/dev/sd0a / ffs rw,softdep 1 1
```

Hinweis für Sparc Anwender: Aktiviere Soft Updates nicht auf sun4 oder sun4c Maschinen. Diese Architekturen unterstützen nur eine sehr begrenzte Menge an Kernelspeicher und können diese Funktion nicht verwenden. Trotzdem sind sun4m Maschinen in Ordnung.

14.6 - Wie bootet OpenBSD/i386?

Der Bootprozess für OpenBSD/i386 ist nicht einfach und verstehen, wie es funktioniert, kann brauchbar sein, um ein Problem zu lösen, wenn Dinge nicht laufen. Es existieren vier Schlüsselmomente im Bootprozess:

1. ***Master Boot Record (MBR):*** Der Master Boot Record ist der erste physikalische Sektor (512 Bytes) auf der Platte. Er beinhaltet die primäre Partitionstabelle und ein kleines Programm, um den Partition Boot Record (PBR) zu laden. Bedenke, dass in einigen Umgebungen der Begriff "MBR" verwendet wird, um nur auf den Code Teil dieses ersten Blockes auf der Platte zu verweisen, statt auf den gesamten ersten Block (einschließlich der Partitionstabelle). Es ist äußerst wichtig, die Bedeutung von "initialize the MBR" zu verstehen -- in der Terminologie von OpenBSD würde es den gesamten MBR Sektor neu schreiben, nicht nur den Code, so wie es auf anderen Systemen der Fall sein könnte. Du wirst das nur selten machen wollen. Verwende stattdessen fdisk(8)s "-u" Kommandozeilen Option ("`fdisk -u wd0`").

   Während OpenBSD einen MBR beinhaltet, wirst du nicht gezwungen, ihn zu verwenden, da so gut wie jeder MBR OpenBSD booten kann. Der MBR wird von dem fdisk(8) Programm verändert, welches verwendet wird um die Partitionstabelle zu editieren und ebenfalls um den MBR Code auf die Platte zu installieren.

   OpenBSDs MBR kündigt sich selbst mit der Meldung an:

   ```
   Using drive 0, partition 3.
   ```

   die die Platte und Partition anzeigt, von der er den PBR laden wird. Zusätzlich zu dem Offensichtlichen, zeigt sie ebenfalls einen angehängten Punkt ("."), welcher darauf deutet, dass diese Maschine in der Lage ist, LBA Übersetzung zum Booten zu verwenden. Wenn die Maschine nicht in der Lage ist, LBA Übersetzung zu verwenden, wäre der obige Punkt mit einem Semikolon (";") ausgewechselt worden, das auf CHS Übersetzung deutet:

   ```
   Using Drive 0, Partition 3;
   ```

   Bedenke, dass der angehängte Punkt oder das angehängte Semikolon als ein Indikator für den "neuen" OpenBSD MBR angesehen werden kann, der mit OpenBSD 3.5 eingeführt wurde.
2. ***Partition Boot Record (PBR):*** Der Partition Boot Record, auch der PBR oder biosboot(8) (benannt nach dem Namen der Datei, der den Code beinhaltet) genannt wird, ist der erste physikalische Sektor der OpenBSD Partition auf der Platte. Der PBR ist der "first-stage Bootloader" für OpenBSD. Er wird vom MBR Code geladen und hat die Aufgabe, den OpenBSD ‚second-stage' Bootloader boot(8) zu laden. Wie der MBR ist auch der PBR eine sehr kleine Sektion von Code und Daten, insgesamt nur 512 Bytes. Das ist genug, um eine vollständig Dateisystem-bewusste Applikation zu laden, die BIOS-verfügbare Stelle von `/boot` wird physikalisch in den PBR während der Installation eingetragen.

   Der PBR wird von installboot installiert, das später in diesem Dokument genauer beschrieben wird. Der PBR kündigt sich selbst mit der Meldung an:

   ```
   Loading...
   ```

   die einen Punkt für jeden Dateisystemblock anzeigt, den er versucht auszulesen. Ebenfalls zeigt der PBR an, ob er LBA oder CHS zum Laden verwendet, wenn er CHS Übersetzung verwendet, zeigt er eine Nachricht mit einem Semikolon an:

   ```
   Loading;...
   ```

   Das ältere (vor v3.5) biosboot(8) zeigte die Nachricht "`reading boot...`" an.
3. ***Second Stage Bootloader, `/boot`:*** `/boot` wird vom PBR geladen und hat die Aufgabe, auf das OpenBSD Dateisystem durch das

BIOS der Maschine zuzugreifen und den aktuellen Kernel ausfindig zu machen und zu laden. boot(8) übergibt ebenfalls verschiedene Optionen und Informationen an den Kernel.

boot(8) ist ein interaktives Programm. Nachdem es geladen ist, versucht es, `/etc/boot.conf` ausfindig zu machen und zu laden, wenn sie existiert (was auf einer standardmäßigen Installation nicht der Fall sein muss) und verarbeitet sämtliche Kommandos in ihr. Wenn es durch `/etc/boot.conf` nicht anders angeordnet wurde, gibt es dem Benutzer einen Prompt aus:

```
probing: pc0 com0 com1 apm mem[636k 190M a20=on]
disk: fd0 hd0+
>> OpenBSD/i386 BOOT 2.06
boot>
```

Es gibt dem Benutzer (standardmäßig) fünf Sekunden lang die Möglichkeit, andere Aufgaben auszuführen, aber wenn keine vor dem Ablauf der Zeit eingegeben wurde, startet es sein normales Verhalten: den Kernel, `bsd`, von der root Partition der ersten Festplatte laden. Der second-stage Bootloader untersucht (examines) deine Systemhardware durch das BIOS (da der OpenBSD Kernel noch nicht geladen ist). Oben kannst du ein paar Dinge sehen, die er gesucht und gefunden hat:

- **pc0** - Die Standard-Tastatur und Bildschirmausgabe eines i386 Systems.
- **com0, com1** - Zwei serielle Schnittstellen
- **apm** - Advanced Power Management BIOS Funktionen
- **636k 190M** - Die Menge vom herkömmlichen (unterhalb von 1M) und erweiterten (überhalb von 1M) Speicher, den er gefunden hat
- **fd0 hd0+** - Die BIOS Laufwerke, die er gefunden hat, in diesem Fall ein Floppy und ein Festplatten Laufwerk.

Das ‚+' Zeichen nach "hd0" zeigt an, dass das BIOS `/boot` mitgeteilt hat, dass diese Festplatte über LBA angesprochen werden kann. Wenn eine erstmalige Installation ausgeführt wird, siehst du ab und zu einen ‚*' nach einer Festplatte -- dies deutet auf eine Platte hin, die so scheint, als wenn sie kein OpenBSD Disklabel beinhaltet.

4. ***Kernel: `/bsd`*:** Dies ist das Ziel des Bootprozesses, den OpenBSD Kernel in den RAM zu laden und sauber auszuführen. Wenn der Kernel einmal geladen wurde, kann OpenBSD direkt auf die Hardware zugreifen, nicht mehr durch das BIOS.

So, der Anfang vom Start des Bootprozesses könnte wie folgt aussehen:

```
Using drive 0, partition 3.                       <- MBR
Loading....                                       <- PBR
probing: pc0 com0 com1 apm mem[636k 190M a20=on] <- /boot
disk: fd0 hd0+
>> OpenBSD/i386 BOOT 2.06
boot>
booting hd0a:/bsd 4464500+838332 [58+204240+181750]=0x56cfd0
entry point at 0x100120

[ using 386464 bytes of bsd ELF symbol table ]
Copyright (c) 1982, 1986, 1989, 1991, 1993       <- Kernel
        The Regents of the University of California.  All rights reserved.
Copyright (c) 1995-2003 OpenBSD.  All rights reserved.  http://www.OpenBSD.org

OpenBSD 3.6 (GENERIC) #59: Fri Sep 17 12:32:57 MDT 2004
   ...
```

Was kann fehlschlagen

- **Schlechter/ungültiger/inkompatibler MBR:** Normalerweise hat eine gebrauchte Festplatte irgendeinen MBR Code installiert, aber wenn die Platte neu ist oder von einer anderen Plattform übernommen wurde UND du nicht mit "Yes" auf die "Use entire disk" Frage vom Installationsprozess geantwortet hast, kann es sein, dass du mit einer Platte ohne gültigem MBR da stehst und daher nicht in der Lage sein wirst zu Booten, obwohl sie eine gültige Partitionstabelle hat.

  Du kannst den OpenBSD MBR auf deine Festplatte unter Verwendung vom fdisk Programm installieren. Boote dein Installationsmedium, wähle "Shell" aus, um auf den Kommando-Prompt zu gelangen:

  ```
  # fdisk -u wd0
  ```

  Du kannst auch einen spezifischen MBR auf deine Platte mit fdisk installieren:

  ```
  # fdisk -u -f /usr/mdec/mbr wd0
  ```

  das die Datei `/usr/mdec/mbr` als deinen System-MBR installieren wird. Diese bestimmte Datei einer standardmäßigen OpenBSD Installation ist ebenfalls der standardmäßige MBR, der in fdisk integriert wurde, aber jeder andere MBR könnte hier angegeben werden.
- **Ungültige `/boot` Ortsangabe im PBR installiert:** Wenn installboot(8) den Partition Boot Record installiert, schreibt er die Blocknummer und den Offset von `/boots` Inode in den PBR. Daher wird das Löschen oder Ersetzen von `/boot`, ohne installboot(8) erneut auszuführen, dein System in eine Situation versetzen, in der es nicht mehr booten kann, da der PBR laden wird, auf was auch immer die Inode zeigt, die angegeben wurde, was vermutlich nicht mehr der erhoffte second-stage Bootloader sein wird! Seit `/boot` unter Verwendung von BIOS Aufrufen ausgelesen wird, waren ältere Versionen vom PBR sehr sensibel auf BIOS Plattenübersetzungen. Wenn du die Plattengeometrie (z.B., wenn du die Platte aus einem Computer genommen hast, der CHS Übersetzung verwendet hat, und es in einen steckst, der LBA Übersetzung verwendet oder sogar die Übersetzungsoption im BIOS

geändert hast) geändert hast, wird es *für das BIOS so wirken*, als wenn sie an einem anderen Ort liegen würde (es muss auf einen anderen nummerischen Block zugegriffen werden, um die gleichen Daten von der Platte zu erhalten), so dass du installboot(8) erneut ausführen musst, bevor das System neugestartet werden kann. Der neue (von OpenBSD 3.5 und später) PBR ist sehr viel tolleranter im Bezug auf Übersetzungsänderungen.

Da der PBR sehr klein ist, ist die Anzahl an Fehlermeldungen sehr begrenzt und recht kryptisch. Typische Nachrichten sind:

- **ERR R** -- Das BIOS gab einen Fehler zurück, als es versucht hat, einen Block von der Platte zu lesen. Es bedeutet meistens genau das, was es aussagt: von deiner Platte konnte nicht gelesen werden.
- **ERR M** -- Eine ungültige magic(5) Nummer wurde aus dem second-stage Bootloader Header gelesen. Dies bedeutet normalerweise, dass, was auch immer eingelesen wurde, NICHT `/boot` war, was darauf hinweist, dass installboot(8) nicht korrekt ausgeführt wurde, die /boot Datei geändert wurde oder du die Fähigkeit deines BIOS erschöpft hast, um von einer großen Platte zu lesen.

Andere Fehlermeldungen werden in der biosboot(8) Manual Seite ausführlich besprochen. Für weitere Informationen über den i386 Bootprozess, siehe

- boot_i386(8)
- http://www.ata-atapi.com/hiw.htm Hale Landis' "How it Works" documents.

14.7 - Welche Probleme treten bei großen Festplatten mit OpenBSD auf?

OpenBSD unterstützt ein individuelles Dateisystem von bis zu 2^{31}-1, oder 2,147,483,647, Sektoren und da jeder Sektor 512 Bytes groß ist, ist das ein kleiner Betrag unterhalb von 1T.

Natürlich sind die Fähigkeit eines Dateisystems und die Fähigkeit einer bestimmten Hardware zwei unterschiedliche Dinge. Eine neue 250G IDE Festplatte wird nicht mit älteren (vor >137G Standard) Interfaces funktionieren und einige sehr alte SCSI Adapter sind bekannt dafür, dass sie Probleme mit moderneren Laufwerken haben und einige alte BIOSe werden hängen, wenn sie einer modern-bestückten Festplatte begegnen. Du musst die Fähigkeiten deiner Hardware natürlich respektieren.

Partitionsgröße und Lokalitätsbegrenzungen

Leider ist die volle Funktionalität des OS nicht verfügbar, bis NACHDEM das OS in den Speicher geladen wurde. Der Bootprozess verwendet (und ist daher auch darauf beschränkt) die Boot ROM des Systems.

Aus diesem Grund muss die /bsd Datei (der Kernel) innerhalb des vom boot ROM adressierbarem Bereich liegen. Das bedeutet für einige ältere i386 Systeme, dass die root Partition vollständig innerhalb der ersten 504M liegen muss, aber neuere Computer können diese Grenze bei 2G, 8G, 32G, 128G oder mehr haben. Es ist ebenfalls sinnvoll zu erwähnen, dass viele relativ neue Computer, die Laufwerke mit mehr als 128G Speicher unterstützen tatsächlich eine BIOS Begrenzung für die ersten 128G für das Booten haben. Du kannst diese Systeme mit großen Laufwerken betreiben, aber deine root Partition muss innerhalb der ersten 128G liegen.

Bedenke, dass es möglich ist, ein 40G Laufwerk in einen alten 486er einzubauen und auf diesem OpenBSD mit einer großen Partition zu installieren und zu denken, dass du erfolgreich die vorherige Regel gebrochen hast. Trotzdem kann es dich auf einem höchst unangenehmen Weg verfolgen:

- Du installiert eine 40G / Partition. Es funktioniert, da das Basis OS und alle seine Dateien (einschließlich /bsd) innerhalb der ersten 504M liegen.
- Du benutzt das System und endest mit mehr als 504M Dateien auf ihm.
- Du aktualisiert und erstellst deinen eigenen Kernel, was auch immer, und kopierst deinen neuen /bsd über den alten.
- Du startest neu.
- Du bekommst eine Meldung wie "ERR M" oder andere Probleme während dem Booten.

Warum? Weil, wenn du eine neue /bsd Datei "über" die alte kopierst, überschreibt sie nicht die alte, ihr wird einem neuen Ort auf der Platte zugewiesen, möglicherweise außerhalb der 504M Grenze, die das BIOS hat. Der Bootloader wird nun nicht mehr in der Lage sein, die /bsd Datei zu erhalten und das System hängt.

Um OpenBSD zum Booten zu bringen, müssen die Bootloader (biosboot(8) und `/boot` im Falle von i386) und der Kernel (`/bsd`) innerhalb des Bereiches sein, den die Boot ROM unterstützt und innerhalb ihrer eigenen Fähigkeiten. Um sicher zu gehen, ist die Regel einfach:

Die gesamte root Partition muss innerhalb des BIOS (oder Boot ROM) vom Computer adressierbaren Speicher liegen.

Einige nicht-i386 Anwender denken, dass sie dies nicht betrifft, jedoch haben die meisten Plattformen eine Art Begrenzung des ROMs bezüglich der Plattengröße. Herauszufinden, wie groß diese Begrenzung denn nun tatsächlich ist, kann schwer sein.

Dies ist ein weiterer guter Grund deine Festplatte zu partitionieren, statt nur eine große Partition zu verwenden.

fsck(8) Zeit- und Speicheranforderungen

Eine weitere Überlegung mit großen Dateisystemen ist die Zeit und der Speicher, die benötigt werden, um as Dateisystem nach einem Crash oder einer Stromunterbrechung einem fsck(8) zu unterziehen. Man sollte nicht ein 120G Dateisystem auf ein System mit 32M RAM setzen und erwarten, dass es fsck(1) erfolgreich nach einem Crash ausführt. Eine grobe Richtlinie ist, dass das System zumindest 1M Arbeitsspeicher für jedes 1G des Plattenspeichers haben sollte, um erfolgreich fsck gegen die Platte auszuführen. Die benötigte Zeit, um fsck gegen ein Laufwerk auszuführen kann ein Problem werden, sobald das Dateisystem an Größe gewinnt.

14.8 - Installieren von Bootblocks - i386 spezifisch

Ältere Versionen von MS-DOS können nur mit Festplattengeometrien von 1024 Zylindern oder weniger klarkommen. Da nahezu alle modernen Betriebssysteme mehr als 1024 Zylinder haben, haben die meisten SCSI BIOS Chips (die auf den SCSI Controller Karten) und IDE BIOSe (was Teil des restlichen PC BIOS ist) eine Option, manchmal auch als Grundeinstellung, die wirkliche Geometrie in etwas zu übersetzen, mit dem MS-DOS umgehen kann. Wie dem auch sei, nicht alle BIOS Chips "übersetzen" die Geometrie in der selben Weise. Wenn du dein BIOS wechselst (entweder mit einem neuen Motherboard oder einem neuen SCSI Controller), und das neue benutzt eine andere "übersetzte" Geometrie, wirst du nicht in der Lage sein, den ‚second stage' Bootloader zu laden (und kannst daher den Kernel auch nicht laden) (Das liegt daran, dass der ‚first stage' Bootloader eine Liste der Blöcke enthält, die /boot in der "übersetzten" Geometrie enthalten.) Falls du IDE Platten benutzt und du Änderungen an deinen BIOS Einstellungen machst, kannst du seine Übersetzung ebenfalls (ungewollt) ändern. (die meisten IDE BIOSe bieten 3 verschiedene Übersetzungen.) Um deinen Bootblock zu reparieren, damit du normal booten kannst, stecke einfach eine Bootfloppy in dein Diskettenlaufwerk und gib am Bootprompt "b hd0a:/bsd" ein, um ihn zu zwingen, von der ersten Festplatte zu booten (und nicht von der Floppy). Deine Maschine sollte normal booten. Jetzt musst du die erste Stufe des Bootloaders auf den neuen Stand bringen. (Und dazu passend den Bootblock schreiben.)
Unser Beispiel geht davon aus, dass deine Bootdisk sd0 ist (bei IDE wäre es wd0, etc..):

```
# cd /usr/mdec; ./installboot /boot biosboot sd0
```

Wenn eine neuere Version von bootblocks benötigt wird, wirst du diese selber kompilieren müssen. Und das geht so:

```
# cd /sys/arch/i386/stand/
# make && make install
# cd /usr/mdec; cp ./boot /boot
# ./installboot /boot biosboot sd0 (oder wie deine Festplatte auch immer heißen mag)
```

14.9 - Sich auf das Schlimmste vorbereiten: Backups und Wiederherstellen von Band.

Einführung:

Wenn du so etwas wie einen Poduktionsserver laufen lassen willst, ist es ratsam, irgendeine Form von Backup zu haben, für den Fall, dass eine deiner Festplatten versagt oder einen Crash hat.

Diese Information wird dir helfen, die Standard-Werkzeuge dump(8) und restore(8) zu benutzen, die als Teil von OpenBSD ausgeliefert werden. Ein fortgeschritteneres Werkzeug ist "Amanda", das auch mehrere Server auf ein Bandlaufwerk sichern kann. In den meisten Umgebungen sind dump(8)/restore(8) aber ausreichend. Wenn du aber mehrere Maschinen auf ein Band sichern willst, ist Amanda auf jeden Fall einen Blick wert.

Die Beispiele in diesem Dokument benutzen sowohl SCSI Festplatten als auch Bänder. In einer Produktionsumgebung empfehlen wir SCSI und kein IDE wegen der Art und Weise, wie IDE mit ‚bad blocks' umgeht. Das heißt aber nicht, dass diese Informationen nutzlos sind, wenn du IDE benutzt, sondern einzig deine Gerätenamen werden sich leicht unterscheiden. Zum Beispiel wäre sd0a in einem IDE-basierten System wd0a.

Backup auf's Band bringen:

Um sein Backup auf ein Band zu bringen, muss man wissen, wo die Dateisysteme gemountet sind. Das findet man mit dem "mount"-Kommando am Shell-Prompt heraus. Dabei sollte eine Ausgabe wie diese herauskommen:

```
# mount
/dev/sd0a on / type ffs (local)
/dev/sd0h on /usr type ffs (local)
```

In diesem Beispiel ist das root-Dateisystem (/) physikalisch auf sd0a, also auf der SCSI-Festplatte 0, Partition a. Das /usr Dateisystem befindet sich auf sd0h, also SCSI Festplatte 0, Partition h.

Ein weiteres Beispiel einer etwas größeren mount-Tabelle könnte so aussehen:

```
# mount
/dev/sd0a on / type ffs (local)
/dev/sd0d on /var type ffs (local)
/dev/sd0e on /home type ffs (local)
/dev/sd0h on /usr type ffs (local)
```

In diesem fortgeschritteneren Beispiel befindet sich das root (/) Dateisystem auf sd0a. Das /var Dateisystem befindet sich auf sd0d, das /home Dateisystem auf sd0e und schlussendlich /usr auf sd0h.

Um ein Backup deiner Maschine zu machen, musst du dump mit jeder festgelegten Partition füttern. Hier ist ein Beispiel der Kommandos, um die einfachere mount-Tabelle weiter oben zu sichern:

```
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0a
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0h
```

```
# mt -f /dev/rst0 rewind
```

Für die etwas fortgeschrittenere mount-Tabelle würde man etwas wie das hier benutzen:

```
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0a
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0d
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0e
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0h
# mt -f /dev/rst0 rewind
```

Du kannst die dump(8) Manual Seite ansehen, um genau zu erfahren, was jede Kommandozeilenoption macht. Hier ist eine kurze Übersicht über die Parameter, die oben verwendet wurden:

- **0** - Führe einen Level 0 Dump durch, hole alles
- **a** - Versuche automatisch die Bandlänge herauszufinden
- **u** - Bringe die Datei /etc/dumpdates auf den neuesten Stand, um zu reflektieren, wann die letzte Sicherung gemacht wurde
- **f** - Welches Bandlaufwerk benutzt werden soll (/dev/nrst0 in diesem Fall)

Zuletzt welche Partition gesichert werden soll (/dev/rsd0a, usw.)

Das mt(1) Kommando wird am Ende benutzt, um das Band zurückzuspulen. Sieh dir die mt Manual Seite an, wenn du mehr Informationen haben willst (wie etwa eject).

Wenn du dir nicht sicher bist, wie dein Bandlaufwerk heißt, benutze einfach dmesg, um das herauzufinden. Ein Beispieleintrag von dmesg für ein Bandlaufwerk könnte so aussehen:

```
st0 at scsibus0 targ 5 lun 0: <ARCHIVE, Python 28388-XXX, 5.28>
```

Du hast vielleicht bemerkt, dass bei der Sicherung das Bandlaufwerk als "nrst0" anstatt von "st0" bezeichnet wird, wie man es in dmesg sieht. Wenn du auf st0 statt nrst0 zugreifst, benutzt du das selbe physikalische Gerät, aber sagst ihm, es soll nicht zurückspulen, nachdem der Job im ‚raw' Modus beendet wurde. Um mehrere Dateien auf ein einziges Band zu sichern, stelle sicher, dass du nicht zurückspulst, sprich das richtige Gerät benutzt, ansonsten wirst du mit der zweiten Sicherung die erste überschreiben, usw. Du findest in der dump Manual Seite eine ausführlichere Beschreibung.

Wenn du ein kleines Skript namens "backup" schreiben würdest, könnte es z.B. so aussehen:

```
echo "  Starting Full Backup..."
/sbin/dump -0au -f /dev/nrst0 /dev/rsd0a
/sbin/dump -0au -f /dev/nrst0 /dev/rsd0d
/sbin/dump -0au -f /dev/nrst0 /dev/rsd0e
/sbin/dump -0au -f /dev/nrst0 /dev/rsd0h
echo
echo -n "  Rewinding Drive, Please wait..."
mt -f /dev/rst0 rewind
echo "Done."
echo
```

Wenn regelmäßige nächtliche Backups gefordert sind, könnte man cron(8) benutzen, um das Backup jede Nacht automatisch zu starten.

Es ist außerdem hilfreich, auf einem Blatt Papier aufzuschreiben, wie groß jedes Dateisystem sein muss. Du kannst df -h benutzen, um herauszufinden, wieviel Platz jede Partition momentan verbraucht. Das ist dann nützlich, wenn eine Platte versagt und du die Partitionstabelle auf der neuen Platte wieder erstellen musst.

Deine Daten wiederherzustellen hilft außerdem noch gegen Fragmentierung. Der beste Weg, um sicherzustellen, dass du alle Dateien erwischst, ist es, im ‚single user' Modus zu booten. Dateisysteme müssen nicht gemountet werden, um gesichert zu werden. Vergiss aber nicht root (/) zu mounten, denn sonst wird dein dump versagen, wenn er versucht, Dumpdaten zu schreiben. Gib einfach "`bsd -s`" am boot> Prompt ein, um in den ‚single user' Modus zu kommen.

Den Inhalt eines dump Bandes ansehen:

Nachdem du deine Dateisysteme zum ersten Mal gesichert hast, ist es sicher eine gute Idee, dein Band zu testen und sicherzustellen, dass es auch die Daten enthält, die darauf sein sollen.

Du kannst den folgenden Befehl benutzen, um eine Auflistung der Dateien auf einem dump Band zu bekommen:

```
# /sbin/restore -tvs 1 -f /dev/rst0
```

Das listet die Dateien auf der 1. Partition auf dem dump Band (dem Sicherungsband) auf. Wie in den Beispielen weiter oben, ist 1 dein root (/) Dateisystem.

Um den Inhalt der 2. Partition zu sehen und die Ausgabe in eine Datei umzulenken, würde man z.B. solch ein Kommando benutzen:

```
# /sbin/restore -tvs 2 -f /dev/rst0 > /home/me/list.txt
```

Wenn du eine mount-Tabelle wie die oben aufgeführte hast, wäre 2 /usr, wenn deine aber etwas größer wäre, könnte 2 auch /var sein oder

irgendwas anderes. Die Sequenznummer ist auf jeden Fall die gleiche Reihenfolge, in der das Dateisystem auf Band gesichert wird.

Wiederherstellen vom Band:

Das Beispielszenario wäre sinnvoll, wenn deine eigentliche Festplatte komplett ausgefallen wäre. Falls du aber nur eine einzige Datei wiederherstellen willst, sieh dir die restore Manual Seite genau an und achte besonders auf die Anweisungen zum interaktiven Modus.

Wenn du gut vorbereitet bist, kann der Prozess des Ersetzens einer Festplatte sehr schnell von statten gehen. Die Standard OpenBSD install/boot Floppy enthält bereits das benötigte restore Werkzeug, genauso wie die ausführbaren Dateien, um neue Partitionen zu erstellen und deine Festplatte bootfähig zu machen. In den meisten Fällen sind diese Floppies und dein Sicherungsband alles, was du brauchst, um wieder alles betriebsbereit zu bekommen.

Nachdem du das kaputte Laufwerk physikalisch ersetzt hast, sind die grundlegenden Schritte zur Wiederherstellung folgende:

- Boote von der OpenBSD install/boot Floppy. An der Menüauswahl wähle Shell. Nimm dein neuestes und schreibgeschütztes Band und lege es in dein Laufwerk ein.
- Benutze das fdisk(8) Kommando, um eine primäre OpenBSD Partition auf dieser neu installierten Festplatte zu erzeugen. Beispiel:

  ```
  # fdisk -e sd0
  ```

  Sieh einfach in die fdisk FAQ, um genaueres zu erfahren.
- Mit dem disklabel Kommando stellst du dann deine OpenBSD Partitionstabelle in der primären OpenBSD Partition wieder her, die du gerade mit fdisk erzeugt hast. Beispiel:

  ```
  # disklabel -E sd0
  ```

  (Vergiss den Swap nicht, siehe dazu die disklabel FAQ für weitere Informationen)
- Benutze das newfs Kommando, um ein neues sauberes Dateisystem auf jeder Partition zu erstellen, die du mit dem oben aufgeführten Schritten erstellt hast. Beispiel:

  ```
  # newfs /dev/rsd0a
  # newfs /dev/rsd0h
  ```
- Mounte dein neu vorbereitetes root (/) Dateisystem auf /mnt. Beispiel:

  ```
  # mount /dev/sd0a /mnt
  ```
- Wechsel in das gemountete root Dateisystem und beginne mit dem restore Prozess. Beispiel:

  ```
  # cd /mnt
  # restore -rs 1 -f /dev/rst0
  ```
- Wenn die Platte bootfähig sein soll, schreibe mit dem folgenden Befehl einen neuen MBR auf deine Festplatte. Beispiel:

  ```
  # fdisk -i sd0
  ```
- Zusätzlich zum Schreiben eines neuen MBR musst du Bootblöcke installieren, um davon booten zu können. Das folgende ist ein kurzes Beispiel:

  ```
  # cp /usr/mdec/boot /mnt/boot
  # /usr/mdec/installboot -v /mnt/boot /usr/mdec/biosboot sd0
  ```
- Dein neues root Dateisystem auf der eingebauten Festplatte sollte jetzt fertig sein, so dass du davon booten und damit beginnen kannst, den Rest der Dateien wiederherzustellen. Da dein Betriebssystem noch nicht komplett ist, solltest du alles im ‚single user' Modus wiederherstellen. Am Shellprompt benutze die folgende Kommandos, um deine Festplatten "abzumelden" (umount) und das System anzuhalten:

  ```
  # umount /mnt
  # halt
  ```
- Entferne die install/boot Floppy aus dem Laufwerk und reboote dein System. Am OpenBSD boot> Prompt benutze das folgende Kommando:

  ```
  boot> bsd -s
  ```

  Das bsd -s führt dazu, dass der Kernel im ‚single user' Modus gestartet wird, der nur ein root (/) Dateisystem braucht.
- Unter der Annahme, das du die obigen Schritte richtig ausgeführt hast und nichts schief gegangen ist, solltest du von einem Prompt begrüßt werden, der dich nach einem Pfad zu einer Shell fragt, oder du sollst Return drücken. Drücke Return, um die sh zu benutzen. Als nächstes willst du sicher root im r/w Modus (Schreib/Lese) remounten und nicht mehr im Nur-Lese-Modus benutzen (ro). Dazu benutze folgendes:

```
# mount -u -w /
```

- Sobald du im r/w Modus remountet hast, kannst du fortfahren, deine restlichen Dateisysteme wiederherzustellen. Beispiel:

```
(einfache mount-Tabelle)
# mount /dev/sd0h /usr; cd /usr; restore -rs 2 -f /dev/rst0

(umfassendere mount-Tabelle)
#       mount /dev/sd0d /var; cd /var; restore -rs 2 -f /dev/rst0
#       mount /dev/sd0e /home; cd /home; restore -rs 3 -f /dev/rst0
#       mount /dev/sd0h /usr; cd /usr; restore -rs 4 -f /dev/rst0
```

  Benutze "**restore rvsf**" statt eines einfachen rsf, um die Namen von Objekten zu sehen, während sie vom dump Set ausgepackt werden.

- Nachdem du jetzt auch alle Dateien der anderen Dateisysteme wiederhergestellt hast, führe einen Reboot in den ‚multi user' Modus durch. Wenn alles geklappt hat, sollte dein System wieder genauso sein, wie zum Zeitpunkt deiner letzten Sicherung, und bereit, wieder eingesetzt zu werden.

14.10 - Diskimages unter OpenBSD mounten

Um ein Diskimage (ISO-Images, Diskimages, die mit dd erstellt wurden, etc) unter OpenBSD zu mounten, musst du ein vnd(4) Device konfigurieren. Zum Beispiel, wenn du ein ISO-Image unter */tmp/ISO.image* hast, würdest du die folgenden Schritte machen, um es zu mounten:

```
# vnconfig svnd0 /tmp/ISO.image
# mount -t cd9660 /dev/svnd0c /mnt
```

Bedenke bitte, dass du den Typ *cd9660* angeben musst, wenn es eine CD ist. Das gilt aber auch für die anderen Typen, also musst du z.B. ffs beim Mounten eines Diskimages angeben.

Um das Image wieder zu unmounten, benutze die folgenden Kommandos.

```
# umount /mnt
# vnconfig -u svnd0
```

Mehr Informationen gibt es in der vnconfig(8) Manual Seite.

14.11 - Hilfe! Ich erhalte Fehler mit IDE DMA!

DMA IDE Übertragungen, die durch pciide(4) unterstützt werden, sind unzuverlässig. Bis vor kurzem wurden die meisten "mainstream" Betriebssysteme, die behaupteten, dass sie DMA Übertragungen mit IDE Laufwerken unterstützen, nicht mit standardmäßig aktivierter Unterstützung wegen unzuverlässiger Hardware ausgeliefert. Nun werden viele dieser gleichen Maschinen mit OpenBSD verwendet.

OpenBSD ist aggressiv und versucht, den höchsten DMA Modus zu benutzen, den es konfigurieren kann. Dies führt in einigen Konfigurationen zu Datenkorruptionen aufgrund von defekten Motherboard Chipsets, fehlerhaften Treibern, die buggy sind und/oder Lärm auf den Kabeln. Glücklicherweise schützt Ultra-DMA die Datenübertragungen mit einem CRC, um Korruptionen zu entdecken. Falls ein Fehler bei einem solchen Ultra-DMA CRC geschieht, wird OpenBSD eine Fehlermeldung ausgeben und erneut versuchen, die Daten zu übertragen.

```
wd2a:  aborted command, interface CRC error reading fsbn 64 of 64-79
(wd2 bn 127; cn 0 tn 2 sn 1), retrying
```

Nach ein paar Fehlversuchen wird OpenBSD zu einem langsameren (und damit hoffentlich zuverlässigeren) DMA-Modus herunterschalten. Nach den Ultra-DMA Modi wird dann zu einem PIO Modus heruntergeschaltet.

UDMA Fehler werden meistens durch minderwertige oder beschädigte Kabel verursacht. Kabelprobleme sollten normalerweise zuerst in Betracht gezogen werden, wenn du viele DMA Fehler oder unerwartet niedrige DMA Leistung erhälst. Es ist ebenfals eine schlechte Idee, das CD-ROM an den gleichen Kanal wie die Festplatte zu stecken.

Wenn das Ersetzen der Kabel nicht zur Lösung des Problems führt und OpenBSD nicht erfolgreich herunterschaltet oder der Prozess zu einem ‚hard lock' deiner Maschine führt, möchtest du vielleicht dein System auf einen niedrigereren DMA oder UDMA Level standardmäßig begrenzen. Dies kann unter Verwendung von UKC oder config(8) realisiert werden, indem man die Optionen des wd(4) Device ändert.

14.13 - RAID Optionen unter OpenBSD

RAID (Redundant Array of Inexpensive Disks) gibt die Möglichkeit, mehrere Laufwerke zu verwenden, um bessere Leistung, Kapazität und/oder Redundanz zu erhalten, als man aus einem einzelnen Laufwerk herausholen kann. Während eine vollständige Diskussion über die Vorteile und Risiken von RAID außerhalb des Rahmens dieses Artikels liegt, existieren einige Punkte, die so wichtig sind, dass sie nun besprochen werden sollten:

- RAID hat nichts mit Backup zu tun.
- RAID allein wird die Ausfallzeit nicht eliminieren.

Wenn diese Information neu für dich ist, ist das kein guter Ausgangspunkt, um RAID zu erforschen.

Software Möglichkeiten

OpenBSD beinhaltet RAIDframe, eine software-basierende RAID Lösung. Dokumentation hierfür kann an folgenden Stellen gefunden werden:

- FAQ 11, RAID
- RAIDframe Homepage
- Manual Seite für raidctl(8)
- Manual Seite für raid(4)

Die root Partition kann direkt von OpenBSD unter Verwendung der "Autoconfiguration" Option von RAIDframe gespiegelt werden.

Hardware Möglichkeiten

Viele OpenBSD Plattformen beinhalten Unterstützung für etliche Hardware RAID Produkte. Die Möglichkeiten variieren von Plattform zu Plattform, siehe die passende Hardwareunterstützungsseite (hier aufgelistet).

Eine andere Möglichkeit, die für viele Plattformen bereit steht, ist eine der vielen Produkte, die mehrere Laufwerke dazu bringt, wie ein großes IDE oder SCSI Laufwerk zu agieren und die dann in einen standardmäßigen IDE oder SCSI Adapter gesteckt werden. Diese Geräte können nahezu auf jeder Hardwareplattform funktionieren, die entweder SCSI oder IDE verwenden.

Einige Hersteller dieser Produkte:

- Arco
- Accusys
- Maxtronic
- Infortrend

(Hinweis: dies sind nur Produkte, die von OpenBSD Anwendern verwendet und gemeldet wurden -- dies ist weder eine Art von Werbung noch ist es eine ausführliche Liste.)

Keine Möglichkeit

Eine häufig gestellte Frage in den Mailinglisten ist "Werden die kostengünstigen IDE- oder SATA-RAID-Controller (wie zum Beispiel jene, die die Highpoint-, Promise- oder Adaptec-HostRAID-Chips benutzen) unterstützt?" Die Antwort ist "Nein". Diese Karten und Chips sind nicht echte Hardware RAID Controller, stattdessen eher BIOS-assistierte Bots für ein Software RAID. Da OpenBSD bereits Software RAID in einem Hardware-unabhängigen Weg unterstützt, besteht kein großes Verlangen bei den OpenBSD Entwicklern, diese spezielle Unterstützung für diese Karten zu implementieren.

Fast alle on-board SATA- oder IDE-‚RAID'-Controller sind von diesem softwarebasierendem Stil und funktionieren normalerweise einwandfrei als SATA- oder IDE-Controller unter Verwendung des standardmäßigen IDE-Treibers (pciide(4)), aber werden nicht als Hardware-RAID-System unter OpenBSD funktionieren.

14.14 - Warum sagt mir `df(1)`, dass ich mehr als 100% von meiner Platte belegt habe?

Leute sind manchmal erstaunt darüber, herausfinden zu müssen, dass sie *negativen* verfügbaren Plattenspeicher haben oder mehr als 100% einer Partition in Verwendung ist, wie es von df(1) angezeigt wird.

Wenn eine Partition mit newfs(8) erstellt wird, wird ein Teil des verfügbaren Speichers vor den normalen Benutzern in Reserve gehalten. Dies stellt einen Spielraum für Fehler bereit, wenn du versehentlich die Platte füllst und hilft, die Plattenfragmentierung auf einem Minimum zu halten. Standardwert hierfür ist 5% der Plattenkapazität, so dass, falls der root Anwender sorglos die Platte auffüllt, du bis zu 105% Speicher sehen kannst, der verwendet wird.

Wenn der 5% Wert für dich nicht angemessen erscheint, kannst du ihn mit dem tunefs(8) Kommando ändern.

[FAQ Index] [Zum Kapitel 12 - Plattform-spezifische Fragen]

---

www@openbsd.org

$OpenBSD: faq14.html,v 1.28 2005/02/12 20:36:53 jufi Exp $