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

Weitere Dokumente

Upgrade-MiniFAQ
Ports and Packages
Port Testing Guide
Updating via AnonCVS
Stable
CVSup
Manual pages
Bug Reporting
Mail lists
PF User's Guide

PDF Dateien

OpenBSD FAQ
PF FAQ

Text Dateien

OpenBSD FAQ
PF FAQ

Zurück zu OpenBSD

OpenBSD

Dokumentation und häufig gestellte Fragen (FAQs)

Häufige Probleme Neue Updates

Diese FAQ sind zusätzliche Informationen zur Dokumentation in den man pages, 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 3.3. Die Entwickler-Version (-current) von OpenBSD wird *nicht* von dieser FAQ behandelt.

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:

- 4 - Installationsanleitung
- 8 - Allgemeine Fragen
- 10 - Systemverwaltung
- 14 - Disk setup
- PF User's Guide

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

8 - Allgemeine Fragen

PF User's Guide

Häufige Probleme

- Soll ich nun Ports oder Packages benutzen?
- Wie konfiguriere ich ein Multi-Boot System?
- Probleme mit grossen Laufwerken und OpenBSD

Neueste Updates

- FAQ 1, Was enthält OpenBSD?
- FAQ 8, Soll ich nun Ports oder Packages benutzen?
- FAQ 5, Häufige Probleme beim Übersetzen und Erzeugen
- Neue PF FAQ!
- FAQ 4, Wie bekomme ich ein dmesg(8), um damit ein Problem mit der Installation zu melden ?
- FAQ 4 Anpassen des Installationsprozesses
- FAQ 4, Wie installiere ich eine Anzahl gleichartiger Systeme?

Der FAQ maintainer ist Nick Holland.
Weitere Mitarbeiter an der FAQ sind Joel Knight, Eric Jackson, Wim Vandeputte und Chris Cappuccio.

Information über die Übersetzung dieser FAQ und den Rest der OpenBSD Website finden sich auf der Translation page.

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

Back to OpenBSD

Originally [OpenBSD: index.html,v 1.188]
$Translation: index.html,v 1.90 2003/12/15 13:46:18 grunk Exp $
$OpenBSD: index.html,v 1.79 2003/12/15 19:16:04 horacio 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] [Zu Sektion 2 - Andere Informationsquellen über 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.

1.2 - Auf welchen Systemen läuft OpenBSD?

OpenBSD 3.3 läuft auf den folgenden Plattformen:

- i386 - von CD bootfähig
- sparc - von CD bootfähig
- sparc64 - von CD bootfähig
- hp300 - nur FTP
- mac68k - nur FTP
- macppc - von CD bootfähig
- mvme68k - nur FTP
- alpha - nur FTP
- vax

- hppa - Nur FTP* *Neu mit 3.3 **

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 in Kapitel 3 dieser FAQ.

Vorige 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
- mvme88k
- pmax

OpenBSD unterstützt zur Zeit nicht mehr als eine CPU. In der FAQ 8, SMP findest du dazu weitere Informationen.

1.3 - Ist OpenBSD wirklich frei?

OpenBSD is ganz 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.

Für weitere Informationen über populäre Lizenzen siehe:
http://www.openbsd.org/policy.html.

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 Resourcen 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 1,5 Jahre langen ausführlichen Quelltextsicherheitsaudits durch ein 10-Mann-Team 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.
- OpenBSD bietet die Möglichkeit für gewöhnliche Menschen, aktiv an der Entwicklung und Erprobung des Produktes teilzuhaben und teilzunehmen.

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. Sie 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 den angestrebten Daten 1. Mai und 1. November. Mehr Informationen zum Entwicklungsprozess gibt es hier.

[FAQ Index] [Zu Kapitel 2 - Andere Informationsquellen]

 www@openbsd.org

Originally [OpenBSD: faq1.html,v 1.48]
$Translation: faq1.html,v 1.48 2003/05/28 13:39:52 jufi Exp $
$OpenBSD: faq1.html,v 1.41 2003/05/28 14:03:30 jufi Exp $

2 - Andere Informationsquellen über OpenBSD

Inhaltsverzeichnis

2.1 - Interessante WWW Seiten

Die offizielle WWW Seite des OpenBSD Projekts findest du unter: http://www.OpenBSD.org.

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

Zusätzlich Hinweise für Laptop Benutzer kann man hier finden:
http://www.monkey.org/openbsd-mobile/.

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 einzugetragen. Zum Beispiel:

```
subscribe announce
```

Der Mailinglistendienst wird dir antworten und dich um Bestätigung bitten. Die Bestätigung schickst du zurück an den Mailinglistendienst. Du bekommst die Option eines Hypertextlink oder das Zurücksenden einer Antwort, um deine Absicht zu bestätigen. Eine Email-Antwort sollte etwas wie das folgende enthalten:

```
accept A56D-70D4-52C3
```

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

Um sich 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. Einige der beliebtesten OpenBSD Mailinglisten sind:

- **announce** - Wichtige Ankündigungen. Sehr wenige e-mails.
- **security-announce** - Bekanntmachungen von Sicherheitsemfpehlungen.
- **misc** - Benutzerfragen und Antworten. Dies ist die aktivste Liste, und sollte daher für fast alle Fragen genutzt werden.
- **tech** - Technische Themen für OpenBSD Entwickler und fortgeschrittene User. Bitte leite 'neue User' und Installations-bezogene Fragen an *misc*, und nicht an *tech*. Und bitte niemals gleichzeitig in *misc* und *tech*.
- **bugs** - Via sendbug(1) eingereichte Fehler und Diskussionen darüber.
- **source-changes** - Automatisierter Verteiler von Änderungen des CVS der Quelltexte.
- **ports** - Diskussionen über das OpenBSD Ports Systems.
- **ports-changes** - Automatische Mails der ports-spezifischen CVS source tree Änderungen.

- **advocacy** - Diskussionen, um die Nutzung von OpenBSD populärer zu machen.

Bevor du eine Frage auf **misc** oder einer anderen Liste postest, überprüfe bitte zunächst, ob deine Frage nicht schon in den Archiven auftaucht, die meisten Fragen sind schon einmal gerstellt worden. Auch wenn du das Problem zum ersten Mal siehst, ist es anderen dich oftmal schon begegnet, wurden die Mailinglisten vielleicht schon letzte Woche damit überschwemmt, und freuen sich nicht auf eine Wiederholung. Einige Archive und Mailinglisten-Richtlinien finden sich unter http://www.openbsd.org/mail.html

Eine weitere interessante Mailingliste ist openbsd-mobile@monkey.org, die sich mit der Nutzung von OpenBSD auf Notebooks und Laptops befasst.

Um diese Liste zu abonnieren:

'echo subscribe | mail "openbsd-mobile-request@monkey.org"'

Das Archiv dieser Liste findest du unter: http://www.monkey.org/openbsd-mobile/archive/

2.3 - Manualseiten

OpenBSD wird von einer ausführlichen Dokumentation in Form von Manualseiten (manual pages) und weiteren, längeren Artikeln, die sich auf spezielle Anwendungen beziehen, begleitet. Um die Manualseiten lesen zu können, müssen die man und misc tar balls installiert sein.

Hier eine Liste der nützlichsten Manualseiten für Anfänger:

- **help(1)** - Hilfe für neue User und Administratoren.
- **afterboot(8)** - Dinge, die man nach dem 1. Start beachten sollte.
- **boot(8)** - Systemstartprozeduren.
- **login.conf(5)** - Format der Passwortkonfigurationsdatei.
- **adduser(8)** - Befehle, um neue Benutzer anzulegen.
- **vipw(8)** - Editieren der Passwortdatei.
- **man(1)** - Anzeigen der on-line Manualseiten.
- **sendbug(1)** - Schicken eines Fehlerberichtes über OpenBSD an die Supportzentrale.
- **disklabel(8)** - Auslesen und Schreiben des Disklayouts.
- **ifconfig(8)** - Konfiguration der Netzwerkadapter.
- **route(8)** - Manipulation der Routingtabellen.
- **netstat(1)** - Anzeigen des Netzwerkstatus.
- **reboot, halt(8)** - Restarten und Stoppen des Systems.
- **shutdown(8)** - Runterfahren des Systems zu einer bestimmten Zeit.
- **boot_config(8)** - Änderung der Kernelkonfiguration beim Starten.

Die OpenBSD Manualseiten sind im Web über http://www.openbsd.org/cgi-bin/man.cgi zu finden, oder auch auf deinem eigenen Computer, wenn du die Datei `man33.tgz` 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 man page des vi Editors. Solltest du den genauen Namen des Befehls nicht wissen oder "man befehl" nicht die gewünschte Manual Seite bringen, 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 Manualseiten 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 Manualseiten (enthalten in der misc Distribution) gibt es noch einen weiteren Dokumentensatz 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 fürs Lesen auf einer Terminalanzeige formatieren. In jedem Makefile mußt du nur die Option `-Tascii` bei jedem der groff(1) Befehle setzen (oder händisch ausführen). Einige der Dokumente verwenden `ms` Formatierungsmakros, andere die `me` Makros. Das Makefile jedes Dokumentenunterverzeichnisses `(zb: /usr/share/doc/usd/04.csh/Makefile)` zeigt dir das zu verwendende Format an. Zum Beispiel:

```
# cd /usr/share/doc/usd/04.csh
# groff -Tascii -ms tabs csh.1 csh.2 csh.3 csh.4 csh.a csh.g > csh.txt
# more csh.txt
```

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

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

Wie kann ich ein "man page" Quelltext anzeigen? (d.h., eine Datei, deren Namen in einer Zahl endet, wie tcpdump.8).

Dies gilt für den gesamten Quelltextbaum. Die "man pages" befinden sich unformatiert im Verzeichnisbaum und werden automatisch durch CVS upgedated. Um sich diese Seiten anzusehen:

```
# nroff -mandoc <file> | 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 <command> | col -b
```

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

Beachte, daß `[man_quelltext_datei]` die "man page" Quelltextdatei ist (wahrscheinlich eine Datei, die in einer Zahl endet; z.B. `tcpdump.8`). Die Postscriptversionen der Manualseiten 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 groff(1) Befehlsoptionen um eine PostScript Version von einer OpenBSD System man page zu bekommen:

```
# groff -mandoc -Tps [man_src_file] > outfile.ps
```

2.4 - Fehler berichten

Bevor du einen Fehler berichtest, 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 CDROM 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 Bug 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 Samrt 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 zum 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 ein normaler Benutzer ein funktionierendes OpenBSD System, von dem er einen Fehlerbericht abschicken will, hat,

dann sollte er sendbug(1) verwenden, um den Fehlerbericht zu verfassen und zum GNATS Problemerfassungssystem zu senden. Wenn sein System nicht startet, dann sollte er mit sendbug(1) seinen Fehlerbericht an das GNATS Problemerfassungssystem melden. Aber du solltest es, wenn möglich, immer verwenden. Du wirst noch zusätzliche Informationen, wie z. B. was geschah, deine genaue Konfiguration inkludieren 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 Email versenden zu können.

Nach dem Einsenden eines Bug-Report via sendbug(1) wirst du per Email 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 auch den Status in der Bug Report online abfragen, und zwar im Bug Tracking System.

[FAQ Index] [Zu Kapitel 1 - Einführung zu OpenBSD] [Zu Kapitel 3 - OpenBSD beziehen]


www@openbsd.org

Originally [OpenBSD: faq2.html,v 1.62]
$Translation: faq2.html,v 1.41 2003/07/27 15:48:47 jufi Exp $
$OpenBSD: faq2.html,v 1.36 2003/07/27 15:59:26 jufi Exp $

3 - OpenBSD beziehen

Inhaltsverzeichnis

3.1 - OpenBSD CD kaufen

Eine OpenBSD CD zu kaufen ist der allgemein beste Weg, mit deiner Unterstützung zu beginnen. Hier ist die Bestellseite: http://www.openbsd.org/de/orders.html.

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

- CD Verkäufe unterstützten 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 erwerben

Ja, OpenBSD hat T-Shirts zu deinem Tragevergnügen. Hier kannst du sie sehen: http://www.OpenBSD.org/de/tshirts.html. Nett, oder? :)

3.3 - Gibt es ein OpenBSD ISO image?

Einige andere OpenSource 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 weiterer 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 `cd34.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 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:

http://www.openbsd.org/de/ftp.html.

3.5 - Aktueller Quelltext (CVS)

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:

http://www.openbsd.org/de/anoncvs.html.

sowie auch in der FAQ 8, CVS.

Sollte deine Netzwerkanbindung für AnonCVS unzureichend sein oder dein Internetzugang besteht via UUCP, 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:

http://www.openbsd.org/de/ctm.html.

Eine weitere Möglichkeit an den Quelltext zu kommen bietet dir das Webinterface "cvsweb" an:

http://www.openbsd.org/cgi-bin/cvsweb/.

[FAQ Index] [Zu Kapitel 2 - Andere Informationsquellen] [Zu Kapitel 4 - Installationsleitfaden]


www@openbsd.org

Originally [OpenBSD: faq3.html,v 1.41]
$Translation: faq3.html,v 1.36 2003/12/13 19:50:59 jufi Exp $
$OpenBSD: faq3.html,v 1.32 2003/12/13 20:23:03 jufi Exp $

4 - OpenBSD 3.4 Installation Guide

Table of Contents

4.1 - Overview of the OpenBSD installation procedure

OpenBSD has a robust and adaptable text-based installation procedure, and can be installed from a single floppy disk. Most platforms follow a similar installation procedure; however there are some differences in the details. In all cases, you are urged to read the platform-specific INSTALL document in the *platform* directory on the CD-ROM or FTP sites (for example, `i386/INSTALL.i386`, `mac68k/INSTALL.mac68k` or `sparc/INSTALL.sparc`).

On most platforms, the OpenBSD installation uses a special kernel with a number of utilities and install scripts embedded in

a preloaded RAM disk. After this kernel is booted, the operating system is extracted from a number of compressed tar(1) (`.tgz`) files. There are several ways to boot this install kernel:

- **Floppy disk:** Floppy disk images are provided which can be used to create an install floppy on another Unix-like system, or on a DOS/Windows system. Typical file names are `floppy34.fs`, though several platforms have multiple floppy images available.
- **CD-ROM:** On several platforms a CD-ROM image (`cd34.iso`) is provided allowing creation of a bootable CD-ROM. This just contains the installation kernel - install files must still be retrieved via FTP or other source. You can, of course, build your own CD-ROM with whatever files and tools you desire.
- **bsd.rd:** The RAM disk kernel, intended for booting off either an already existing OpenBSD partition or booting over the network.
- **Network:** Some platforms support booting over a network.
- **Writing a file system image to disk:** a filesystem image that can be written to an existing partition, and then can be booted.
- **Bootable Tape:** Some platforms support booting from tape. These tapes can be made following the INSTALL.*platform* instructions.

Not every platform supports all boot options:

- **alpha:** Floppy, CD-ROM, writing a floppy image to hard disk.
- **hp300:** CD-ROM, network.
- **hppa:** Network.
- **i386:** Floppy, CD.
- **mac68k:** Installed (and booted) using MacOS utilities. See INSTALL.mac68k for details.
- **macppc:** CD-ROM, network.
- **mvme68k:** Network, bootable tape.
- **sparc:** Floppy, CD-ROM, network, writing image to existing swap partition, bootable tape.
- **sparc64:** Floppy (U1/U2 only), CD-ROM, network, writing image to existing partition.
- **vax:** Floppy, network.

All platforms other than mac68k can also use a bsd.rd to reinstall or upgrade.

Once the install kernel is booted, you have several options of where to get the install file sets. Again, not every platform supports every option.

- **CD-ROM:** Of course, we prefer you use the Official CD-ROM set, but for special needs, you can also make your own.
- **FTP:** Either one of the OpenBSD FTP mirror sites or your own local FTP server holding the file sets.
- **HTTP:** Either one of the OpenBSD HTTP mirror sites or your own local web server holding the file sets.
- **Local disk partition:** In many cases, you can install file sets from another partition on a local hard disk. For example, on i386, you can install from a FAT partition or a CD-ROM formatted in ISO9660, Rock Ridge or Joliet format. In some cases, you will have to manually mount the file system before using it.
- **NFS:** Some platforms support using NFS mounts for the file sets.
- **Tape:** File sets can also be read from a supported tape.

4.2 - Pre-installation checklist

Before you start your install, you should have some idea what you want to end up with. You will want to know the following items, at least:

- Machine name
- Hardware installed and available
  - Verify compatibility with your platform's hardware compatibility page
  - If ISA, you also need to know hardware settings, and confirm they are as OpenBSD requires.
- Install method to be used (CD-ROM, FTP, etc.)
- How will the system be updated and patched?
  - If done locally, you will need to have sufficient space available for the source tree and building it.
  - Otherwise, you will need access to another machine to build a patched release on.
- Desired disk layout

    - Does existing data need to be saved elsewhere?
    - Will OpenBSD co-exist on this system with another OS? If so, how both will be booted? Will you need to install a "boot manager"?
    - Will the entire disk be used for OpenBSD, or do you want to keep an existing partition/OS (or space for a future one)?
    - How do you wish to sub-partition the OpenBSD part of your disk?
- Network settings, if not using DHCP:
    - Domain name
    - Domain Name Server(s) (DNS) address
    - IP addresses and subnet masks for each NIC
    - Gateway address
- Will you be running the X Window System?

4.3 - Creating bootable OpenBSD install media

As examples, we will look at the installation images available for the i386 and sparc platforms.

The i386 platform has five separate installation disk images to choose from:

- **`floppy34.fs`** (Desktop PC) supports many PCI and ISA NICs, IDE and simple SCSI adapters and some PCMCIA support. *Most* users will use this image.
- **`floppyB34.fs`** (Servers) supports many RAID controllers, and some of the less common SCSI adapters. However, support for many standard SCSI adapters and many EISA and ISA NICS has been removed.
- **`floppyC34.fs`** (Laptops) supports the CardBus and PCMCIA devices found in many laptops.
- **`cdrom34.fs`** is, in effect a combination of all three boot disks. It can be used to make a bootable 2.88M floppy, or more commonly, as a boot image for a custom recordable CD.
- **`cd34.iso`** is an ISO9660 image that can be used to create a bootable CD with most popular CD-ROM creation software on most platforms. This is `cdrom34.fs` in a "ready-to-record" format.

Yes, there may be situations where one install disk is required to support your SCSI adapter and another disk is required to support your network adapter. Fortunately, this is a rare event, and can usually be worked around.

The sparc platform has three separate installation disk images to choose from:

- **`floppy34.fs:`** Supports systems with a floppy disk.
- **`cd34.iso`** An ISO image usable to make your own CD for booting SPARC systems with a CD-ROM.
- **`miniroot34.fs`** Can be written to a swap partition and booted.

4.3.1 - Creating floppies on Unix

To create a formatted floppy, use the fdformat(1) command to both format and check for bad sectors.

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

If your output is like the above example, then the disk is OK. However, if you do not see ALL "Vr"'s then the disk is most likely bad, and you should try a new one.

Note that some Unix-like systems have different commands for formatting floppies. Refer to your system's documentation for the exact procedure.

Once you have a clean, formatted floppy it is time to write the installation image to floppy. For this, you can use the dd(1) utility. An example usage of `dd(1)` is below:

```
# dd if=floppy34.fs of=/dev/rfd0c bs=32k
```

Once the image is written, check to make sure that the copied image is the same as the original with the cmp(1) command. If the diskette is identical to the image, you will just see another prompt.

```
# cmp /dev/rfd0c floppy34.fs
```

4.3.2 - Creating floppies on Windows or DOS

This section describes how to write the installation images to floppy disk under Windows or DOS. You can get the tools mentioned below from the tools directory on any of the ftp mirrors, or from the `3.4/tools` directory on CD1 of the OpenBSD CD set.

To prepare a floppy in MS-DOS or Windows, first use the native formatting tools to format the disk.

To write the installation image to the prepared floppy you can use *rawrite*, *fdimage*, or *ntrw*. *rawrite* will not work on Windows NT, 2000 or XP.

Note that `FDIMAGE.EXE` and `RAWRITE.EXE` are both MS-DOS applications, and thus are limited to MS-DOS's "8.3" file naming convention. As `floppyB34.fs` and `floppyC34.fs` have longer file names, you will have to find out how your system stored the file in "8.3 format" before using `FDIMAGE.EXE` or `RAWRITE.EXE` to make your boot floppies.

Example usage of *rawrite*:

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

Enter source file name: floppy34.fs
Enter destination drive: a
Please insert a formatted diskette into drive A: and press -ENTER- : Enter
```

Example usage of *fdimage*:

```
C:\> fdimage -q floppy34.fs a:
```

Example usage of *ntrw*:

```
C:\> ntrw floppy34.fs a:
3.5", 1.44MB, 512 bytes/sector
bufsize is 9216
1474560 bytes written
```

4.3.3 - Making a CD-ROM

You can create a CD-ROM using either the `cd34.iso` file or, in the case of the i386 platform, you can also use the `cdrom34.fs` as the bootable floppy image that is used to boot an i386 system from CD-ROM. The exact details here are left to the reader to determine with the tools they have at their disposal.

Some of the tools in OpenBSD are:

- mkhybrid(8)
- cdrecord, part of the cdrtools collection in the OpenBSD Ports System.

4.4 - Booting OpenBSD install media

Booting i386

Booting an install image on the i386 PC platform is nothing new to most people. If you are using a floppy disk, simply insert the floppy into the floppy drive and boot the system. The install image will then load, provided floppy boot is enabled in your BIOS. If you want to boot from CD, you must go into your system's BIOS and set the boot options to allow booting from CD. Some older BIOSes do not have this option, and you must use a floppy for booting your installation image. Don't worry though; even if you boot from floppy you can still install from the CD.

Booting sparc/sparc64

NOTE: On the sparc64 platform, only the SBus machines (Ultra 1, Ultra 2) are bootable from floppy.

To boot from floppy, place the floppy disk with the OpenBSD installation image on it into the floppy drive. Then use the following command to boot from the floppy:

```
ok boot floppy
```

To boot from CD-ROM, place the OpenBSD CD-ROM disk into the drive. If your Sun only has one CD-ROM drive, then just go to the boot prompt, where you can '`boot cdrom`':

```
ok boot cdrom
```

Of course, this will only work in new command mode. If you are at the old command mode prompt (a right arrow), type 'n' for the new command mode. (If you are using an old sparc that is pre-sun4c, you probably don't have a new command mode. In this case, you need to experiment.) If you have multiple CD-ROM devices, you need to boot from the correct one. Try `probe-scsi` from the new command mode.

```
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
```

Figure out which disk is the CD-ROM you want to boot from. Note the target number.

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

4.5 - Performing an install

4.5.1 - Starting the install

Whatever your means of booting is, it is now time to use it. During the boot process, the kernel and all of the programs used to install OpenBSD are loaded into memory. The most common problem when booting is a bad floppy disk or a drive alignment problem. The boot floppy is quite tightly packed -- any bad spot will cause problems.

At almost any point during the OpenBSD install process, you can terminate the current install attempt by hitting CTRL-C and can restart it without rebooting by running `install` at the shell prompt.

When your boot is successful, you will see a lot of text messages scroll by. This text, on many architectures in white on blue, is the dmesg, the kernel telling you what devices have been found, and where. Don't worry about remembering this text, as a copy is saved as `/var/run/dmesg.boot`. On some architectures, SHIFT+PGUP will let you examine text that has scrolled off the screen.

Then, you will see the following:

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

And with that, we reach our first question. Most of the time, you have the three options shown:

- **Install:** load OpenBSD onto the system, overwriting whatever may have been there. Note that it is possible to leave some partitions untouched in this process, such as a `/home`, but otherwise, assume everything else is overwritten.
- **Upgrade:** Install a new set of install files on this machine, but do not overwrite any configuration information, user data, or additional programs. No disk formatting is done, nor are the /etc or /var directories overwritten. A few important notes:
  - You will not be given the option of installing the `etc34.tgz` file. After the install, you will have to manually merge the changes of `etc34.tgz` into your system before you can expect it to be fully functional. This is an important step which must be done, as otherwise certain key services (such as pf(4)) may not start.
  - The Upgrade process is not designed to skip releases! While this will often work, it is not supported. For OpenBSD 3.4, upgrading 3.3 to 3.4 is the only supported upgrade. If you have to upgrade from an older version, a complete reinstall is recommended.
  - The i386 platform has switched to the ELF binary format, and will require uninstalling all installed packages and ports before upgrade. Reinstallation is *highly* recommended over upgrade.
- **Shell:** Sometimes, you need to perform repairs or maintenance to a system which will not (or should not) boot to a normal kernel. This option will allow you to do maintenance to the system.

On occasion, you will not see the "Upgrade" option listed. After a *flag day* event, it is not possible to directly upgrade; one

must rebuild the system from scratch.

In this example, we will do an install, but the upgrade process is similar.

```
Welcome to the OpenBSD/i386 3.4 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]
Do you wish to select a keyboard encoding table? [n] ENTER
```

In most cases, the default terminal type is appropriate; however if you are using a serial console for install, don't just take the default, respond appropriately.

If you do not select a keyboard encoding table, a US keyboard layout will be assumed.

```
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? [n] y
```

If you take the default here, the install process will terminate and drop you to a shell prompt.

4.5.2 - Setting up disks

Setting up disks in OpenBSD varies a bit between platforms. For i386 and macppc, disk setup is done in two stages. First, the OpenBSD slice of the hard disk is defined using fdisk(8), then that slice is subdivided into OpenBSD partitions using disklabel(8).

Some users may be a little confused by the terminology used here. It will appear we are using the word "partition" in two different ways. This observation is correct. There are two layers of partitioning in several OpenBSD platforms, the first, one could consider the Operating System partitioning, which is how multiple OSs on one computer mark out their own space on the disk, and the second one is how the OpenBSD partition is sub-partitioned into individual filesystems. The first layer is visible as a disk partition to DOS, Windows, and any other OS that can coexist with other Operating Systems on the IBM AT descended machines. The second layer of partitioning is visible only to OpenBSD and those OSs which can directly read an OpenBSD filesystem.

```
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
```

The root disk is the disk the system will boot from, and normally where swap space resides. Usually, this will be the default -- if it isn't, you will need to know how to force your computer to boot from a non-standard disk. IDE disks will show up as `wd0`, `wd1`, etc., SCSI disks and RAID devices will show up as `sd0`, `sd1`, and so on. All the disks OpenBSD can find are listed here -- if you have drives which are not showing up, you have unsupported or improperly configured hardware.

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

If you say "yes" to this question, the entire disk will be allocated to OpenBSD. This will result in a standard Master Boot Record and partition table being written out to disk -- one partition, the size of the entire hard disk, set to the OpenBSD partition type, and flagged as the bootable partition. This will be a common choice for most production uses of OpenBSD; however, there are some systems this should not be done on. Many Compaq systems, many laptops, some Dell and other

systems use a "maintenance" or "Suspend to Disk" partition, which should be kept intact. If your system has any other partitions of any type you do not wish to erase, do not select "yes" to the above question.

For the sake of this example, we will assume the disk is to be split between OpenBSD and a pre-existing Windows 2000 partition, so we take the default of "no", which will take us into the fdisk(8) program. You can also get more information on fdisk(8) here.

Important Note: Users with a large hard disk (larger than 8G on a newer i386, though on older machines and different platforms, often much smaller) will want to see this section before going any further.

```
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>
```

A few commands are worthy of elaboration:

- **`r`** or **`reinit:`** Clears existing partition table, makes one big OpenBSD partition, flags it active. Equivalent to saying "yes" to the "`use *all* of ...`" question.
- **`p`** or **`print:`** Displays the current partition table in sectors. "`p m`" will show the partition table in megabytes, "`p g`" will show it in gigabytes.
- **`e`** or **`edit:`** edit or alter a table entry.
- **`f`** or **`flag:`** Marks a partition as the active partition, the one that will be booted from
- **`exit`** and **`quit:`** Careful on these, as some users are used to "exit" and "quit" having opposite meanings.

It is worth pointing out once again, a error here will result in significant data loss. If you are going to do this on a drive with important data, it might be worth practicing on a "disposable" drive, in addition to having a good backup.

Our drive here has a 1.5G partition for Windows 2000 (using the FAT filesystem). Looking at the info from the above display, we can see that the Windows partition occupies through cylinder 202 on the drive. So, we are going to allocate the rest of the disk to OpenBSD, starting at cylinder 203. You could also calculate OpenBSD's starting sector of 3069360 by adding the existing partition's starting sector (63) and its size (3069297).

You can edit the drive layout in either Cylinder/Heads/Sectors form or just raw sectors. Which is easier depends upon what you are doing; in this case, working around an existing partition, using CHS format will probably be easier.

```
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>
```

Note that the prompt changed to include an asterisk ('*') to indicate you have unsaved changes. As we can see from the output of `p m` we have not altered our Windows partition, we have successfully allocated the rest of the drive for OpenBSD, and the partitions do not overlap. We are in business. Almost.

What we haven't done is flagged the partition as active so the machine will boot OpenBSD on the next reboot:

```
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>
```

And now, we are ready to save our changes:

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

Creating a disklabel

The next step is to use disklabel(8) to slice up the OpenBSD partition. More details on using disklabel(8) can be found in 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:
        p [unit]  - print label.
        M         - show entire OpenBSD man page for disklabel.
        e         - edit drive parameters.
        a [part]  - add new partition.
        b         - set OpenBSD disk boundaries.
        c [part]  - change partition size.
        d [part]  - delete partition.
        D         - set label to default.
        g [d|b]   - Use [d]isk or [b]ios geometry.
        m [part]  - modify existing partition.
        n [part]  - set the mount point for a partition.
        r         - recalculate free space.
        u         - undo last change.
        s [path]  - save label to file.
        w         - write label to disk.
        q         - quit and save changes.
        x         - exit without saving changes.
        X         - toggle expert mode.
        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).
        Non-sector units will be rounded to the nearest cylinder.
Entering '?' at most prompts will give you (simple) context sensitive help.
>
```

Again, a few of these commands could use a little elaboration:

- **p** - displays (prints) the current disklabel to the screen, and you can use the modifiers `k`, `m` or `g` for kilobytes,

megabytes or gigabytes.

- **`D`** - Clears any existing disklabel, creates a new default disklabel which covers just the current OpenBSD partition. This can be useful if the disk previously had a disklabel on it, and the OpenBSD partition was recreated to a different size -- the old disk label may not get deleted, and may cause confusion.
- **`m`** - Modifies an existing entry in a disklabel. Do not over estimate what this will do for you. While it may alter the size of a disklabel partition, it will NOT alter the filesystem on the drive. Using this option and expecting it to resize existing partitions is a good way of losing large amounts of data.

Slicing up your disk properly is important. The answer to the question, "How should I partition my system?" is "Exactly how you need it". This will vary from application to application. There is no universal answer. If you are unsure of how you want to partition your system, see this discussion.

In this system, we have over 17G available for OpenBSD. That's a lot of space, and it isn't likely we will need most of it. So, we will deliberately not use absolute minimum sizes. We would rather have a few hundred megabytes of unused space than a kilobyte too little.

On the root disk, the two partitions 'a' and 'b' **must** be created. The installation process will not proceed until these two partitions are available. 'a' will be used for the root filesystem (/) and 'b' will be used as swap space.

After a little thought, we decide to create just enough partitions to allow the creation of the recommended separate filesystems (`/, /tmp, /var, /usr, /home`) along with a swap partition:

- ***wd0a:*** / (root) - 150M. Should be more than enough.
- ***wd0b:*** (swap) - 300M.
- ***wd0d: /tmp*** - 120M. `/tmp` is used for building some software, 120M will probably be enough for most things.
- ***wd0e: /var*** - 80M. If this were to be a web or mail server, we'd have made this partition much larger, but, that's not what we are doing.
- ***wd0g: /usr*** - 2G. We want this partition to be large enough to load quite a few user applications, plus be able to update and rebuild the system if desired or needed. The Ports tree will be here as well, which will take almost 100M of this space before ports are built.
- ***wd0h: /home*** - 4G. This will allow plenty of user file space.

Now, if you add those up, you will see over 10G of space is unused! Unused space won't hurt anything, and it gives us flexibility to enlarge things in the future if need be. Need more */tmp*? No problem, create a new one in the unused space, change */etc/fstab* and problem solved.

```
> 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
```

You will note there is a *c* partition we seem to have ignored. This partition is your entire hard disk; don't attempt to alter it. You will also note the *i* partition wasn't defined by us; this is the pre-existing Windows 2000 partition. Partitions are not assigned any particular letters -- with the exception of *a* (root), *b* (swap) and *c* (entire disk), the rest of the partitions (through letter *p*) are available for use as you desire.

If you look closely at the output of the disklabel, you will note that your drive RPM rating is probably wrong. This is historical; the drive speed is not used in any way by the system. Do not worry about it.

Configuring your mount points and formatting your filesystems

Now comes the final configuration of your mount points. If you configured the mount points through disklabel(8), this step consists of just verifying your selections; otherwise, you can specify them now.

```
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? [n] 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 2 002)
/dev/wd0h on /mnt/home type ffs (rw, asynchronous, local, nodev, nosuid,
 ctime=Thu Oct 10 21:50:36 2002)
/dev/wd0d on /mnt/tmp type ffs (rw, asynchronous, local, nodev, nosuid,
ctime=Thu Oct 10 21:50:36 2002)
/dev/wd0g on /mnt/usr type ffs (rw, asynchronous, local, nodev, ctime=Th
u Oct 10 21:50:36 2002)
/dev/wd0e on /mnt/var type ffs (rw, asynchronous, local, nodev, nosuid,
ctime=Th u Oct 10 21:50:36 2002)
```

You may wonder why the installer again asks for mount points. This allows you to recover from any errors or omissions in the mount points specified during the creation of the disklabel. For instance, the installation process will automatically delete any duplicate mount points you enter during the configuration of the disklabel. The disklabel program will allow you to enter such duplicates, and thus they must be checked for after the disklabel program exits. The deleted duplicate mount points will result in partitions without mount points, that you must assign new mount points for if you wish to use the space.

Notice the "Are you really sure that you are ready to proceed?" question defaults to no, so you will have to deliberately tell it to proceed and format your partitions. If you chose no, you would simply be dropped into a shell and could start the install again by typing install, or just by rebooting again with your boot disk.

At this point all filesystems will formatted for you. This could take some time depending on the size of the partitions and the speed of the disk.

4.5.3 - Setting the system hostname

Now you must set the system hostname. This value, along with the DNS domain name (specified below), will be saved in the file `/etc/myname`, which is used during normal boot to set the hostname of the system. If you do not set the domain name of the system, the default value of 'my.domain' will be used.

It is important to set this name now, because it will be used when the cryptographic keys for the system are generated during

the first boot after installation. This generation takes place whether the network is configured or not.

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

4.5.4 - Configuring the network

Now it is time to configure your network. The network must be configured if you are planning on doing a ftp or nfs based install, considering it will be based upon the information you are about to enter. Here is a walk through of the network configuration section of the install process.

```
Configure the network? [y] 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? [n] 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? [y] Enter
Default route? (IP address, 'dhcp' or 'none') 199.185.137.128
add net default: gateway 199.185.137.128
Edit hosts with ed? [n] Enter
Do you want to do any manual network configuration? [n] Enter
```

In the above example, we use a static IP address. As indicated, you can use "`dhcp`" instead on most platforms (not Alpha), assuming your environment supports it. In the case of DHCP, most of the information will be grabbed from the remote DHCP server; you will be given a chance to confirm it. Here is a sample of the network configuration part of the install, this time done with DHCP:

```
Configure the network? [y] 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? [n] Enter
IP address for fxp0? (or 'dhcp') dhcp
Issuing hostname-associated DHCP request for fxp0.
Internet Software Consortium DHCP Client 2.0pl5-OpenBSD
Listening on BPF/fxp0/00:08:c7:77:b4:6b
Sending on   BPF/fxp0/00:08:c7:77:b4:6b
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? [y] Enter
Default route? (IP address, 'dhcp' or 'none') [199.185.137.128] Enter
add net default: gateway 199.185.137.128
Edit hosts with ed? [n] Enter
Do you want to do any manual network configuration? [n] Enter
```

NOTE: Only **one** interface can easily be configured using DHCP during an install. If you attempt to configure more than one interface using DHCP you will encounter errors. You have to manually configure the additional interfaces after the

installation.

Now, we set the password for the root account:

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

Use a secure password for the root account. You will create other user accounts after the system is booted. From 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 - Choosing installation media

After your network is set up, the install script will give you a chance to make manual adjustments to the configuration. Then the filesystems you created will be mounted and a root password set. This will get your local disks ready for the OpenBSD packages to be installed upon them.

Next, you will get a chance to choose your installation media. The options are listed below.

```
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 this example we are installing from CD-ROM. This will bring up a list of devices on your computer identified as a CD-ROM. Most people will only have one. If you need to, make sure you pick the device which you will use to install OpenBSD from.

NOTE: All possible sources for install sets are listed, but not all may be available on your system. e.g. (n)fs is shown but not all architectures allow NFS installations. If you choose a source that is not available, you will get an error message and be given the chance to choose another source for your installation sets.

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

Pathname to the sets?  (or 'done') [3.4/i386] Enter
```

Here, you are prompted for which directory the installation files are, which is `3.4/i386/` on the official CD-ROM.

4.5.6 - Choosing filesets.

Now it's time to choose which packages you will be installing. You can get a description of these files in the next section. The files that the install program finds will be shown to you on the screen. Your job is just to specify which files you want. By default all the non-X packages are selected; however, some people may wish to limit this to the bare minimum required to run OpenBSD, which would be `base34.tgz`, `etc34.tgz` and `bsd`. Others will wish to install all packages. The example below is that of a full install.

```
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
        [ ] bsd.rd
        [X] base34.tgz
        [X] etc34.tgz
        [X] misc34.tgz
        [X] comp34.tgz
        [X] man34.tgz
        [X] game34.tgz
        [ ] xbase34.tgz
        [ ] xshare34.tgz
        [ ] xfont34.tgz
        [ ] xserv34.tgz

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
        [X] bsd.rd
        [X] base34.tgz
        [X] etc34.tgz
        [X] misc34.tgz
        [X] comp34.tgz
        [X] man34.tgz
        [X] game34.tgz
        [X] xbase34.tgz
        [X] xshare34.tgz
        [X] xfont34.tgz
        [X] xserv34.tgz
```

You can do all kinds of nifty things here -- `-x*` would remove all X components, if you changed your mind. In this case, we are going to load all the sets. While the system will run with fewer sets, either the starting default or installing all sets is recommended. More details on selecting sets here.

Once you have successfully picked which packages you want, you will be prompted to make sure you want to extract these packages and they will then be installed. A progress bar will be shown that will keep you informed on how much time it will take. The times range greatly depending on what system it is you are installing OpenBSD on, the packages installed, and the speed of the source media. This part may from a few minutes to several hours.

```
File Name? (or 'done') [done] Enter
Ready to install sets? [y] Enter
Getting bsd ...
100% |***************************************************|  4735 KB    00:03
Getting bsd.rd ...
100% |***************************************************|  4275 KB    00:02
Getting base34.tgz ...
100% |***************************************************| 30267 KB    00:21
Getting etc34.tgz ...
100% |***************************************************|  1545 KB    00:01
Getting misc34.tgz ...
100% |***************************************************|  1909 KB    00:01
Getting comp34.tgz ...
100% |***************************************************| 17074 KB    00:13
Getting man34.tgz ...
100% |***************************************************|  6139 KB    00:04
Getting game34.tgz ...
100% |***************************************************|  2534 KB    00:01
```

```
Getting xbase34.tgz ...
100% |***********************************************| 10940 KB    00:06
Getting xshare34.tgz ...
100% |***********************************************|  1656 KB    00:02
Getting xfont34.tgz ...
100% |***********************************************| 31160 KB    00:21
Getting xserv34.tgz ...
100% |***********************************************| 15228 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')
```

At this point, you can pull additional files from other sources if desired, or hit '`done`' if you have installed all the file sets you need.

4.5.7 - Finishing up

You will now be asked if you plan to run X on this system. If you answer 'Y', `/etc/sysctl.conf` will be modified to include the line `machdep.allowaperture=1` or `machdep.allowaperture=2`, depending on your platform.

Your last task is to enter the time zone. Depending on where your machine lives, there are may be several equally valid answers for the question. In the example that follows, we used `US/Eastern`, but could also have used `EST5EDT` or `US/Michigan` and had the same result. Hitting `?` at the prompts will guide you through your choices.

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

If you are concerned about very precise time, you may wish to read this.

The last steps are for the system to create the `/dev` directory (which may take a while on some systems, especially if you have a small amount of RAM), and install the boot blocks.

```
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
room for 12 filesystem blocks at 0x16f
Will load 7 blocks of size 8192 each.
Using disk geometry of 63 sectors and 240 heads.
 0:  9 @(203 150 55) (3078864-3078872)
 1: 63 @(203 151 1) (3078873-3078935)
 2: 24 @(203 152 1) (3078936-3078959)
```

```
 3: 16 @(203 8 47) (3069910-3069925)
/mnt/boot: 4 entries total
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 is now installed on your system and ready for its first boot, but before you do...

Before you reboot

At this point, your system is installed and ready to be rebooted and configured for service. Before doing this, however, it would be wise to check out the Errata page to see if there are any bugs that would immediately impact you.

After you reboot

One of your first things to read after you install your system is afterboot(8).

You may also find the following links useful:

- Adding users in OpenBSD
- Initial Network Setup
- Man Pages of popular/useful commands
- OpenBSD man pages on the Web
- The OpenBSD Ports and Packages system for installing software, as well as here and here

One last thing...

The OpenBSD developers ask you to Send in a copy of your dmesg. This is really appreciated by the developers, and ultimately, all users.

4.6 - What files are needed for installation?

The complete OpenBSD installation is broken up into a number of separate *file sets*. Not every application requires every file set. Here is an overview of each:

- ***bsd*** - This is the Kernel. **Required**
- ***bsd.rd*** - RAM disk kernel
- ***base34.tgz*** - Contains the base OpenBSD system **Required**
- ***etc34.tgz*** - Contains all the files in /etc **Required**
- ***comp34.tgz*** - Contains the compiler and its tools, libs. **Recommended**
- ***man34.tgz*** - Contains man pages **Recommended**
- ***misc34.tgz*** - Contains misc info, setup documentation
- ***game34.tgz*** - Contains the games for OpenBSD
- ***xbase34.tgz*** - Contains the base install for X11
- ***xfont34.tgz*** - Contains X11's font server and fonts
- ***xserv34.tgz*** - Contains X11's X servers
- ***xshare34.tgz*** - Contains manpages, locale settings, includes, etc for X

4.7 - How much space do I need for an OpenBSD installation?

The following are minimum suggested filesystem sizes for a full system install. The numbers include enough extra space to

permit you to run a typical home system that is connected to the Internet.

- These are minimum values.
- If you plan to install a significant amount of third party software, make your `/usr` partition large! **At least** triple these values!
- For a system that handles lots of email or web pages (stored, respectively, in `/var/mail` and `/var/www`) you will want to make your `/var` partition significantly larger, or put them on separate partitions.
- For a multiuser system which may generate lots of logs, you will still want to make your `/var` partition significantly larger (`/var/log`).
- If you plan to rebuild the kernel or system from source, you will want to make the `/usr` partition significantly larger, **at least** 800M-1G larger than indicated below.

As you read this, keep in mind that `/usr` and `/usr/X11R6` are usually both parts of the same filesystem, that is, `/usr`, as there is no big advantage to making them into separate filesystems.

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	120M

In addition, it is recommended that a `/tmp` partition be used. The `/tmp` partition is used in the compiling of ports, among other things, so how big you make it depends on what you do with it. 50M may be plenty for most people, but some large applications may require 100M or more of `/tmp` space.

When you are in the disklabel editor, you may choose to make your entire system have just an 'a' (main filesystem) and 'b' (swap) . The 'a' filesystem which you set up in disklabel will become your root partition, which should be the sum of all the 3 main values above (`/`, `/usr`, and `/var`) plus some space for `/tmp`. The 'b' partition you set up automatically becomes your system swap partition -- we recommend a minimum of 32MB but if you have disk to spare make it at least 64MB. If you have lots of disk space to spare, make this 256MB, or even 512MB.

Swap space is used to store system core dumps on in the event of a crash(8). If this is a consideration for you, your swap space should be slightly larger than the amount of main memory you are likely to ever have in the system. Note that upon reboot, savecore(8) will attempt to save the contents of the swap partition to a file in `/var/crash` so again, if this is a priority for you, your `/var` partition must have enough *free space* to hold these dump files.

There are five main reasons for using separate filesystems, instead of shoving everything into one or two filesystems:

- **Security**: You can mark some filesystems as 'nosuid', 'nodev', 'noexec', 'readonly', etc. This is now done by the install process, in fact, if you use the above described partitions.
- **Stability**: A user, or a misbehaved program, can fill a filesystem with garbage if they have write permissions for it. Your critical programs, which of course run on a different filesystem, do not get interrupted.
- **Speed**: A filesystem which gets written to frequently may get somewhat fragmented. (Luckily, the ffs filesystem, what OpenBSD uses, is not prone to heavy fragmentation.)
- **Integrity**: If one filesystem is corrupted for some reason then your other filesystems are still OK.
- **Size**: Many platforms have limits on the area of a disk where the boot ROM can load the kernel from. In some cases, this limit may be very small (504M for an older 486), in other cases, a much larger limit (8G on new i386 systems). As the kernel can end up anywhere in the root partition, the entire root partition should be within this area. For more details, see this section. A good guideline might be to keep your / partition completely below 2G, unless you know your platform (and particular machine!) can handle more (or less!) than that.

Some additional thoughts on partitioning:

- For your first attempt at an experimentation system, one big `/` partition and swap may be easiest until you know how much space you need. By doing this you will be sacrificing some of the default security features of OpenBSD that require separate filesystems for `/, /tmp, /var, /usr` and `/home`.
- A system exposed to the Internet or other hostile forces should have a separate `/var` (and maybe even a separate `/var/log`) for logging.

- A `/home` partition can be nice. New version of the OS? Wipe and reload everything else, leave your `/home` partition untouched. Remember to save a copy of your configuration files, though!
- A separate partition for anything which may accumulate a large quantity of files that may need to be deleted can be faster to reformat and recreate than to delete. See the upgrade-minifaq for an example (`/usr/obj`).
- If you wish to rebuild your system from source for any reason, the source will be in `/usr/src`. If you don't make a separate partition for `/usr/src`, make sure `/usr` has sufficient space.
- A commonly forgotten fact: you do **not** have to allocate all space on a drive when you set the system up! Since you will now find it a challenge to buy a new drive smaller than 20G, it can make sense to leave a chunk of your drive unallocated. If you outgrow a partition, you can allocate a new partition from your unused space, duplicate your existing partition to the new partition, change /etc/fstab to point to the new partition, remount, you now have more space.
- If you make your partitions too close to the minimum size required, you will probably regret it later, when it is time to upgrade your system.
- If you permit users to write to `/var/www` (i.e., personal web pages), you might wish to put it on a separate partition, so you can use quotas to restrict the space they use, and if they fill the partition, no other parts of your system will be impacted.

4.8 - Multibooting OpenBSD/i386

Multibooting is having several operating systems on one computer, and some means of selecting the which OS is to boot. It is *not* a trivial task! If you don't understand what you are doing, you may end up deleting large amounts of data from your computer. New OpenBSD users are *highly* encouraged to start with a blank hard drive on a dedicated machine, and then practice your desired configuration on a non-production system before attempting a multiboot configuration on a production machine. FAQ 14 has more information about the OpenBSD boot process.

When multibooting, the requirements of all operating systems must be met by your configuration. People often ask if there is a way around the 8G boot limit of OpenBSD. While there are some programs that claim to get around various limits of various operating systems, none of them are known to do this with current versions of OpenBSD.

Here are several options to multibooting:

Setting active partitions

This is probably the most overlooked, and yet, sometimes the best solution for multibooting. Simply set the active partition in whatever OS you are currently using to be the one you want to boot by default when you next boot. Virtually every OS offers a program to do this; OpenBSD's is fdisk(8), similar named programs are in Windows 9x and DOS, and many other operating systems. This can be highly desirable for OSs or systems which take a long time to shut down and reboot -- you can set it and start the reboot process, then walk away, grab a cup of coffee, and come back to the system booted the way you want it -- no waiting for the Magic Moment to select the next OS.

Boot floppy

If you have a system that is used to boot OpenBSD infrequently (or don't wish other users of the computer to note anything has changed), consider using a boot floppy. Simply use one of the standard OpenBSD install floppies, and create a `/etc/boot.conf` file (yes, you will also have to create an `/etc` directory on the floppy) with the contents:

```
boot hd0a:/bsd
```

to cause the system to boot from hard drive 0, OpenBSD partition 'a', kernel file `/bsd`. Note you can also boot from other drives with a line like: "`boot hd2a:/bsd`" to boot off the third hard drive on your system. To boot from OpenBSD, slip your floppy in, reboot. To boot from the other OS, eject the floppy, reboot.

In this case, the boot(8) program is loaded from the floppy, looks for and reads `/etc/boot.conf`. The "`boot hd0a:/bsd`" line instructs boot(8) where to load the kernel from -- in this case, the first HD the BIOS sees. Keep in mind, only a small file (`/boot`) is loaded from the floppy -- the system loads the entire kernel off the hard disk, so this only adds about five seconds to the boot process.

Windows NT/2000/XP NTLDR

To multiboot OpenBSD and Windows NT/2000/XP, you can use NTLDR, the boot loader that NT uses. To multi-boot with NT, you need a copy of your OpenBSD Partition Boot Record (PBR). After running installboot, you can copy it to a file using dd(1):

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

Now boot NT and put `openbsd.pbr` in C:. Add a line like this to the end of `C:\BOOT.INI`:

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

When you reboot, you should be able to select OpenBSD from the NT loader menu. There is much more information available about NTLDR at the NTLDR Hacking Guide.

On Windows XP you can also edit the boot information using the GUI; see the XP Boot.ini HOWTO.

Note: The Windows NT boot loader is only capable of booting OSs from the primary hard drive. You can not use it to load OpenBSD from the second drive on a system.

Other boot loaders

Some other bootloaders OpenBSD users have used successfully include GAG, OSBS, The Ranish Partition Manager and GRUB.

OpenBSD and Linux (i386)

Please refer to INSTALL.linux, which gives in depth instructions on getting OpenBSD working with Linux.

4.9 - Sending your dmesg to dmesg@openbsd.org after the install

Just to remind people, it's important for the OpenBSD developers to keep track of what hardware works, and what hardware doesn't work perfectly.

A quote from /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!
```

Make sure you send email from an account that is able to also receive email so developers can contact you if they have something they want you to test or change in order to get your setup working. It's not important at all to send the email from the same machine that is running OpenBSD, so if that machine is unable to receive email, just

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

and then forward that message to

```
 dmesg@openbsd.org
```

where `your-account@yourmail.dom` is your regular email account. (or transfer the dmesg output using ftp/scp/floppydisk/carrier-pigeon/...)

NOTE - Please send only GENERIC kernel dmesgs. Custom kernels that have device drivers removed are not helpful.

4.10 - Adding a file set after install

"Oh no! I forgot to add a file set when I did the install!"

Sometimes, you realize you really DID need `comp34.tgz` (or any other system component) after all, but you didn't realize this at the time you installed your system. Good news: There are two easy ways to add file sets after the initial install:

Using the upgrade process

Simply boot your install media (CD-ROM or Floppy), and choose Upgrade (rather than Install). When you get to the lists of file sets to install, choose the sets you neglected to install first time around, select your source, and let it install them for you.

Using tar(1)

The install file sets are simply compressed tar files, and you can expand them manually from the root of the filesystem:

```
# cd /
# tar xzvpf comp34.tgz
```

Do NOT forget the `'p'` option in the above command in order to restore the file permissions properly!

One common mistake is to think you can use pkg_add(1) to add a missing file sets. This does not work. pkg_add(1) is for package files, not generic tar files like the install sets.

4.11 - What is 'bsd.rd'?

`bsd.rd` is a "RAM Disk" kernel. This file can be very useful; many developers are careful to keep it on the root of their system at all times.

Calling it a "RAM Disk kernel" describes the root filesystem of the kernel -- rather than being a physical drive, the utilities available after the boot of `bsd.rd` are stored in the kernel, and are run from a RAM-based filesystem. `bsd.rd` also includes a healthy set of utilities to allow you to do system maintenance and installation.

On some platforms, `bsd.rd` is actually the preferred installation technique -- you place this kernel on an existing filesystem, boot it, and run the install from it. On most platforms, if you have a running older version of OpenBSD, you can FTP a new version of `bsd.rd`, reboot from it, and install a new version of OpenBSD without using any removable media at all.

Here is an example of booting `bsd.rd` on an 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 . . .
```

As indicated, you will be brought to the install program, but you can also drop to the shell to do maintenance on your system.

The general rule on booting `bsd.rd` is to change your boot kernel from `/bsd` to `bsd.rd` through whatever means used on your platform.

4.12 - Common installation problems

4.12.1 - My Compaq only recognizes 16M RAM

Some Compaq systems have an issue where the full system RAM is not detected by the OpenBSD second stage boot loader properly, and only 16M may be detected and used by OpenBSD. This can be corrected either by creating/editing `/etc/boot.conf` file, or by entering commands at the "`boot>`" prompt before OpenBSD loads. If you had a machine with 64M RAM, but OpenBSD was only detecting the first 16M, the command you would use would be:

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

to add 48M (`0x3000000`) after the first 16M (`0x1000000`). Typically, if you had a machine with this problem, you would enter the above command first at the install floppy/CD-ROM's `boot>` prompt, load the system, reboot, and create a `/etc/boot.conf` file with the above line in it so all future bootings will recognize all available RAM.

It has also been reported that a ROM update will fix this on *some* systems.

4.12.2 - My i386 won't boot after install

Your install seemed to go fine, but on first boot, you see no sign of OpenBSD attempting to boot. There are a few common reasons for this problem:

- **No partition was flagged active in fdisk(8).** To fix this, reboot the machine using the boot floppy or media, and "flag" a partition as "active" (bootable). See here and here
- **No valid boot loader was ever put on the disk.** If you answer "Y" to the "Use entire disk for OpenBSD?" question during the install, or use the "`reinit`" or "`update`" options of fdisk(8), the OpenBSD boot record is installed on the Master Boot Record of the disk; otherwise, the existing master boot code is untouched. This will be a problem if no other boot record existed. The solution is to boot the install media again, drop to the shell and invoke fdisk(8) and use the "`update`" option.
- **In some rare occasions, something may go wrong with the second stage boot loader install.** Reinstalling the second stage boot loader is discussed here.
- **You mixed *a.out* and *ELF* boot disk and install files.** OpenBSD/i386 has transitioned from the older *a.out* format to the *ELF* format for binaries shortly after 3.3-release. The OpenBSD 3.4 boot loader can not boot an OpenBSD 3.3-release's kernel, nor can the 3.3-release boot loader boot an OpenBSD 3.4 kernel. You must use a boot disk that matches the version of OpenBSD you intend to install. Failure to follow this will show itself with a message such as "`failure(79)`" and/or "`Inappropriate file type or format`".
- **You ignored all the warnings and explanations about the 8G boot limit.** This results in a hang at the very start of the boot process, sometimes with the message, "`Bad magic`".

4.12.3 - My (older, slower) machine booted, but hung at the ssh-keygen steps

It is very likely your machine is running fine, just taking a while to do the ssh key generation process. A SparcStation2 or a Macintosh Quadra may take 45 minutes or more to complete the three ssh-keygen(1) steps, some machines will take even longer. Just let it finish; it is only done once per install.

4.12.4 - I got the message "Failed to change directory" when doing an install

When doing an FTP install of a snapshot during the *-beta* stage of the OpenBSD development cycle, you may see this:

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

This is normal and expected behavior during this pre-release part of the cycle. The install program looks for the FTP list on the primary FTP server in a directory that won't be available until the release date, so you get the above message.

Simply use the FTP mirror list to find your favorite FTP mirror, and manually enter its name when prompted.

Note: You should not see this if you are installing *-release* or from CD-ROM.

4.12.5 - When I login, I get "login_krb4-or-pwd: Exec format error"

Kerberos IV has been removed from OpenBSD 3.4, but if you did an upgrade, the old Kerberos IV binaries still will be on your system. This is a problem on the i386 platform, as the old Kerberos files are in a.out, format and thus unable to run on the standard ELF kernel (which has a.out emulations disabled, as mentioned here). If you have encountered this problem, you need to override the krb4 authentication method when you log in:

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

login: joeuser:passwd
password:
```

You can use the same "*username*`:passwd`" syntax with an ssh connection and with su(1) to access your system. Now edit `/etc/login.conf`, and remove the krb4 references.

4.13 - Customizing the install process

`siteXX.tgz` file

The OpenBSD install/upgrade scripts allow the selection of a user-created set called "`siteXX.tgz`", where XX is the release version (e.g. 34). The `siteXX.tgz` file set is, like the other file sets, a gzip(1) compressed tar(1) archive rooted in '/' and is un-tarred like the other sets with the options `xzpf`. This set will be installed last, after all other file sets.

This file set allows the user to add to and/or override the files installed in the 'normal' sets and thus customize the installation or upgrade.

Some example uses of a `siteXX.tgz` file:

- Create a siteXX.tgz file that contains all the file changes you made since first installing OpenBSD. Then, if you have to re-create the system you simply select siteXX.tgz during the re-install and all of your changes are replicated on the new system.
- Create a series of machine specific directories that each contain a siteXX.tgz file that contains files specific to those machine types. Installation of machines (e.g. boxes with different graphics cards) of a particular category can be completed by selecting the appropriate siteXX.tgz file.
- Put the files you routinely customize in a same or similar way in a siteXX.tgz file -- `/etc/skel` files, `/etc/pf.conf`, `/var/www/conf/httpd.conf`, `/etc/rc.conf`, etc.

`install.site/upgrade.site` scripts

As the last step in the install/upgrade process, the scripts look in the root directory of the newly installed/upgraded system for `install.site` or `upgrade.site`, as appropriate to the current process, and runs this script in an environment chrooted to the installed/upgraded system's root. Remember, the upgrade is done from a booted file system, so your target file system is actually mounted on `/mnt`. However, your script can be written as if it it is running in the "normal" root of your file system. Since this script is run after all the files are installed, you have almost full functionality of your system (though, in single user mode) when your script runs.

Note that the `install.site` script would have to be in a `siteXX.tgz file`, while the `upgrade.site` script could be put in the root directory before the upgrade, or could be put in a `siteXX.tgz` file.

The scripts can be used to do anything possible in a script.

- Remove files that are installed/upgraded that you don't want present on the system.
- Remove/upgrade/install the packages you want on the installed system.
- Do an immediate backup/archive of the new system before you expose it to the rest of the world.

The combination of `siteXX.tgz` and `install.site/upgrade.site` files is intended to give the user broad customization capabilities without having to build their own custom install sets.

4.14 - How can I install a number of similar systems?

Here are some tools you can use when you have to deploy a number of similar OpenBSD systems.

`siteXX.tgz` and `install/upgrade.site` files

See the above article.

Restore from dump(8)

On most platforms, the boot media includes the restore(8) program, which can be used to restore a backup made by dump(8). Thus, you could boot from a floppy, CD, or bsd.rd file, then fdisk, disklabel, and restore the desired configuration from tape or other media, and install the boot blocks. More details here.

Disk imaging

Unfortunately, there are no known disk imaging packages which are FFS-aware and can make an image containing only the active file space. Most of the major disk imaging solutions will treat an OpenBSD partition as a "generic" partition, and can make an image of the whole disk. This often accomplishes your goal, but usually with huge amounts of wasted space -- an empty, 10G `/home` partition will require 10G of space in the image, even if there isn't a single file in it. While you can typically install a drive image to a larger drive, you would not be able to directly use the extra space, and you would not be

able to install an image to a smaller drive.

If this is an acceptable situation, you may find the dd command will do what you need, allowing you to copy one disk to another, sector-for-sector. This would provide the same functionality as commercial programs without the cost.

4.15 - How can I get a dmesg(8) to report an install problem?

When reporting a problem, it is critical to include the complete system dmesg(8). However, often when you need to do this, it is because the system is working improperly or won't install so you may not have disk, network, or other resources you need to get the dmesg to the appropriate mail list. There are other ways, however:

- **Floppy disk:** The boot disks and CD-ROM has enough tools to let you record your dmesg to an MSDOS floppy disk for reading on another machine. Place an MSDOS formatted floppy in your disk drive and execute the following commands:

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

  If you have another OpenBSD system, you can also write it to an OpenBSD compatible floppy -- often, the boot floppy has enough room on it to hold the dmesg. In that case, leave off the "`-t msdos`" above.
- **Serial Console:** See this article on setting up the serial console, then capture the output to a file.
- **FTP:** Under some circumstances, you may be able to use the ftp(1) client on the boot disk or CD-ROM to send the dmesg to a local FTP server, where you can retrieve it later.

4.16 - Upgrading/reinstalling OpenBSD/i386 using `bsd.rd-a.out`

It is normally possible to perform upgrades and installs using the bsd.rd kernel. However, with OpenBSD 3.4, the i386 platform switched executable format from a.out to ELF, so older boot loaders (OpenBSD 3.3 and before) are unable to run the new-format `bsd.rd` kernel.

To circumvent this problem, and allow upgrades to be performed using `bsd.rd`, an a.out version of `bsd.rd` has been made available on the FTP distribution. This file, `bsd.rd-a.out`, can be booted by OpenBSD 3.3 and below, but is a genuine OpenBSD 3.4 kernel, including the new ELF boot loader, so can be used to bootstrap OpenBSD/i386 3.4 from an older system.

Simply download `bsd.rd-a.out` and place it in your machine's root directory. Boot it instead of the normal `bsd` or `bsd.rd` kernels as shown here (specifying `bsd.rd-a.out` as your boot kernel, of course).

If you wish to install *-current*, it is recommended you first install a minimal 3.4-release system (`base34.tgz, etc34.tgz, bsd`), then reinstall using the *-snapshot* `bsd.rd` file.

[FAQ Index] [To Section 3 - Obtaining OpenBSD] [To Section 5 - Building the System from Source]

---

www@openbsd.org

$OpenBSD: faq4.html,v 1.163 2004/02/10 03:28:27 nick Exp $

5 - Neuerzeugen des Systems aus den Quelltexten

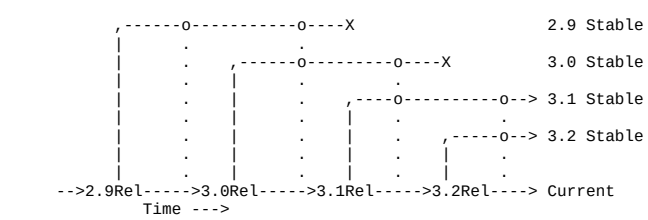
Inhaltsverzeichnis

5.1 - OpenBSD's 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 verwandelt.

Grafisch sieht die Entwicklung dieser flavors ungefähr so aus:



Der code tree wird alle sechs Monate in *-current*, *-release* und *-stable* aufgeteilt -- *-release* wird zu einem festen Punkt (einem "Tag") in der Geschichte des source tree - der niemals geändert, und genau der ist auch auf den CDs und FTP Servern.
-Current dagegen ist die Baustelle, an der die aktive Arbeit gemacht wird, und die dann der nächste *-release* von OpenBSD wird.

Der *-stable* branch (auch bekannt als der "Patch branch") basiert auf *-release*, und ist, wie man sehen kann, ein "branch" aus dem Entwicklungspfad von OpenBSD. Wenn wichtige Korrekturen und Fixes an *-current* vorgenommen werden, werden diese in die *-stable* branches "zurückportiert". In der obigen Illustration sind die vertikalen gepunkteten Linien Bug Fixes, die in die *-stable* branches eingefügt werden. Du wirst auch erkennen, dass der Lebenszyklus des *2.9-stable* branch mit Erscheinen des *3.1-release* beendet war und der *3.0-stable* branch mit Erscheinen von *3.2-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 Features. Der *-stable* branch kann vom Design her sehr leicht aus dem *-release* branch der selben Version erzeugt werden (z.B. von *3.2-release* zu *3.2-stable*).

Der *-stable* branch ist *-release* plus patches, die man auf der errata Seite findet, und ein paar einfache fixes, die keinen errata-Entrag verdienen. Normalerweise ist die Bedienung und der Betrieb von *-stable* genauso wie der des *-release*, auf dem er basiert. Wenn die man pages 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 Features werden nur dann eingefügt, wenn das als sehr wichtig erachtet wird.

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. Wie vorher schon gesagt, gibt es keinerlei Garantie, dass das System kompiliert oder gar funktioniert (natürlich hoffen wir das immer). Es ist absolut möglich und nicht ungewöhnlich, dass man sich den *-current* Source holt, und er nicht kompiliert werden kann, während es fünf Minuten später wieder klappt. **Wenn du damit nicht umgehen kannst, lass deine Finger von *-current*.**

Die meisten User sollten also einfach *-stable* oder *-release* benutzen. Wo 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 Features 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 kein nützlicher bug report. "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, wo auch ein "normaler" User 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 ein neues Feature 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 Sites verfügbar gemacht. Wie der Name schon sagt, sind das Versionen von Code, der gerade zu eben diesem Zeitpunkt aktuell war, als der Übersetzer des snapshot sich eine Kopie des Source Code für eben diese Plattform gemacht hat. Denke bitte daran, dass das 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 sind, oder auch nur installationsfähig. 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 ein wunderbarer Ausgangspunkt, bevor du versuchst *-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 könnte man z.B. mehrere Snapshots an einem Tag erzeugen. Auf den langsameren kann das schon mal eine ganze Woche dauern. Und das Platzieren eines "Tag" für jeden Snapshot in den Source Code ist nun wirklich extrem unpraktisch.

Die Dinge in Balance 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, wenn man ein Upgrade mittels Source durchführt, **nur in eine Richtung unterstützt wird: Von älter zu neuer**, und von *-stable* zu *-current*. Du kannst kein *3.2-current* (oder einen Snapshot) laufen lassen, und dich dann entscheiden, dass dir das zu gefährlich ist, und einfach nach *3.2-stable* zurückgehen. Du bist auf dich allein gestellt, wenn du einen anderen Weg wählst als den normalen und unterstützen, nämlich dein System von Grund auf neu zu installieren, erwarte bitte keinerlei Hilfeleistung vom OpenBSD Entwicklerteam.

Ja, das soll wirklich heissen, dass du lang und breit darüber nachdenken sollst, bevor du dich an *-current* wagst.

5.2 - Warum brauche ich einen selber kompilierten Kernel?

Es gibt dafür mehrere Gründe, obwohl das Kompilieren eines eigenen Kernels eher für erfahrenere Benutzer geeignet ist, die bereits ein gutes Allgemeinverständnis vom System besitzen.

- Dein Computer hat sehr wenig Hauptspeicher, den du so gut wie möglich ausnutzen willst, indem du nicht benötigte Gerätetreiber entfernst.
- Du möchtest Standardoptionen entfernen oder Optionen hinzufügen, die nicht standardmäßig aktiviert sind.
- Du möchtest experimentelle Optionen hinzufügen.

In den meisten Fällen wirst du keinen eigenen Kernel kompilieren müssen. Ein GENERIC Kernel wird normalerweise alles beinhalten, das du brauchst. Es gibt sogar mehrere Gründe, warum du keinen eigenen Kernel kreieren willst. Der Hauptgrund ist, daß man zwar logisch aussehende Änderungen an der Kernelkonfiguration auf einfache Weise durchführen kann, aber dein Kernel funktioniert dann nicht. Dies soll ein Warnsignal sein. Wenn etwas nicht zu funktionieren scheint, dann probiere bitte vorher einen GENERIC Kernel zu starten, bevor du einen Fehlerbericht einschickst. Die Entwickler werden im Normalfall jeden Fehlerbericht ignorieren, der nicht mit einem GENRIC Kernel erzeugt und getestet wurde, es sei denn das Problem kann ebenfalls mit einem GENERIC Kernel reproduziert werden. Nimm das als Warnung.

5.3 - Kernelkonfigurationsoptionen

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 von Möglichkeiten, deinen Kernel auf deine Wünsche abzustimmen. Hier werden wir nur auf einige der am häufigsten benutzten Möglichkeiten eingehen. Siehe options(4) für eine komplette Liste der Optionen. Es stehen auch Beispielkonfigurationsdateien für deine Architektur bereit.

Nicht alle Kerneloptionen sind auf Kompatibilität mit allen anderen Optionen getestet worden. Daher solltest du keine Option in deinem Kernel setzen, wenn du nicht einen triftigen Grund dafür hast! Am meisten wird der GENERIC Kernel getestet. Dies ist normalerweise die Kombination der Optionen in /usr/src/sys/arch/<deine Arch>/conf/GENERIC und /usr/src/sys/conf/GENERIC.

- Alpha Kernel Konfgurationsdateien

- i386 Kernel Konfigurationsdateien
- macppc Kernel Konfigurationsdateien
- sparc Kernel Konfigurationsdateien
- sparc64 Kernel Konfigurationsdateien
- vax Kernel Konfigurationsdateien
- Andere Architekturen

Wenn du diese Dateien genauer betrachtest, dann wird dir eine Zeile wie diese auffallen:
include "../../../conf/GENERIC"
Dies bedeutet, daß ein Verweis auf eine andere Konfigurationsdatei vorhanden ist. Diese Datei beinhaltet architekturunabhängige Optionen. Wenn du also deinen eigenen Kernel konfigurieren willst, dann mußt du auch /sys/conf/GENERIC beachten. Dort **sind** Optionen enthalten, die **benötigt** werden.

Alle nachfolgend aufgelisteten Optionen sollten in deiner Kernelkonfigurationsdatei im Format von:
option OPTION
angeführt sein. Um die Option debug im Kernel zu plazieren, füge eine Zeile wie diese ein:
option DEBUG
Die Optionen im OpenBSD Kernel werden in Compilerpräprozessoroptionen übersetzt, daher würde eine Option wie DEBUG den Quelltext mit der Option -DDEBUG kompilieren, was einem #define DEBUG im Kernel entspricht.

OpenBSD hat viele Kompatibilitätsmöglichkeiten, die dir erlauben, Binärdateien von anderen Betriebsystemen auszuführen. Nicht alle Optionen sind auf jeder Architektur vorhanden, daher lies die entsprechende Manualseite, ob die Architektur unterstützt wird.

- COMPAT_SVR4(8) - Kompatibilität mit SVR4 Binärdateien.
- COMPAT_BSDOS(8) - Kompatibilität mit BSD/OS Binärdateien.
- COMPAT_LINUX(8) - Kompatibilität mit Linux Binärdateien.
- COMPAT_SUNOS(8) - Kompatibilität mit SunOS Binärdateien.
- COMPAT_ULTRIX(8) - Kompatibilität mit Ultrix Binärdateien.
- COMPAT_FREEBSD(8) - Kompatibilität mit FreeBSD Binärdateien.
- COMPAT_HPUX(8) - Kompatibilität mit HP-UX Binärdateien. Nur auf einigen m68k Architekturen verfügbar.
- COMPAT_IBCS2(8) - Kompatibilität mit ibcs2 Binärdateien.
- COMPAT_OSF1(8) - Digital Unix Binärdateien laufen lassen. Nur auf der Alpha Plattform verfügbar.
- COMPAT_43 - Kompatibilität mit 4.3BSD. Von dessen Gebrauch wird zwar abgeraten, aber wird für einige Applikationen benötigt.
- COMPAT_11 - Kompatibilität mit NetBSD 1.1.
- COMPAT_NOMID - Kompatibilität mit a.out executables ohne machine id.

Es ist immer hilfreich, Probleme mit dem Kernel debuggen zu können. Aber viele wollen diese Optionen nicht in ihren Kernel setzen, weil diese Optionen den Kernel sehr vergrößern. Sie sind aber extrem hilfreich im Falle eines Fehlers. Ein Entwickler wird die Quelle deiner Probleme viel schneller finden können. Hier eine Liste der debug Optionen:

- DDB(4) - Kernelinterner Debugger. Nicht auf allen Plattformen. Daher konsultiere die Manualseite.
- KGDB(7) - Kompiliert einen `remote kernel debugger` mittels `remote target` Option von gdb.
- makeoptions DEBUG="-g" - Erzeugt bsd.gdb gemeinsam mit bsd. Dies ist nützlich, um crash dumps mit gdb zu debuggen.
- DEBUG - Diverse debugging Optionen im Kernel, die im Quelltext definiert wurden.
- KTRACE - Fügt "hooks" für "system call tracing facility" ein. Dies erlaubt den Benutzern: ktrace(1).
- DIAGNOSTIC - Fügt Code im Kernel für interne Konsistenzüberprüfungen ein.
- GPROF - Fügt Code im Kernel für Kernelprofilierung mit kgmon(8) ein.
- makeoptions PROF="-pg" - Die -pg Option dient zur Unterstützung von "profiling" im Kernel. Die Option GPROF wird dafür benötigt.

Dateisystemoptionen.

- FFS - Berkeley Fast Filesystem. **ANMERKUNG:** Diese Option wird immer benötigt.
- EXT2FS - Second Extended File System, benötigt für das Lesen von Linux Partitionen.
- MFS - Memory File System, speichern von Dateien in swap - Speicher.
- NFS - Network File System, benötigt für NFS.
- CD9660 - Das iso9660 + rockridge Dateisystem; zum Lesen von CDs.
- MSDOSFS - Benötigt, um MS-DOS FAT Dateisysteme zu lesen; bietet Unterstützung für Windows 95 (lange Dateinamen + Groß-/Kleinschreibungerweiterungen).
- FDESC - Beinhaltet Code für ein Dateisystem, das auf /dev/fd gemountet werden kann.
- KERNFS - Beinhaltet Code für das Mounten eines speziellen Dateisystems (normalerweise auf /kern), in dem sich Dateien befinden, die spezielle Kernelvariablen und -parameter repräsentieren.
- NULLFS - Code für ein loopback Dateisystem. In mount_null(8) hat dazu weitere Informationen
- PROCFS - Beinhaltet Code für ein spezielles Dateisystem (normalerweise auf /proc)
- PORTAL - Beinhaltet das (experimentelle) portal Dateisystem. Dies erlaubt interessante Tricks wie TCP sockets durch das Öffnen von Dateien im Dateisystem zu öffnen.
- UMAPFS - Beinhaltet ein loopback Dateisystem, in dem Benutzer- und Gruppenids umgelegt werden dürfen -- nützlich, wenn man fremde Dateisysteme mit anderen uids und gids als das lokale System mounten will (zB.: NFS).
- UNION - Beinhaltet Code für das union Dateisystem, welches das Mounten von Verzeichnissen übereinander erlaubt, so daß beide Dateisysteme sichtbar bleiben. Dieser Code ist noch nicht sehr stabil.
- XFS - Fügt "hooks" für ein Dateisystem ein, das kompatibel mit dem AFS Dateisystem ist. Derzeit vom Arla/AFS Code benutzt.
- FFS_SOFTUPDATES - Erlaubt den Gebrauch von softupdates. Hier mehr dazu: Softupdates FAQ.
- NFSSERVER - Serverseitiger NFS Code im Kernel.
- NFSCLIENT - Klientseitiger NFS Code im Kernel.
- FIFO - Unterstützung für FIFOs. **Empfohlen**.
- NVNODE=integer - Wobei integer eine Ganzzahl ist, die der Größe des Caches bei name-to-inode Übersetzungsroutinen entspricht (dh. der namei() Cache, obwohl er auch von vielen anderen Funktionen im Kernelquelltext aufgerufen wird).
- EXT2FS_SYSTEM_FLAG - Diese Option ändert das Verhalten der APPEND- und der IMMUTABLE-option für Dateien in einem EXT2FS-Dateisystem. Siehe options(4) für weitere Details.
- QUOTA - Unterstützung für Quoten im Dateisystem. Hier mehr dazu: FAQ 10, Quotas.

Diverse Optionen

- PCIVERBOSE - Mehr Details während des Startprozesses von PCI Geräten.
- EISAVERBOSE - Mehr Details während des Startprozesses von EISA Geräten.
- PCMCIAVERBOSE - Mehr Details während des Startprozesses von PCMCIA Geräten.
- APERTURE - Liefert kernelinterne Unterstützung für VGA Bildzwischenspeicher durch Umlegen von Benutzerprozessen. Benötigt für X.
- LKM - Unterstützung für Loadable Kernel Modules (ladbare Kernelmodule). Nicht auf allen Architekturen. Siehe lkm(4) für weitere Informationen.
- INSECURE - Setzt das Kernelsicherheitsniveau auf -1. Siehe init(8) für weitere Informationen bezüglich der Kernelsicherheitseinstellungen.
- RAM_DISK_HOOKS - Erlaubt den Aufruf von rechnerabhängigen Funktionen, wenn der Ramdisktreiber konfiguriert ist.
- RAM_DISK_IS_ROOT - Zwingt die Ramdisk, root zu sein.
- CCDNBUF=integer - Setzt die Anzahl des "component buffers" von CCD(4). Standard ist 8. Für mehr über CCD siehe CCD(4) oder das Performancetuning FAQ Kapitel.
- KMEMSTATS - Damit unterhält malloc(9) (kernel memory allocator) Statistiken über seinen Gebrauch. Wenn die Option DEBUG benutzt wird, wird diese Option automatisch von config aktiviert.
- BOOT_CONFIG - Fügt die Unterstützung für die -c boot Option ein.

Netzwerkoptionen
Siehe auch das Netzwerk FAQ oder das Networking Performancetuning FAQ.

- GATEWAY - Ermöglich IPFORWARDING und erhöht (auf den meisten Plattformen) die Größe der NMBCLUSTERS.
- NMBCLUSTERS=integer - Steuert die Größe der mbuf cluster map.
- IPFORWARDING - Ermöglich IP routing Verhalten. Wenn diese Option aktiviert ist, wird der Rechner IP Datagramme zwischen seinen Netzwerkkarten weiterleiten, die für andere Rechner bestimmt sind.
- MROUTING - Beinhaltet Unterstützung für IP multicast Router.
- INET - Beinhaltet Unterstützung für den TCP/IP protocol stack. Diese Option wird BENÖTIGT.
- MCLSHIFT=Wert - Diese Option ist der Logarithmus zur Basis 2 der Größe des mbuf clusters. Siehe options(4) für mehr Information über diese Option.
- NS - Inkludiere Unterstützung für den Xerox XNS protocol stack. Siehe ns(4).
- ISO,TPIP - Inkludiere Unterstützung für den allgegenwärtigen OSI protocolstack. Siehe iso(4) für mehr Information.
- EON - Inkludiere Unterstützung für OSI tunneling über IP.
- CCITT,LLC,HDLC - Inkludiere Unterstützung für den X.25 protocol stack.
- IPX, IPXIP - Inkludiere Unterstützung für: Internetwork Packet Exchange protocol, das von Novell NetWare verwendet wird.
- NETATALK - Inkludiere Kernelunterstützung für die AppleTalk Protokolfamilie.
- TCP_COMPAT_42 - Vom Gebrauch dieser Option wird sehr abgeraten, daher sollte sie auch nicht aktiviert werden: TCP Fehlerkompatibilität mit 4.2BSD. In 4.2BSD waren die TCP Sequenzzahlen 32-bit signierte Werte. Moderne Implementationen des TCP verwenden unsignierte Werte.
- TCP_NEWRENO - Aktiviert "NewReno fast recovery phase", welche die Wiederherstellung eines verlorenen Segmentes pro Rücklaufzeit erlaubt.

- TCP_SACK - Aktiviert selektive Bestätigungen.
- TCP_FACK - Aktiviert weiterleitende Bestätigungen, die präzisere Schätzungen über noch ausstehende Daten während der "fast recovery phase" mittels SACK Informationen erlauben. Diese Option kann gemeinsam mit TCP_SACK benutzt werden.
- PPP_FILTER - Diese Option aktiviert auf pcap(3) basierende Filterung für PPP Verbindungen.
- PPP_BSDCOMP - PPP BSD Kompression.
- PPP_DEFLATE - Wird gemeinsam mit PPP_BSDCOMP verwendet.
- IPSEC - Diese Option ermöglicht die Unterstützung des IP security protocol. Siehe ipsec(4) für mehr Details. Dies impliziert nun die Option KEY, welche Unterstützung für PFKEYv2 bietet.
- ENCDEBUG - Diese Option ermöglicht, daß Debuginformationen aufgezeichnet werden, wenn IPSEC Fehler bemerkt.

SCSI Subsystemoptionen

- SCSITERSE - Knappere SCSI Fehlermeldungen. Dies läßt die Tabelle, um ASC/ASCQ Informationen zu dekodieren, aus, was etwa 8 Bytes einspart.
- SCSIDEBUG - Druckt zusätzliche Debuginformationen für das SCSI Subsystem auf die Konsole.

5.4 - Einen eigenen Kernel kompilieren

Vollständige Instruktionen zum Kreieren deines eigenen Kernels findest du hier in afterboot(8).

Um deinen eigenen Kernel von CDROM zu kompilieren, mußt du zunächst den Quelltext haben. Der Source ist sowohl auf der offiziellen CD (Disk 3) als auch auf den FTP Sites zu finden. Das folgende Beispiel nimmt an, dass die CD3 unter /mnt gemountet ist:

```
# cd /usr/src
# tar xvzf /mnt/src.tar.gz
```

Hinweis: Wenn du den source vom FTP Server herunterlädst, wirst du ZWEI Dateien finden, *src.tar.gz* und *srcsys.tar.gz*. Das erste ist das "userland" -- alles bis auf den Kernel, das zweite ist der Kernel Source. Lade beide runter und entpacke sie wie oben beschrieben, in den meisten Fällen willst du sowieso beide haben. Auf der CDROM sind die beiden Dateien zu einer zusammengefasst.

Am einfachsten ist es, mit einem GENERIC Kernel zu beginnen. Dieser befindet sich unter `/usr/src/sys/arch/${arch}/conf/GENERIC`, wobei ${arch} deine Architektur ist. In diesem Verzeichnis befinden sich auch weitere Beispielskonfigurationen. Hier zwei Beispiele fürs Kompilieren deines Kernels. Das erste Beispiel behandelt das Kompilieren von einem Quelltextbaum, der nur Leseberechtigung hat. Das zweite Beispiel zeigt, wie es auf einem Quelltextbaum mit Schreibberechtigung funktioniert.

```
# cd /somewhere
# cp /usr/src/sys/arch/$ARCH/conf/SOMEFILE .
# vi SOMEFILE   (to make the changes you want)
# config -s /usr/src/sys -b . SOMEFILE
```

entweder gefolgt von:

```
    # make depend
        - ODER -
    # make clean
# make
```

Du musst '`make depend`' laufen lassen, sobald du irgendwelche Änderungen an deinem source tree (einschliesslich Updates und Patches) gemacht hast (in anderen Worten nahezu immer, es sei denn du musst '`make clean`' benutzen).

Wenn du Änderungen an deinen Kernel-Konfigurations-Optionen gemacht hast und/oder Änderungen an deinem source tree, solltest du '`make clean`' anstelle des obigen '`made depend`' benutzen. Es ist immer besser 'make clean' zu benutzen, obwohl das in längeren Compilerläufen resultieren kann, da eben mehr erzeugt wird.

Um einen Kernel von einem Quelltextbaum mit Schreibberechtigung zu bauen, tue folgendes:

```
# cd sys/arch/$ARCH/conf
# vi IRGENDEINEDATEI (um deine gewünschten Änderung durchzuführen)
# config IRGENDEINEDATEI (hier mehr darüber :
config(8))
# cd ../compile/IRGENDEINEDATEI
# make
```

Wobei $ARCH für die von dir benutzte Architektur ist (z.B. i386). Du kannst auch ein **make depend** durchführen, um die Abhängigkeiten fürs nächste Kernelkompilieren zu schaffen.

Jetzt muß der Kernel noch an den richtigen Platz.

```
# cp /bsd /bsd.old
# cp /sys/arch/$ARCH/compile/IRGENDEINEDATEI/bsd /bsd
```

Um deinen alten Kernel zu starten, mußt du nur

```
boot> bsd.old
```

beim Starten eingeben, und dein alter Kernel wird anstelle von /bsd geladen.

Manchmal wirst du auch neue Bootblöcke installieren müssen, wenn du einen Kernel gebaut hast. Um dies zu tun, siehe FAQ 14.8, Installieren von Bootblocks. Dies wird dir eine Übersicht vom OpenBSD Bootloaders geben.

5.5 - Boot-Time Konfiguration

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 Boot-Time 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 durch das Neukompilieren deines eigenen Kernels lösen. Dein Kernel wird die **option BOOT_CONFIG** benötigen, die GENERIC bereits beinhaltet.

Den Großteil dieses Dokumentes kannst du in der Manualseite boot_config(8) finden. Um in die Benutzerkernelkonfiguration (User Kernel Config) oder UKC zu gelangen, mußt du beim Starten die -c Option verwenden.

```
boot> boot wd0a:/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 au
- `exit/quit` - Setze Starten fort
- `show` **`[attr [val]]`** - Zeige alle Geräte mit Eigenschaft (attribute) und optional mit einem spezifierten Wert (value)

Wenn du einmal dein Gerät konfiguriert hast, dann steige mit quit oder exit aus UKC aus und setze das Starten fort. Nun solltest du deine Kernelkonfiguration korrigieren und einen eigenen Kernel kompilieren. Siehe Einen eigenen Kernel kompilieren für weitere Hilfe.

5.6 - Mehr Lognachrichten während des Startens

Mehr Lognachrichten während des Startens zu bekommen kann sehr hilfreich sein, wenn man Bootprobleme lösen will. Wenn du ein Problem hast und du zu wenig Informationen beim Starten erhältst, dann gib beim Starten bei "boot>" wieder "-c" ein, um zur UKC zu gelangen, dann:

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

Nun wirst du extrem ausführliche Meldungen beim Booten erhalten.

5.7 - Mittels config(8) deine Kernelbinärdatei verändern

Die **-e** und **-u** Options von config(8). können sehr hilfreich sein, und Zeit sparen, die du mit dem Neukompilierewn 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 Anderungen einfach ignoriert und nicht in eine Kernelbinärdatei geschrieben. Für weitere Informationen über Fehler- und Warnmeldungen siehe die config(8) Manualseite.

```
$ sudo config -e /bsd
OpenBSD 3.2 (GENERIC) #25: Thu Oct  3 19:51:53 MDT 2002
    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
 65 ep0 disabled
 66 ep* disabled
 67 ep* disabled
149 ep0 disabled
150 ep0 disabled
151 ep* disabled
152 ep* disabled
202 ep* disabled
ukc> quit
not forced
```

Im obigen Beispiel werden alle ep* Geräte im Kernel deaktiviert und werden 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 sind, 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 3.2 (GENERIC) #25: Thu Oct  3 19:51:53 MDT 2002
    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
```

[FAQ Index] [Zu Kapitel 4 - Installationsleitfaden] [Zu Kapitel 6 - Netzwerk Einstellungen]

---

www@openbsd.org
Originally [OpenBSD: faq5.html,v 1.80]
$Translation: faq5.html,v 1.49 2003/04/09 16:41:45 jufi Exp $
$OpenBSD: faq5.html,v 1.46 2003/04/09 17:03:01 jufi Exp $

6 - Netzwerk

Inhaltsverzeichnis

6.0.1 - Bevor wir weiter gehen

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

Wenn du ein Netzwerkadministrator bist und 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 beim IP Netzwerken aufbauen kann, insbesondere wenn man mit mehreren Netzwerken arbeitet oder für sie verantwortlich ist.

Wenn du mit Anwendungen wie Web-, FTP- oder Mailserver 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.

6.1 - Erste Netzwerkeinstellungen

6.1.1 - Identifzieren 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 mittels des Befehls dmesg(8). Weiterhin kannst du mit dem Befehl ifconfig(8) deine Karte überprüfen. Als Beispiel hier die Ausgabe in dmesg für eine Intel Fast Ethernet Netzwerk-Karte, 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 weisst, sieh bitte in der Liste der unterstützten Hardware für deine Plattform nach. Du wirst eine Liste vieler bekannte 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 Name" (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. Diese Beispielausgabe zeigt ein physikalisches Ethernet Interface, eine 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
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
```

ifconfig(8) gibt uns 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, das standardmässig eine Menge mehr Interfaces aktiviert sind. Dies sind virtuelle Interfaces, die verschiedene Funktionen haben. Informationen dazu findest du auf den folgenden man pages:

- lo - Loopback Interface
- pflog - Packet Filter Logging Interface
- sl - SLIP Network 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

Der 1. Schritt zur Konfiguration deiner Netzwerkkarte ist das Erstellen der **/etc/hostname.xxx** Datei, wobei der Name deiner Karte den Platz von xxx einnehmen sollte. Aus der Information der obigen Beispiele würde der Name **/etc/hostname.fxp0** lauten. Das Layout dieser Datei sollte einfach so aussehen:

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

(Viel mehr Details zu dieser Datei findest du in der hostname.if(5) man page.) Eine typische Interface-Konfigurationsdatei für eine IPv4 Addresse 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 settings 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 172.21.0.0 255.255.255.0 NONE vlan 2 vlandev fxp1
```

Der nächste Schritt ist das Einstellen deines Standard-Gateways (default gateway). Dazu trag 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 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** Script 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 bereits einige der Routen in der kernel routing table. 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 via netstat(1) oder route(8) überprüfen. Wenn du Probleme mit dem Routing hast, möchtest du vielleicht das -n Flag für route(8) benutzen, dass die IP-Adressen ausgibt, statt einen DNS Lookup zu machen, und den Hostnamen anzuzeigen. Hier ist ein Beispiel mit beiden Kommandos:

```
$ 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
```

6.1.2 - Einrichten deines OpenBSD Rechners als Gateway

Dies sind nur die grundlegende 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 -w 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 routed(8), gated, mrtd, und zebra. OpenBSD hat Unterstützung in der ports Kollektion sowohl für gated, zebra als auch mrtd. OpenBSD unterstützt mehrere T1, HSSI, ATM, FDDI, Ethernet und serielle (PPP/SLIP) Schnittstellen.

6.1.3 - Einrichten von Aliases auf deiner Netzwerkkarte

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

- IP für dc0 ist 192.168.0.2
- NETMASK ist 255.255.255.0

Ein paar Bemerkungen zu Aliasen: Bei 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 durchführen. 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
```

6.2 - Packet Filter (PF)

Packet Filter (ab hier nur noch als PF bezeichnet) ist OpenBSD's 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 und ausserdem eine Priorisierung von Paketen durchzuführen. PF ist seit OpenBSD Version 3.0 Teil des OpenBSD GENERIC Kernels. Beschrieben wird PF im neuen **PF User's Guide**. Die alte PF FAQ ist nach wie vor hier zu lesen.

6.4 - DHCP

6.4.1 DHCP Klient

Um den DHCP Klient 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 xl1 xl2 xl3 >/etc/dhcpd.interfaces
```

Dies wird OpenBSD veranlassen, den DHCP Klient 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 Klient von der Befehlszeile 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, du kannst 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), einstetzen willst, editiere /etc/rc.conf. Setze dhcpd_flags="-q" anstelle von dhcpd_flags=NO. 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 Klienten mit, daß die an DNS Anfragen anzuhängende Domäne example.com ist (d. h., wenn der Benutzer schreibt 'telnet joe', 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 starten willst, nachdem du /etc/dhcpd.conf editiert hast, versuche:

```
# dhcpd -q fxp0
```

Wobei `fxp0` die Netzwerkkarte ist, auf der DHCP serviert werden soll. Die `-q` Option setzt die Ausgabe von dhcpd(8) auf ruhig, ansonsten ist sie sehr ausführlich.

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 Klienten wünschen.

6.5 - PPP

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

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

Den ersten, den wir behandeln, wird der Userland PPP Dämon sein. Um zu beginnen, benötigen wir einige einfache Informationen über deinen ISP. Hier eine Liste hilfreicher Informationen, die du brauchen wirst.

- Die Einwahlnummer deines ISP
- Deinen Nameserver
- Deinen Benutzernamen und Password
- Dein Gateway

Einige von diesen benötigst du nicht unbedingt, aber sie wären hilfreich. Der Userland PPP Dämon 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.

Solltest du keinen GENERIC Kernel verwenden, dann stelle sicher, daß du folgende Zeile in deiner Kernelkonfigurationsdatei hast:

```
pseudo-device   tun             2
```

Erste Einstellungen - für PPP(8)

Die ersten Einstellungen für den Userland PPP Dämon 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 kreieren. Hier werde ich mit dem einfachsten und gebrächlichsten Einstellungen beginnen. Hier eine schnelle **ppp.conf** Datei, die uns einfach zu deinem ISP verbindet und die Standardrouten und Nameserver setzt. Für diese Datei brauchst du nur die Telefonnummer deines ISP sowie deinen Benutzernamen und dein Passwort.

```
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 die 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 wird die Geschwindigkeit unserer Dialup Verbindung und mit "set dial" setzen wir unsere Dialup Parameter, mit denen wir die timeout Zeit, usw. setzen können. Diese Zeile sollte eigentlich ziemlich genau so, wie sie jetzt ist, bleiben.

Nun können wir die ISP spezifischen Informationen eintragen. Wir tun dies, indem wir unter **default:** einen weiteren Absatz hinzufügen. Dieser kann als alles benannt werden, 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 Telefonnummer deines ISP. "set login" setzt unsere login-Optionen. Hier haben wir die 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 Passwort senden. In diesem Beispiel ist unser username = ppp und das Password = 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 obigen 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 kein Bits von dieser IP Adresse übereinstimmen muß 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 um 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.

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" 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.

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.

- **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, was sich hervorragend eignet, um Probleme in deiner **ppp.conf** Datei zu debuggen.

ppp(8) Extras

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

- ppp.linkup
- ppp.linkdown

Weitere Informationen findest du im FreeBSD Handbook entry on User PPP.

6.6 - Netzwerkparameter tunen

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 `sysctl -w`, wie z. B. `sysctl -w net.inet.tcp.keepidle=28800`.

6.6.2 - Wie kann ich "directed broadcasts" aktivieren?

Normalerweise willst du dies nicht tun. Dies erlaubt jemand, 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 -w 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 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 -w  net.inet.tcp.baddynamic=749,750,751,760,761,871
```

```
Es ist ebenso möglich ports aus der aktuellen Liste hinzuzufügen
oder zu entfernen.

 #  sysctl -w net.inet.tcp.baddynamic=+748
 #  sysctl -w 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 Manualseiten 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: Ein Server im LAN und Klienten 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 sein, der du Zugriff auf den NFS Server läßt. 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.

NFS hängt von portmap(8) ab, bevor es funktionieren kann. Portmap(8) ist ab OpenBSD 3.2 standardmässig abgeschaltet, du musst es also in rc.conf(8) wieder einschalten, indem du die `portmap` Zeile wie folgt änderst:

```
portmap=YES
```

und ein Reboot ist notwendig, damit die Änderung wirksam wird.

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 exports(5) 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 Klienten jedes Verzeichnis unter dem **/work** Mount-point mounten können. **-ro** bedeutet, daß nur Leseberechtigung gestattet wird. Die letzten zwei Argumente bedeuten, daß nur Klienten 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 **nfs_server=YES** in */etc/rc.conf* eintragen. 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 von NFS Serverdiensten einsetzen, um die maximale Anzahl von gleichzeitigen Klientenanfragen, 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 eien 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 neu startest!

```
# 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**. Z. B.:

```
$ /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 mount_nfs(8). Um ein Dateisystem /work von Host 10.0.0.1 auf dem lokalen Dateisystem /mnt zu laden, tue folgendes (NB: du mußt nicht IP Adressen verwenden, mount wird Hostnamen auflösen):

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

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

```
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. Z. B.:

```
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 Klient untergraben. Wenn du keine Programme auf diesem NFS Dateisystem auf dem NFS Klient ausführen willst, füge noexec hinzu:

6.9 - Eine PPTP Verbindung mit OpenBSD aufsetzen

HINWEIS: Dies bezieht sich nicht auf **ALLE** ADSL Provider, aber viele der Informationen können aus diesem Setup übernommen werden. Dieses Setup funktioniert auf jeden Fall bei Inode, einem ADSL Provider in Österreich.

Als Anfang musst du pptp installiert haben. Der Port befindet sich unter */usr/ports/net/pptp*. Lies FAQ 8, Ports um mehr Informationen über den OpenBSD ports tree zu bekommen.

Wegen des Konflikts der "Im-Kernel" gre(4) Unterstützung und pptp wirst du deinen Kernel neu kompilieren müssen und die Unterstützung für gre(4) entfernen müssen.

Patch, um die GRE(4) Unterstützung zu entfernen.

```
Index: GENERIC
===================================================================
RCS file: /cvs/src/sys/conf/GENERIC,v
retrieving revision 1.86
diff -u -r1.86 GENERIC
--- GENERIC     14 Mar 2002 00:42:25 -0000      1.86
+++ GENERIC     17 May 2002 01:52:17 -0000
@@ -87,7 +87,7 @@
 pseudo-device  enc     1       # option IPSEC needs the encapsulation interface
 pseudo-device  bridge  2       # network bridging support
 pseudo-device  vlan    2       # IEEE 802.1Q VLAN
-pseudo-device  gre     1       # GRE encapsulation interface
+#pseudo-device gre     1       # GRE encapsulation interface
 #pseudo-device strip   1       # Starmode Radio IP interface

 pseudo-device  pty     64      # pseudo-terminals
```

Um deinen Kernel neu zu kompilieren mache einen "check out" von OpenBSD via cvs (siehe die OpenBSD Stable Webseite) , benutze den folgenden Patch, und baue einen neuen Kernel wie unter FAQ 5, Building a kernel.

Nachdem du das **pptp** package und einen neuen Kernel installiert hast, musst du ein paar Dateien für deine neue Verbindung editieren. Diese packages benutzen das standarmässige OpenBSD ppp(8) wenn du dich also mit ppp(8) auskennst, kommt dir vieles bekannt vor. Siehe auch FAQ 6, Setup.

- 1 - /etc/ppp/options

- 2 - /etc/ppp/pap-secrets

Für eine */etc/ppp/options* Datei wird ein Setup wie das unten vermutlich alles notwendige tun:

```
# cat /etc/ppp/options
name "LOGINNAME"
noauth
noipdefault
defaultroute
debug
```

`LOGINNAME` sollte mit deiner User-ID ersetzt werden.

In */etc/ppp/pap-secrets* gehört eine Zeile wie diese hier:

```
# cat /etc/ppp/pap-secrets
LOGINNAME 10.0.0.138 PASSWORD
```

Wobei LOGINNAME deine User-ID und PASSWORD dein Password ist. 10.0.0.138 ist die zugewiesene IP deines Modems im Falle, dass du ADSL nutzt, etc. Stelle sicher, dass diese Datei nur von root gelesen werden kann (mode 600).

6.9.1 - Deinem Network Interface eine Adresse zuweisen

Im obigen Beispiel hatte unser Modem eine vorkonfigurierte Adresse von 10.0.0.138. Jetzt müssen wir UNSEREM Interface noch eine Adresse zuweisen. Es ist am besten eine IP zu wählen, die nahe an der deines MODEMS liegt, oder einfach die statische Adresse zu benutzen, die dir zugewiesen wurde. Mehr darüber, wie man Interfaces IP-Adressen zuweist, kannst du in FAQ 6.1 lesen.

Wenn dein Interface eingerichtet ist, solltest du eine pptp Verbindung mit dem Kommando

```
# /usr/local/sbin/pptp 10.0.0.138 &
```

aufbauen können.

Da hier auch der "in-house" OpenBSD ppp(8) benutzt wird, werden hier zwei Prozesse gestartet. Du kannst pptp beenden, indem du diesen beiden Prozesse killst:

```
# kill -9 [pid of pppd]
$ kill -9 [pid of pptp]
```

Wir empfehlen `/var/log/messages` in einem weiteren Terminalfenster zu öffnen, um mögliche Probleme zu erkennen.

```
# tail -f /var/log/messages
```

Wir schlagen vor, die Startsequenz in */etc/rc.local* unterzubringen, so dass bei jedem reboot die Verbindung automatisch aufgebaut wird.

6.10 - 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 Traffic 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 von 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 Netzwerk-Karte 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 Netzwerk-Karte mit einem Koax-Port erlaubt mir, diese Maschine als Bridge in das Koax-Netzwerk zu benutzen.

Dieses System hat jetzt zwei Netzwerk-Karten (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 man page aufgeführt.

Jetzt müsen wir die Bridge aufsetzen. Bridges werden durch die Existenz einer Datei namens `bridgename.bridge0`. initialisiert. Hier ist zum Beispiel ein 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 welche 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 zu 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 diesen alten Maschinen trifft. Ich möchte, dass nur Web und SSH-Traffic zu Ihnen durchkommt. In diesem Fall lassen wir jeglichen Traffic nach draussen zu, filtern aber 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 diese Regelwerk jeglichen Netzwerk-Verkehr mit Ausnahme von hereinkommendem HTTP und SSH-Traffic 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.

Tips 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.

[FAQ Index] [Zu Kapitel 5 - Neuerzeugen des Systems aus dem Quellcode] [Zu Kapitel 7 - Tastatur- und Bildschirmkontrollen]

 www@openbsd.org

Originally [OpenBSD: faq6.html,v 1.177]
$Translation: faq6.html,v 1.74 2003/07/27 16:22:36 jufi Exp $
$OpenBSD: faq6.html,v 1.58 2003/07/27 17:39:32 jufi Exp $

7 - Tastatur- und Display-Kontrollen

Inhaltsverzeichnis

7.1 - Wie verändere ich die Belegung der Tastatur? *(wscons)*

Die ports, die den wscons(4) Console Treiber benutzen: alpha, hppa, i386, macppc, sparc, und sparc64.

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 "Control L" ist:

```
# wsconsctl -w keyboard.map+="keysym Caps_Lock = Control_L"
```

7.2 - Gibt es so etwas wie gpm für OpenBSD?

Für die Plattformen the alpha und i386 bietet OpenBSD den wsmoused(8), einen port des moused(8) von FreeBSD. Er kann beim 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 *(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]+[PGUP]` und `[SHIFT]+[PGDN]`

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 wechsele ich die Konsolen?*(i386 und einige Alphas)*

Auf dem 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 übrgi lässt. Zwischen ihnen hin- und herspringen kannst du, indem du Tastaturkombinationen `[CTRL]+[ALT]+[F1]`, `[CTRL]+[ALT]+[F2]`, `[CTRL]+[ALT]+[F3]`, `[CTRL]+[ALT]+[F4]` und `[CTRL]+[ALT]+[F6]` benutzt.

Die X Umgebung benutzt ttyC4, `[CTRL]+[ALT]+[F5]`. Wenn du X benutzt, bringen dich die `[CTRL]+[ALT]+[F`*n*`]` Tasten auf die Text-Konsolen, `[CTRL]+[ALT]+[F5]` bringt dich wieder auf die grafische Öberfläche. 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 `[CTRL]+[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? *(i386)*

i386 User haben normalerweise eine Konsole mit 25 Zeilen und jede hat 80 Buchstaben. Viele VGA Grafik-Karten 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 man pages 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 `[CTRL]-[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 80x25 ist mit `[CTRL]-[ALT]-[F6]` erreichbar.

Denke daran, dass `[CTRL]-[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, `[CTRL]-[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 Grafik-Karten. Dummerweise unterstützen nicht alle Grafik-Karten 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 weit über die Möglichkeiten dieses Artikels hinaus, geben wir die 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 User 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 Sessions 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 draran,

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 notwenidg das System im Single-User-Modus zu booten, damit die Änderungen durchgeführt werden können.

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 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 boot loader eingegeben werden, wenn es nur für ein einzelnes Mail (oder zum ersten Mal) genutzt werden soll.

i386 Hinweise:

- OpenBSD zählt die seriellen Ports beginned mit *tty00*, DOS/Windows mit *COM1*. Denke als 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 i386 Systeme sind in der Lage, den seriellen Port als das Console Device zu benutzen. Deine Ergebnisse könen hiervon abweichen -- dieser Port ist nach dem Booten vielleicht nicht dem Betriebssystem zugänglich, und es kann daher notwendig sein, ZWEI serielle Ports zu verwenden.
- 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 seriellen 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 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 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 erreichbar. 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 Geschwidigkeit 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 Mac anschliessen müssen. Wenn das in Zukunft deine Standardeinstellung sein soll, musst du dem Booter-Programm sagen, es soll deine Optionen speichern.

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 Hintzentwicklung neuerer Monitore. Im Modus `off` wird zwar der Bildschirm schwarz, aber der Monitor empfängt weiter das honrizontale und vertikale sync Signal, und die Rückkehr zum Normalbetrieb geschieht augenblicklich.
- **`display.screen_off`** gibt die Verzögerungszeit im 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.

[FAQ Index] [Zum Kapitel 6 - Netzwerken] [Zum Kapitel 8 - Allgemeine Fragen]


www@openbsd.org

Originally [OpenBSD: faq7.html,v 1.56]
$Translation: faq7.html,v 1.35 2003/12/13 19:50:59 jufi Exp $
$OpenBSD: faq7.html,v 1.32 2003/12/13 20:23:03 jufi Exp $

8 - General Questions

Table of Contents

8.1 - I forgot my root password, what do I do now?

A few steps to recovery

1. Boot into single user mode. For i386 arch type boot -s at the boot prompt.
2. mount the drives.
   # **fsck -p / && mount -uw /**
3. If /usr is not the same partition that / is (and it shouldn't be) then you will need to mount it, also
   # **fsck -p /usr && mount /usr**
4. run passwd(1)
5. boot into multiuser mode... and *remember* your password!

8.2 - X won't start, I get lots of error messages

If you have X completely set up and you are using an XF86Config that you know works then the problem most likely lies in the machdep.allowaperture. You also need to make sure that:

```
option APERTURE
```

is in your kernel configuration. [It is already in the GENERIC kernel]

Then you need to edit */etc/sysctl.conf* and set **machdep.allowaperture=2**. This will allow X to access the aperture driver. This would already be set if you said that you would be running X when asked during the install. OpenBSD requires for all X servers that the aperture driver be set, because it controls access to the I/O ports on video boards.

For other X problems on the i386, consult the XFree86 Online Documentation at http://www.xfree86.org/support.html.

8.3 - What is CVS? and How do I use it?

CVS is the tool that OpenBSD project uses to control changes to the source code. CVS stands for Concurrent Versions System. You can read more about CVS at http://www.cvshome.org/. CVS can be used by the end user to keep up to date with source changes, and changes in the ports tree. CVS makes it extremely simple to download the source via one of the many CVS mirrors for the project.

How to initially setup your CVS environment

You can retrieve the sources from one of the OpenBSD AnonCVS servers. These servers are listed on http://www.openbsd.org/anoncvs.html. Once you have chosen a server you need to choose which module you are going to retrieve. There are three main modules available for checkout from the CVS tree. These are:

- *src* - The src module has the complete source code for OpenBSD. This includes userland and kernel sources.
- *ports* - The ports module holds all you need to have the complete OpenBSD ports tree. To read more on the OpenBSD ports tree, read FAQ 8, Ports of the OpenBSD FAQ.
- *XF4* - The XF4 module contains the source to compile XFree 4.

Now that you have decided which module that you wish to retrieve, there is one more step left before you can retrieve it. You must decide which method to use. CVS by default retrieves files using ssh(1), but some AnonCVS servers allow for the use of rsh. For those of you behind a firewall there are also the options of pserver and some AnonCVS servers run ssh on port 2022. Be sure to check http://www.openbsd.org/anoncvs.html for which servers support what protocols. Next I will show how to do a simple source checkout. Here I will be using an AnonCVS server located in the U.S., but remember that if you are outside of the U.S you need to use a server that is located nearby. There are many AnonCVS servers located throughout the world, so choose one nearest you. I will also be using ssh to retrieve the files.

```
$ export CVSROOT=anoncvs@anoncvs.usa.openbsd.org:/cvs
$ cvs get src
Warning: Remote host denied X11 forwarding, perhaps xauth program could not be run on
the server side.
cvs checkout: in directory src:
cvs checkout: cannot open CVS/Entries for reading: No such file or directory
cvs server: Updating src
U src/Makefile
[snip]
```

Notice here also that I set the CVSROOT environment variable. This is the variable that tells cvs(1) which AnonCVS server to use. This can also be specified using the *-d* option. For example:

```
$ cvs -d anoncvs@anoncvs.usa.openbsd.org:/cvs get src
```

These commands should be run in */usr*, which will then create the directories of */usr/src*, */usr/ports*, and */usr/www*. Depending, of course, on which module you checkout. You can download these modules to anywhere, but if you wanted to do work with them (ie make build), it is expected that they be at the place above.

Keeping your CVS tree up-to-date

Once you have your initial tree setup, keeping it up-to-date is the easy part. You can update your tree at any time you choose, some AnonCVS servers update more often then others, so again check http://www.openbsd.org/anoncvs.html. In this example I will be updating my www module from anoncvs.usa.openbsd.org. Notice the -q option that I use, this makes the output not so verbose coming from the server.

```
$ echo $CVSROOT
anoncvs@anoncvs.usa.openbsd.org:/cvs
$ cvs -q up -Pd www
Warning: Remote host denied X11 forwarding, perhaps xauth program could not be run on
the server side.
U www/want.html
M www/faq/faq8.html
ericj@oshibana:~>
```

Other cvs options

For some, bandwidth and time are serious problems when updating repositories such as these. So CVS has a -z[1-9] option which uses gzip to compress the data. To use it, do -z[compression-level], for instance, -z3 for a compression level of 3.

8.4 - What is the ports tree?

The ports tree is a set of Makefiles that download, patch, configure and install userland programs so you can run them in OpenBSD environment without having to do all that by hand. You can get the ports tree from any of the OpenBSD FTP servers in /pub/OpenBSD/3.4/ports.tar.gz. The most recent ports are available via the 'ports' cvs tree, or /pub/OpenBSD/snapshots/ports.tar.gz. For most of you however, packages will be a much better option. Packages are created from ports and are already compiled and ready to use. To read more on packages read FAQ 8, Packages.

Important note about keeping your system and ports in sync

OpenBSD has three "active" versions at any point in time:

- Release: What is on the CD.
- Stable: Release, plus security and reliability enhancements
- Current: The development version of OpenBSD.

DO NOT mix versions of Ports and OpenBSD!

If your system is Release, use the Release version of the ports tree. Don't try to use a -Current version of the Ports tree on a -Release or -Stable system. Not only is it not likely to work, you will irritate people when you ask for help about why "nothing seems to work!" Note that there is a -Stable branch of the Ports tree as well, where critical fixes to -Release ports will be made.

Yes, this really does mean a wonderful new port will not typically work on your "older" system -- even if that system was -current just a few weeks ago.

If you do not have the ports tree installed, you can download it via any of OpenBSD's FTP servers, or of course, from the CD-ROM. The file is ports.tar.gz, and you want to untar this in the /usr directory, which will create /usr/ports, and all the directories under it. For example:

```
$ ftp ftp://ftp.openbsd.org/pub/OpenBSD/3.4/ports.tar.gz
$ sudo cp ports.tar.gz /usr
$ cd /usr; sudo tar xzf ports.tar.gz
```

A snapshot of the ports tree is also created daily and can be downloaded from any of the OpenBSD FTP servers as /pub/OpenBSD/snapshots/ports.tar.gz. If you are installing a snapshot of OpenBSD, you should use a matching snapshot of ports. Again, make sure you keep your ports tree and your OpenBSD system in sync.

What ports are available? How do i find them?

Use the ports tree to search for keywords. To do this use make search key="searchkey". Here is an example of a search for 'samba':

```
$ make search key="samba"
  [...snip...]
Port:   amanda-client-2.4.2.2
Path:   misc/amanda,-client
Info:   network-capable tape backup (client only)
Maint:  Tom Schutter <t.schutter@att.net>
Index:  misc
L-deps:
B-deps: :devel/gmake gnuplot-*:math/gnuplot gtar-*:archivers/gtar
samba-*:net/samba/stable
R-deps:
Archs:  any

Port:   samba-2.2.8a
Path:   net/samba/stable
Info:   SMB and CIFS client and server for UNIX
Maint:  The OpenBSD ports mailing-list <ports@openbsd.org>
Index:  net
L-deps: popt::devel/popt
B-deps: :devel/autoconf/2.13 :devel/metaauto
R-deps:
Archs:  any
  [...snip...]
```

Installing Ports

Ports are set up to be EXTREMELY easy to make and install. Here is an example showing how to install the X11 program `xfig`. You'll notice the dependencies are automatically detected and completed.

First you need to cd to the dir of the program you want. If you are searching for a program, you can either update your locate database, or use the search function talked about above. Once you are in the dir of the program you want, you can just type make install. For example.

```
$ sudo make install
===>  Checking files for xfig-3.2.4
>> xfig.3.2.4.full.tar.gz doesn't seem to exist on this system.
>> Attempting to fetch /usr/ports/distfiles/xfig.3.2.4.full.tar.gz from http://w
ww.xfig.org/xfigdist/.
100% |************************************************|  5042 KB    00:31
>> Checksum OK for xfig.3.2.4.full.tar.gz. (sha1)
===>  xfig-3.2.4 depends on: jpeg.62 - jpeg.62 missing...
===>  Verifying install for jpeg.62 in graphics/jpeg
===>  Checking files for jpeg-6b
>> jpegsrc.v6b.tar.gz doesn't seem to exist on this system.
>> Attempting to fetch /usr/ports/distfiles/jpegsrc.v6b.tar.gz from ftp://ftp.uu
.net/graphics/jpeg/.
'EPSV': command not understood.
100% |************************************************|   598 KB    00:06
>> Checksum OK for jpegsrc.v6b.tar.gz. (sha1)
===>  Extracting for jpeg-6b
===>  Patching for jpeg-6b
===>  Configuring for jpeg-6b
checking for gcc... cc
checking whether the C compiler (cc -O2   ) works... yes
checking whether the C compiler (cc -O2   ) is a cross-compiler... no
checking whether we are using GNU C... yes
  [...snip...]
```

Using Flavors

Many of the applications in the ports tree support different install options, called *flavors*. If a port comes in multiple flavors, you can use these options simply by setting an environment variable before you compile the port. If multiple features are needed, the FLAVOR variable can be set to a space-delimited list of the supported and desired flavors. Currently, many ports have flavors that include database support, support for systems without X, or network additions like SSL and IPv6.

```
$ pwd
/usr/ports/net/mtr
$ make show=FLAVORS
no_x11
$ env FLAVOR="no_x11" make
===>  mtr-0.49-no_x11 depends on: gmake-3.80 - not found
===>  Verifying install for gmake-3.80 in devel/gmake
===>  Checking files for gmake-3.80
>> make-3.80.tar.gz doesn't seem to exist on this system.
>> Attempting to fetch /usr/ports/distfiles/make-3.80.tar.gz from ftp://ftp.gnu.
org/gnu/make/.
Unknown command.
100% |************************************************|  1183 KB    00:07
>> Checksum OK for make-3.80.tar.gz. (sha1)
  [...snip...]
$ sudo env FLAVOR="no_x11" make install
===>  Faking installation for mtr-0.49-no_x11
  [...snip...]
===>  Building package for mtr-0.49-no_x11
Creating package /usr/ports/packages/i386/All/mtr-0.49-no_x11.tgz
Using SrcDir value of /usr/ports/net/mtr/w-mtr-0.49-no_x11/fake-i386-no_x11/usr/
local
Creating gzip'd tar ball in '/usr/ports/packages/i386/All/mtr-0.49-no_x11.tgz'
===>  Installing mtr-0.49-no_x11 from /usr/ports/packages/i386/All/mtr-0.49-no_a
x11.tgz
```

Listing Installed ports/packages

You can see a list of both ports and packages by using the pkg_info command.

```
$ /usr/sbin/pkg_info
zsh-4.1.1           The Z shell.
```

```
screen-3.9.15       A multi-screen window manager.
emacs-21.3          GNU editing macros.
tcsh-6.12.00        An extended C-shell with many useful features.
bash-2.05b          The GNU Borne Again Shell.
zip-2.3             Create/update ZIP files compatible with pkzip.
ircII-20030314      An enhanced version of ircII, the Internet Relay Chat client
ispell-3.2.06       An interactive spelling checker.
tin-1.6.1           TIN newsreader (termcap based)
procmail-3.22       A local mail delivery agent.
strobe-1.06         Fast scatter/gather TCP port scanner
lsof-4.68           Lists information about open files.
ntp-4.1.74          Network Time Protocol Implementation.
ncftp-3.1.5p0       ftp replacement with advanced user interface
nmh-1.0.4p1         The New MH mail handling program
bzip2-1.0.2         A block-sorting file compressor
```

Other Information

More information about the ports can be found in the ports(7) man page and on the Ports page.

Our ports tree is constantly being expanded, and if you would like to help please see: http://www.openbsd.org/porting.html.

8.5 - What are packages?

Packages are the precompiled binaries of some of the most used programs. They are ready for use on an OpenBSD system. Again, like the ports, packages are very easy to maintain and update. Packages are constantly being added so be sure to check each release for additional packages.

Here is a list of tools used in managing packages.

- pkg_add(1) - a utility for installing software package distributions
- pkg_create(1) - a utility for creating software package distributions
- pkg_delete(1) - a utility for deleting previously installed software package distributions
- pkg_info(1) - a utility for displaying information on software packages

Where to find packages

If you are a smart user and bought an OpenBSD CD set, then packages can be found on one of the three CDs, depending on your architecture. If you don't have an OpenBSD CD in your possession you can download packages from any of the ftp mirrors. You can get a list of mirrors http://www.openbsd.org/ftp.html. Packages are located at */pub/OpenBSD/3.4/packages* from there packages are broken down depending on architecture.

Installing Packages

To install packages, the utility pkg_add(1) is used. pkg_add(1) is an extremely easy utility to use, in the following two examples pkg_add(1) will be used to install a package. The first example will show pkg_add(1) installing a package that resides on a local disk, the second example will show an installation of a package via ftp. In both examples screen-3.9.15 will be installed.

Installing via local disk

```
$ sudo pkg_add -v screen-3.9.15.tgz
Requested space: 749864 bytes, free space: 2239117312 bytes in
/var/tmp/instmp.cpsHA27596
Running install with PRE-INSTALL for `screen-3.9.15'
extract: Package name is screen-3.9.15
extract: CWD to /usr/local
extract: /usr/local/bin/screen-3.9.15
extract: execute 'ln -sf screen-3.9.15 /usr/local/bin/screen'
extract: /usr/local/man/man1/screen.1
extract: /usr/local/info/screen.info
extract: execute '[ -f /usr/local/info/dir ] || sed -ne '1,/Menu:/p'
/usr/share/info/dir > /usr/local/info/dir'
extract: execute 'install-info /usr/local/info/screen.info /usr/local/info/dir'
extract: /usr/local/lib/screen/screencap
extract: /usr/local/lib/screen/screenrc
extract: CWD to .
Running mtree for `screen-3.9.15'
mtree -q -U -f +MTREE_DIRS -d -e -p /usr/local
```

```
Running install with POST-INSTALL for `screen-3.9.15'

+---------------
| The file /etc/screenrc has been created on your system.
| You may want to verify/edit its contents
|
| The file /usr/local/lib/screen/screencap contains a
| termcap like description of the screen virtual terminal.
| You may use it to update your terminal database.
| See termcap(5).
+---------------

Attempting to record package into `/var/db/pkg/screen
Package `screen-3.9.15' registered in `/var/db/pkg/screen-3.9.15'
```

In this example the **-v** flag was used to give a more verbose output. This option is not needed but is helpful for debugging and was used here to give a little more insight into what pkg_add(1) is actually doing. Notice however, that there are some valid messages given out mentioning */etc/screenrc*. Messages like this will be given to you whether or not you use the **-v** flag.

Installing via ftp

```
$ sudo pkg_add ftp://ftp.openbsd.org/pub/OpenBSD/3.4/packages/i386/screen-3.9.15.tgz
>>> ftp -o - ftp://ftp.openbsd.org/pub/OpenBSD/3.4/packages/i386/screen-3.9.15.tgz

+---------------
| The file /etc/screenrc has been created on your system.
| You may want to verify/edit its contents
|
| The file /usr/local/lib/screen/screencap contains a
| termcap like description of the screen virtual terminal.
| You may use it to update your terminal database.
| See termcap(5).
+---------------
```

In this example you can see that I installed the i386 package. You should substitute *i386* (above) with your architecture. Notice: Not all architectures have the same packages. Some ports don't work on certain architectures. In this example the **-v** flag wasn't used, so only NEEDED messages are shown.

Viewing and Deleting Installed Packages

The utility pkg_info(1) is used to view a list of packages that are already installed on your system. This is usually needed to find out the correct name of a package before you remove that package. To see what packages are installed on your system simple use:

```
$ pkg_info
mpg123-0.59rp1     mpeg audio 1/2 layer 1, 2 and 3 player
nmap-3.00          port scanning large networks
ircII-20030314     enhanced version of ircII (internet relay chat)
screen-3.9.15      multi-screen window manager
unzip-5.50r2       extract, list & test files in a ZIP archive
ntp-4.1.74         Network Time Protocol implementation
icb-5.0.9p1        Internet CB - mostly-defunct chat client
```

To delete a package, simple take the proper name of the package as shown by pkg_info(1) and use pkg_delete(1) to remove the package. In the below example, the screen package is being removed. Notice that on some occasions there are instructions of extra objects that need to be removed that pkg_delete(1) did not remove for you. As with the pkg_add(1) utility, you can use the **-v** flag to get more verbose output.

```
$ sudo pkg_delete screen-3.9.15

+---------------
| To completely deinstall the screen-3.9.15 package you need to perform
| this step as root:
|
|           rm -f /etc/screenrc
|
| Do not do this if you plan on re-installing screen-3.9.15
| at some future time.
+---------------
```

8.6 - Should I use Ports or Packages?

In general, you are HIGHLY advised to use packages over building an application from ports. The OpenBSD ports team considers packages to be the goal of their porting work, not the ports themselves.

Building a complex application from source is not trivial. Not only must the application be compiled, but the tools used to build it must be built. Unfortunately, OpenBSD, the tools, and the application are all evolving, and often, getting all the pieces working together is a challenge. Once everything works, a revision in any of the pieces the next day could render it broken. Every six months, as a new release of OpenBSD is made, an effort is made to test the building of every port on every platform, but during the development cycle it is likely that some ports will break.

In addition to having all the pieces work together, there is just the matter of time and resources required to compile some applications from source. A common example is CVSup, a tool commonly used to track the OpenBSD source tree. To install CVSup on a moderately fast system with a good Internet connection may take only about ten seconds -- the time required to download and unpack a single 511kB package file. In contrast, building CVSup on the same machine from source is a huge task, requiring many tools and bootstrapping a compiler, takes almost half an hour on the same machine. Other applications, such as Mozilla or KDE may take hours and huge amounts of disk space and RAM/swap to build. Why go through this much time and effort, when the programs are already compiled and sitting on your CD-ROM or FTP mirror, waiting to be used?

Of course, there are a few good reasons to use ports over packages in some cases:

- Distribution rules prohibit OpenBSD from distributing a package.
- You wish to modify or debug the application or study its source code.
- You need a FLAVOR of a port that is not built by the OpenBSD ports team.
- You wish to alter the directory layout (i.e., modifying PREFIX or SYSCONFDIR)

However, for most people and most applications, using packages is a far easier, and is the recommended way of adding applications to OpenBSD.

8.8 - Is there any way to use my floppy drive if it's not attached during boot?

You need to set the kernel to always assume the floppy is attached, even if not detected during the hardware probe, by setting the 0x20 flag bit on fdc(4). This can be done by using User Kernel Config or config(8) to alter your kernel,

```
# config -e -f /bsd
OpenBSD 3.4 (GENERIC) #18: Wed Sep 17 03:34:47 MST 2003
    deraadt@i386.openbsd.org:/usr/src/sys/arch/i386/compile/GENERIC
Enter 'help' for information
ukc> change fd*
204 fd* at fdc0 drive -1 flags 0x0
change [n] y
drive [-1] ? ENTER
flags [0] ? 0x20
204 fd* changed
204 fd* at fdc0 drive -1 flags 0x20
ukc> q
Saving modified kernel.
#
```

You can also change this by adding "flags 0x20" at the end of the fd* entry in your Kernel Configuration file recompile your kernel. The line should read:

```
fd*     at fdc? drive ? flags 0x20
```

8.9 - OpenBSD Bootloader (*i386 specific*)

When booting your OpenBSD system, you have probably noticed the boot prompt.

boot>

For most people, you won't have to do anything here. It will automatically boot if no commands are given. But sometimes problems arise, or special functions are needed. That's where these options will come in handy. To start off, you should read through the boot(8) man page. Here we will go over the most common used commands for the bootloader.

To start off, if no commands are issued, the bootloader will automatically try to boot **/bsd**. If that fails it will try **/obsd**, and if that fails, it will try **/bsd.old**. You can specify a kernel by hand by typing:

```
boot> boot hd0a:/bsd
```

or

```
boot> b /bsd
```

This will boot the kernel named `bsd` from the 'a' partition of the first BIOS recognized hard disk.

Here is a brief list of options you can use with the OpenBSD kernel.

- **-a** : This will allow you to specify an alternate root device after booting the kernel.
- **-c** : This allows you to enter the boot time configuration. Check the Boot Time Config section of the faq.
- **-s** : This is the option to boot into single user mode.
- **-d** : This option is used to dump the kernel into ddb. Keep in mind that you must have DDB built into the kernel.

These are entered in the format of: **boot [image [-acds]]**

For further reading you can read boot_i386(8) man page

8.10 - S/Key

S/Key is a ``one-time password'' scheme. This allows for one-time passwords for use on un-secured channels. This can come in handy for those who don't have the ability to use ssh or any other encrypted channels. OpenBSD's S/Key implementation can use a variety of algorithms as the one-way hash. The following algorithms are available:

- md4
- md5
- sha1
- rmd160.

Setting up S/Key - The first steps

To start off the directory */etc/skey* must exist. If this directory is not in existence, have the super-user create it. This can be done simply by doing:

```
# mkdir /etc/skey
```

Once that directory is in existence, you can initialize your S/Key. To do this you will have to use skeyinit(1). With skeyinit(1), you will first be prompted for your password to the system. This is the same password that you used to log into the system. Running skeyinit(1) over an insecure channel is completely not recommended, so this should be done over a secure channel (such as ssh) or the console. Once you have authorized yourself with your system password you will be asked for yet another password. This password is the S/Key *secret password*, and is **NOT** your system password. Your secret password must be at least 10 characters. We suggest using a memorable phrase containing several words as the secret password. Here is an example user being added.

```
$ skeyinit ericj
[Adding ericj]
Reminder - Only use this method if you are directly connected
   or have an encrypted channel.  If you are using telnet
   or rlogin, exit with no password and use skeyinit -s.
Enter secret password:
Again secret password:

ID ericj skey is otp-md5 99 oshi45820
Next login password: HAUL BUS JAKE DING HOT HOG
```

One line of particular importance in here is *ID ericj skey is otp-md5 99 oshi45820*. This gives a lot of information to the user. Here is a breakdown of the sections and their importance.

- *otp-md5* - This shows which one-way was used to create your One-Time Password (otp).
- *99* - This is your sequence number. This is a number from 100 down to 1. Once it reaches one, another secret password must be created.
- *oshi45820* - This is your key.

But of more immediate importance is your password. Your password consists of 6 small words, combined together this is your password, spaces and all.

Actually using S/Key to login.

By now your skey has been initialized, and you have your password. You're ready to login. Here is an example session using S/Key to login. Starting with OpenBSD 3.0, S/Key logins work differently. For OpenBSD 3.0 and above you append **:skey** to your login name. For versions of OpenBSD previous to 3.0 you use **s/key** for the password at which time you are prompted for your S/Key password (the exception to this is ftpd(8) which will always present an S/Key challenge for S/Key-enabled user prior to OpenBSD < 3.0). The examples below assume OpenBSD 3.0 or higher.

```
$ ftp localhost
Connected to localhost.
220 oshibana.shin.ms FTP server (Version 6.5/OpenBSD) ready.
Name (localhost:ericj): ericj:skey
331- otp-md5 96 oshi45820
331 S/Key Password:
230- OpenBSD 3.4 (GENERIC) #18: Wed Sep 17 03:34:47 MDT 2003
230-
230- Welcome to OpenBSD: The proactively secure Unix-like operating system.
230-
230- Please use the sendbug(1) utility to report bugs in the system.
230- Before reporting a bug, please try to reproduce it with the latest
230- version of the code.  With bug reports, please try to ensure that
230- enough information to reproduce the problem is enclosed, and if a
230- known fix for it exists, include that as well.
230-
230 User ericj logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> quit
221 Goodbye.
```

Note that I appended ":skey" to my username. This tells ftpd that I want to authenticate using S/Key. Some of you might have noticed that my sequence number has changed to *otp-md5 96 oshi45820*. This is because by now I have used S/Key to login several times. But how do you get your password after you've logged in once? Well to do this, you'll need to know what sequence number you're using and your key. As you're probably thinking, how can you remember which sequence number you're on? Well this is simple, use skeyinfo(1), and it will tell you what to use. For example here, I need to generate another password for a login that I might have to make in the future. (remember I'm doing this from a secure channel).

```
$ skeyinfo
95 oshi45820
```

An even better way is to use **skeyinfo -v**, which outputs a command suitable to be run in the shell. For instance:

```
$ skeyinfo -v
otp-md5 95 oshi45820
```

Not only is *otp-md5* a description of the hash used, it is also an alternate name for the skey(1) command. So, the simplest way to generate the next S/Key password is simply:

```
$ `skeyinfo -v`
Reminder - Do not use this program while logged in via telnet or rlogin.
Enter secret password:
NOOK CHUB HOYT SAC DOLE FUME
```

Note the backticks in the above example.

I'm sure many of you won't always have a secure connection to create these passwords, and creating them over an insecure connection isn't feasible, so how can you create multiple passwords at one time? Well you can supply skey(1) with a number of how many passwords you want created. This can then be printed out and taken with you wherever you go.

```
$ otp-md5 -n 5 95 oshi45820
Reminder - Do not use this program while logged in via telnet or rlogin.
Enter secret password:
91: SHIM SET LEST HANS SMUG BOOT
92: SUE ARTY YAW SEED KURD BAND
93: JOEY SOOT PHI KYLE CURT REEK
94: WIRE BOGY MESS JUDE RUNT ADD
95: NOOK CHUB HOYT SAC DOLE FUME
```

Notice here though, that the bottom password should be the first used, because we are counting down from 100.

Using S/Key with telnet(1), ssh(1), and rlogin(1)

Using S/Key with telnet(1), ssh(1), or rlogin(1) is done in pretty much the same fashion as with ftp--you simply tack ":skey" to the end of your username. Example:

```
$ telnet localhost
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

OpenBSD/i386 (oshibana) (ttyp2)

login: ericj:skey
otp-md5 98 oshi45821
S/Key Password: SCAN OLGA BING PUB REEL COCA
Last login: Thu Oct  7 12:21:48 on ttyp1 from 156.63.248.77
OpenBSD 3.4 (GENERIC) #18: Wed Sep 17 03:34:47 MDT 2003

Welcome to OpenBSD: The proactively secure Unix-like operating system.

Please use the sendbug(1) utility to report bugs in the system.
Before reporting a bug, please try to reproduce it with the latest
version of the code.  With bug reports, please try to ensure that
enough information to reproduce the problem is enclosed, and if a
known fix for it exists, include that as well.

You have mail.
$
```

8.12 - Does OpenBSD support SMP? (Symmetric Multi-Processor)

Not at this time. Work is on-going, there is an OpenBSD SMP project, though there is nothing usable yet, nor is there any time schedule for OpenBSD SMP support.

On most platforms, OpenBSD will run on an SMP system, but only utilizing one processor. The exception to this is the SPARC platform -- OpenBSD/sparc will sometimes require that extra MBus modules be removed for the system to boot. Multi-processor SPARC64 systems run as long as the base machine is supported.

How can I help?

- Write code. There is lots of work to do.
- Donate hardware. The goal of the OpenBSD SMP project is to support as many platforms as possible, including SPARC, SPARC64, Alpha, MacPPC and of course, the i386. Donations of desired hardware are welcome!
- Don't offer to "test". The SMP project is not part of the core OpenBSD project right now for a reason -- it isn't ready for production use yet, or even general "testing". Informing the developers that it didn't work on your hardware without code to make it work isn't useful at this point. Send in your dmesg, if a developer sees something interesting in your hardware, they can contact you for testing.

8.13 - I get Input/output error when trying to use my tty devices

You need to use /dev/cuaXX for connections initiated from the OpenBSD system, the /dev/ttyXX devices are intended only for terminal or dial-in usage. While it was possible to use the tty devices in the past, the OpenBSD kernel is no longer compatible with this usage.

From cua(4):

For hardware terminal ports, dial-out is supported through matching device nodes called calling units. For instance, the terminal called /dev/tty03 would have a matching calling unit called /dev/cua03. These two devices are normally differentiated by creating the calling unit device node with a minor number 128 greater than the dial-in device node. *Whereas the dial-in device (the tty) normally requires a hardware signal to indicate to the system that it is active, the dial-out device (the cua) does not, and hence can communicate unimpeded with a device such as a modem.* This means that a process like getty(8) will wait on a dial-in device until a connection is established. Meanwhile, a dial-out connection can be established on the dial-out device (for the very same hardware terminal port) without disturbing anything else on the system. The getty(8) process does not even notice that anything is happening on the terminal port. If a connecting call comes in after the dial-out connection has finished, the getty(8) process will deal with it properly, without having noticed the intervening dial-out action.

8.14 - What web browsers are available for OpenBSD?

Lynx, a text-based browser, is in the base system, and has SSL support. Other browsers in the ports tree, include (in no particular order):

Graphical (X) Browsers

- Dillo Minimal feature set, very fast and small, runs well on slower hardware.
- Konqueror Installed as part of the KDE desktop environment.
- Konqueror-embedded (konq-e) Konqueror, using only the KDE libraries rather than all of KDE.
- Netscape 4 For sparc and i386 only, not Open Source, no package available.
- Opera Commercial browser, i386 only.
- Amaya The W3C's browser and editor.
- Links+ Another fast and small graphical browser. (Also has a text-only mode)
- *(new for 3.4)* Mozilla and Firebird Feature-filled browsers. Mozilla includes a many non-browser features (mail client, IRC client, etc.), Firebird is just a browser, based on Mozilla. Works on alpha, i386, sparc, and sparc64 platforms.

Console (Text mode) Browsers

- links Has table support.

You will find all these in the ports collection. All the above mentioned browsers are located in `/usr/ports/www/` after the installation of the ports tree. Most are also available as pre-compiled packages, available on the FTP servers and on the CD-ROM. As most of the graphical browsers are very large and require quite some time to download and compile, one should *seriously* consider the use of packages where available.

Unfortunately, due to a bug in one of the libraries used by w3m, it does not work on 3.4-release on any platform.

8.15 - How do I use the mg editor?

Mg is a micro Emacs-style text editor included in OpenBSD. Micro means that it's small (Emacs is very large!) For the basics, read the mg(1) manual page and the tutorial, as included with the source code. For more interesting questions (such as, "I don't have a Meta key!") check out the Emacs FAQ.

Note that since mg is a small Emacs implementation, which is mostly similar to the text editor features of Emacs 17, it does not implement many of Emacs' other functionality. (Including mail and news functionality, as well as modes for Lisp, C++, Lex, Awk, Java, etc...)

8.16 - ksh(1) does not appear to read my `.profile`!

There are two likely reasons for ksh(1) to seemingly ignore a user's `.profile` file.

- `.profile` is not owned by the user. To fix for **username**,

  ```
  # chown username ~username/.profile
  ```

- You are using ksh(1) from within X Window System

  Under xterm(1), argv[0] for ksh(1) is not prepended with a dash ("-"). Prepending a dash to argv[0] will cause csh(1) and ksh(1) to know they should interpret their login files. (For csh(1) that's `.login`, with a separate `.cshrc` that is always run when csh(1) starts up. With ksh(1), this is more noticeable because there is only one startup script, `.profile`. This file is ignored unless the shell is a login shell.)

  To fix this, add the line "`XTerm*loginShell: true`" to the file `.Xdefaults` in your home directory. Note, this file does not exist by default, you may have to create it.

  ```
  $ echo "XTerm*loginShell: true" >> ~/.Xdefaults
  ```

  You may not have had to do this on other systems, as some installations of X Window System come with this setting as default. OpenBSD has chosen to follow the XFree86 behavior.

8.17 - Why does my `/etc/motd` file get overwritten when I modified it?

The `/etc/motd` file is edited upon every boot of the system, replacing everything up to, but not including, the first blank line with the system's kernel version information. When editing this file, make sure that you start after this blank line, to keep `/etc/rc` from deleting these lines when it edits `/etc/motd` upon boot.

8.18 - Why does www.openbsd.org run on Solaris?

Although none of the developers think it is particularly relevant, this question comes up frequently enough in the mailing lists that it is answered here. *www.openbsd.org* and the main OpenBSD ftp site are hosted at a SunSITE at the University of Alberta, Canada. These sites are hosted on a large Sun system, which has access to lots of storage space and Internet bandwidth. The presence of the SunSITE gives the OpenBSD group access to this bandwidth. This is why the main site runs here. Many of the OpenBSD mirror sites run OpenBSD, but since they do not have guaranteed access to this large amount of bandwidth, the group has chosen to run the main site at the University of Alberta SunSITE.

8.19 - I'm having problems with my PCI devices being detected

There exists a condition where some machines might not detect some PCI devices properly, or might freeze while detecting multiple NIC's in one machine. This is the fault of PCIBIOS, and involves a simple workaround to make it work properly. Simply enter the boot time configuration and disable PCIBIOS. An example is below:

```
boot> boot -c
Copyright (c) 1982, 1986, 1989, 1991, 1993
        The Regents of the University of California.  All rights reserved.
Copyright (c) 1995-2002 OpenBSD. All rights reserved.  http://www.OpenBSD.org

OpenBSD 3.4 (GENERIC) #18: Wed Sep 17 03:34:47 MDT 2003
    deraadt@i386.openbsd.org:/usr/src/sys/arch/i386/compile/RAMDISK_CD
cpu0: Intel Pentium III (Coppermine) ("GenuineIntel" 686-class, 128KB L2 cache)
1 GHz
cpu0: FPU,V86,DE,PSE,TSC,MSR,PAE,MCE,CX8,SYS,MTRR,PGE,MCA,CMOV,PAT,PSE36,MMX,FXS
R,SIMD
real mem  = 267956224 (261676K)
avail mem = 243347456 (237644K)
using 3296 buffers containing 13500416 bytes (13184K) of memory
User Kernel Config
UKC> disable pcibios
UKC> quit
[... snip ...]
```

Once this is done, you can follow the directions in FAQ 5, Building a Kernel to create a new kernel so that you don't have to worry about this in the future.

8.20 - Antialiased and TrueType fonts in XFree86

See this document.

8.21 - Does OpenBSD support any journaling filesystems?

No it doesn't. We use a different mechanism to achieve similar results that is called Soft Updates. Please read in FAQ 14 to get more details.

8.22 - Reverse DNS

- or -

Why is it taking so long for me to log in?

Many new users to OpenBSD experience a two minute login delay when using services such as ssh, ftp, or telnet. This can also be experienced when using a proxy, such as ftp-proxy, or when sending mail out from a workstation through sendmail.

This is almost always due to a reverse-DNS problem. DNS is Domain Name Services, the system the Internet uses to convert a name, such as "www.openbsd.org" into a numeric IP address. Another task of DNS is the ability to take a numeric address and convert it back to a "name", this is "Reverse DNS".

In order to provide better logging, OpenBSD performs a reverse-DNS lookup on any machine that attaches to it in many different ways, including ssh, ftp, telnet, sendmail or ftp-proxy. Unfortunately, in some cases, the machine that is making the connection does not have a proper reverse DNS entry.

An example of this situation:

A user sets up an OpenBSD box as a firewall and gateway to their internal home network, mapping all their internal computers to one external IP using NAT. They may also use it as an outbound mail relay. They follow the installation guidelines, and are very happy with the results, except for one thing -- every time they try to attach to the box in any way, they end up with a two minute delay before things happen.

What is going on:

From a workstation behind the NAT of the gateway with an unregistered IP address of 192.168.1.35, the user uses ssh to access the gateway system. The ssh client prompts for username and password, and sends them to the gateway box. The gateway then tries to figure out who is trying to log in by performing a reverse DNS lookup of 192.168.1.35. The problem is 192.168.0.0 addresses are for private use, so a properly configured DNS server outside your network knows it should have no information about those addresses. Some will quickly return an error message, in these cases, OpenBSD will assume there is no more information to be gained, and it will quickly give up and just admit the user. Other DNS servers will not return ANY response. In this case you will find yourself waiting for the OpenBSD name resolver to time out, which takes about two minutes before the login will be permitted to continue. In the case of ftp-proxy, some ftp clients will timeout before the reverse DNS query times out, leading to the impression that ftp-proxy isn't working.

This can be quite annoying. Fortunately, it is an easy thing to fix.

Fix, using `/etc/hosts`:

The simplest fix is to populate your `/etc/hosts` file with all the workstations you have in your internal network, and ensure that your `/etc/resolv.conf` file contains the line `lookup file bind` which ensures that the resolver knows to start with the `/etc/hosts` file, and failing that, to use the DNS servers specified by the "nameserver" lines in your `/etc/resolv.conf` file.

Your `/etc/hosts` file will look something like this:

```
::1 localhost.in.example.org localhost
127.0.0.1 localhost.in.example.org localhost
192.168.1.1 gw.in.example.org gw
192.168.1.20 scrappy.in.example.org scrappy
192.168.1.35 shadow.in.example.org shadow
```

Your `resolv.conf` file will look something like this:

```
search in.example.org
nameserver 24.2.68.33
nameserver 24.2.68.34
lookup file bind
```

A common objection to this is "But, I use DHCP for my internal network! How can I configure my `/etc/hosts`?" Rather easily, actually. Just enter lines for all the addresses your DHCP server is going to give out, plus any static devices:

```
::1 localhost.in.example.org localhost
127.0.0.1 localhost.in.example.org localhost
192.168.1.1 gw.in.example.org gw
192.168.1.20 scrappy.in.example.org scrappy
192.168.1.35 shadow.in.example.org shadow
192.168.1.100 d100.in.example.org d100
192.168.1.101 d101.in.example.org d101
192.168.1.102 d102.in.example.org d102
        [... snip ...]
192.168.1.198 d198.in.example.org d198
192.168.1.199 d199.in.example.org d199
```

In this case, I am assuming you have the DHCP range set to 192.168.1.100 through 192.168.1.199, plus the three static definitions as listed at the top of the file.

If your gateway must use DHCP for configuration, you may well find you have a problem -- dhclient will overwrite your `/etc/resolv.conf` every time the lease is renewed, which will remove the "lookup file bind" line. This can be solved by putting the line "lookup file bind" in the file `/etc/resolv.conf.tail`.

Fix, using a local DNS server

Details on this are somewhat beyond the scope of this document, but the basic trick is to setup your favorite DNS server, and make sure it knows it is authoritative for both forward and reverse DNS resolution for all nodes in your network, and make sure your

computers (including your gateway) know to use it as a DNS server.

8.23 - Why do the OpenBSD web pages not conform to HTML4/XHTML?

The present web pages have been carefully crafted to work on a wide variety of actual browsers going back to browser versions 4.0 and later. We do not want to make these older pages conform to HTML4 or XHTML until we're sure that they will also work with older browsers; it's just not a priority. We welcome new contributors, but suggest you work on writing code, or on documenting new aspects of the system, not on tweaking the existing web pages to conform to newer standards.

8.24 - Why is my clock off by twenty-some seconds?

When using rdate(8) to synchronize your clock to a NTP server, you may find your clock is off by twenty-some seconds from your local definition of time.

This is caused by a difference between the UTC (Coordinated Universal Time, based on astronomical observations) time and TAI (International Atomic Time, based on atomic clocks) time. To compensate for variations in the earth's rotation, "leap seconds" are inserted into UTC, but TAI is unadjusted. These leap seconds are the cause of this discrepancy. For a more detailed description, search the web for "`leap seconds UTC TAI`".

Addressing the problem is fairly simple. In most countries you will get the correct time if you use the "`-c`" parameter to rdate(8) and use a time zone out of the directory `/usr/share/zoneinfo/right/`. For example, if you are located in Germany, you could use these commands:

```
# cd /etc && ln -sf /usr/share/zoneinfo/right/CET localtime
# rdate -ncv ptbtime1.ptb.de
```

In other countries, the rules may differ.

[FAQ Index] [To Section 7 - Keyboard and Display Controls] [To Section 9 - Migrating from Linux]

---

www@openbsd.org

$OpenBSD: faq8.html,v 1.142 2004/02/10 04:18:12 nick Exp $

9 - Umstieg von Linux

Inhaltsverzeichnis

Weitere Informationen für Linux Benutzer gibt es unter http://sites.inka.de/mips/unix/bsdlinux.html.

9.1 - Einfache Tips für Linux (und andere freie Unix-artige BS) User

Es gibt einige Unterschiede zwischen OpenBSD und Linux. Diese Unterschiede sind unter anderem die Bootup-Sequenz, Netzwerkkarten Benutzung und das Festplatten- Management. Die meisten Unterschiede sind gut dokumentiert, aber benötigen ein Suchen in den man-pages. Dieses Dokument versucht als Index dieser Unterschiede zu dienen.

- OpenBSD hat einen ports tree. Das wirkt der Tatsache entgegen, dass bis jetzt nicht allzuviele Anwendungen rein für die OpenBSD-Umgebung geschrieben wurden. Dies ist sowohl ein Versuch mehr Anwendungen für OpenBSD verfügbar zu machen, als auch darauf hinzuwirken, dass mehr OpenBSD-kompatible Anwendungen entwickelt werden. Eventuell wird dieser ports tree dazu verwendet, eine nette Zusammenstellung von Binärpaketen zu machen.
- OpenBSD benutzt CVS für Source-Änderungen. Bei Linux ist der Source-Code zwischen verschiedenen Distributionen verteilt und zum Teil unterschiedlich. OpenBSD hat das 'anonymous CVS' zuerst eingeführt, was jedermann erlaubt, den vollen 'source tree' für jede Version von OpenBSD herunterzuladen, (von 2.0 bis -current, und alle Zwischenversionen von allen Dateien zwischendurch) und das zu jeder Zeit! Es gibt auch ein sehr bequemes und einfach zu bedienendes Webinterface zu CVS.
- OpenBSD gibt regelmässig 'snapshots' für verschiedene Architekturen heraus und alle 6 Monate gibt es eine stabile offizielle Version auf CD.
- OpenBSD enthält STARKE KRYPTOGRAFIE, die Betriebssysteme aus den USA nicht enthalten können. (Siehe http://www.openbsd.org/de/crypto.html) OpenBSD wurde auch einer starken Sicherheitsüberprüfung unterzogen und viele Sicherheitsfeatures wurden bereits in den Source tree eingefügt. (IPSEC, KERBEROS).
- Der Kernel von OpenBSD's heisst /bsd.
- Die Namen von Festplatten sind für gewöhnlich */dev/wd* (IDE) und */dev/sd* (SCSI oder ATA Geräte, die SCSI Festplatten emulieren)
- /sbin/ifconfig ohne Argumente ergibt unter Linux den Status aller Netzwerkinterfaces. Unter OpenBSD benötigst du das -a Flag.
- /sbin/route ergibt unter Linux den Status aller aktiven Routen. Unter OpenBSD brauchst du den "*show*" Parameter, oder nimm einfach **netstat -r**.
- OpenBSD unterstützt KEINE Journaling Filesysteme wie ReiserFS, IBMs JFS oder SGIs XFS. Stattdessen benutzen wir Soft Updates.
- OpenBSD enthält das Paket Filter Paket (pf), nicht ipfw, ipchains, netfilter, iptables, oder IP Filter. Das bedeutet:
  - Network Adress Translation (=NAT, in Linux unter IP-Masquerading bekannt) wird durch pfctl realisiert (mittels -N). (pfctl(8))
  - ipfwadm wird durch pfctl realisiert. (pfctl(8), pf(4), pf.conf(5))
  - Du solltest dir Kapitel 6 der FAQ für detailliertere Konfigurationshilfen und Informationen ansehen.
- Die Interface-Adresse wird in /etc/hostname.<interfacename> aufbewahrt. Sie kann auch ein Hostname anstatt

einer IP Addresse sein.

- Der Name deiner Maschine ist in */etc/myname*
- Das Standard-Gateway ist in */etc/mygate*
- OpenBSD's Standard-Shell ist /bin/sh, also die Korn shell. Shells wie bash und tcsh können als Pakete hinzugefügt oder aus dem ports tree installiert werden.
- Das Passwort-Management ist deutlich verändert. Die Hauptdateien sind verschieden. (passwd(1), passwd(5))
- Devices werden nach dem Treiber benannt und nicht nach ihrem Typ. Zum Beispiel gibt es keine eth* Devices. Aber es gibt ne0 für eine NE2000 Ethernet-Karte, und xl0 für 3Com Etherlink XL und Fast Etherlink XL Ethernet Device, etc.
- Die OpenBSD Entwickler haben grosse Anstrengungen unternommen, um die 'manual pages' auf den aktuellen Stand zu bringen und aufzuräumen. Unter man(1) gibt es weitere Informationen dazu.

9.2 - Dual boot mit Linux und OpenBSD

Ja! Es ist machbar!

Lies INSTALL.linux

9.3 - Deine Linux (oder andere System-7-artige) Passwort-Datei in BSD-Style konvertieren.

Zuerst finde 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 System 7-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 an:195 gelangt, 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 System 7-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 System7-artigen Passwort-Dateien 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` *`username`*.

9.4 - Wie man OpenBSD und Linux zur Zusammenarbeit bewegt

Wenn du von Linux zu OpenBSD wechselst, denk daran, dass im OpenBSD GENERIC Kernel die Option COMPAT_LINUX standardmässig aktiviert ist. 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 page folgen. Ein einfacher Weg, die meisten nützlichen Linux Libraries 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 Libraries zu installieren:

```
# cd /usr/ports/emulators/redhat/base
# make install
```

OpenBSD unterstützt das EXT2FS Dateisystem. Benutze **disklabel (8)** ***disk*** (wobei ***disk*** der 'device name' für deine Festplatte ist.) um zu überprüfen, was OpenBSD denkt, was deine Linux Partition ist (aber benutze disklabel oder fdisk **nicht** um daran Änderungen vorzunehmen). Für weitere Informationen über die Benutzung von disklabel lies FAQ 14 - Disklabel.

[FAQ Index] [Zum Kapitel 8 - Allgemeine Fragen] [Zum Kapitel 10 - System Administration]

 www@openbsd.org

Originally [OpenBSD: faq9.html,v 1.51]
$Translation: faq9.html,v 1.31 2003/07/27 15:48:47 jufi Exp $
$OpenBSD: faq9.html,v 1.32 2003/07/27 15:59:26 jufi Exp $

10 - System Management

Table of Contents

10.1 - Why does it say that I'm in the wrong group when I try to su root?

Existing users must be added to the "wheel" group by hand. This is done for security reasons, and you should be cautious with whom you give access to. On OpenBSD, users who are in the wheel group are allowed to use the su(1) userland program to become root. Users who are not in "wheel" cannot use su(1). Here is an example of a /etc/group entry to place the user **ericj** into the "wheel" group.

If you are adding a new user with adduser(8), you can put them in the wheel group by answering wheel at "`Invite` *`user`* `into other groups:`". This will add them to /etc/group, which will look something like this:

```
wheel:*:0:root,ericj
```

If you are looking for a way to allow users limited access to superuser privileges without putting them in the "wheel" group, use sudo(8).

10.2 - How do I duplicate a filesystem?

To duplicate your filesystem use dump(8) and restore(8). For example, to duplicate everything under directory SRC to directory DST, do a:

```
# cd /SRC; dump 0f - . | (cd /DST; restore -rf - )
```

dump is designed to give you plenty of backup capabilities, and it may be an overkill if you just want to duplicate a part of a (or an entire) filesystem. The command tar(1) may be faster for this operation. The format looks very similar:

```
# cd /SRC; tar cf -  . | (cd /DST; tar xpf - )
```

10.3 - How do I start daemons with the system? (Overview of rc(8))

OpenBSD uses an rc(8) style startup. This uses a few key files for startup.

- /etc/rc - Main script. Should not be edited.
- /etc/rc.conf - Configuration file used by */etc/rc* to know what daemons should start with the system.
- /etc/rc.conf.local - Configuration file you can use to override settings in /etc/rc.conf so you don't have to touch /etc/rc.conf itself, which is convenient for people who upgrade often.
- /etc/netstart - Script used to initialize the network. Shouldn't be edited.
- /etc/rc.local - Script used for local administration. This is where new daemons or host specific information should be stored.
- /etc/rc.securelevel - Script which runs commands that must be run before the security level changes. See init(8)
- /etc/rc.shutdown - Script run on shutdown. Put anything you want done before shutdown in this file. See rc.shutdown(8)

How does rc(8) work?

The main files a system administrator should concentrate on are */etc/rc.conf* (or */etc/rc.conf.local*), */etc/rc.local* and */etc/rc.shutdown*. To get a look of how the rc(8) procedure works, here is the flow:

After the kernel is booted, */etc/rc* is started:

- Filesystems are checked. This will be bypassed if the file /etc/fastboot exists. This is certainly not a good idea though.
- Configuration variables are read in from */etc/rc.conf* and, afterwards, */etc/rc.conf.local*. Settings in rc.conf.local will override those in rc.conf.
- Filesystems are mounted
- Clears out */tmp* and preserves any editor files
- Configures the network via */etc/netstart*
  - Configures your interfaces up.
  - Sets your hostname, domainname, etc.
- Starts system daemons
- Performs various other checks (quotas, savecore, etc)
- Local daemons are run, via */etc/rc.local*

Starting Daemons and Services that come with OpenBSD

Most daemons and services that come with OpenBSD by default can be started on boot by simply editing the */etc/rc.conf* configuration file. To start out take a look at the default /etc/rc.conf file. You'll see lines similar to this:

```
ftpd_flags=NO           # for non-inetd use: ftpd_flags="-D"
```

A line like this shows that ftpd is not to start up with the system (at least not via rc(8), read the Anonymous FTP FAQ to read more about this). In any case, each line has a comment showing you the flags for **NORMAL** usage of that daemon or service. This doesn't mean that you must run that daemon or service with those flags. You can always use man(1) to see how you can have that daemon or service start up in any way you like. For example, here is the default line pertaining to httpd(8).

```
httpd_flags=NO          # for normal use: "" (or "-DSSL" after reading ssl(8))
```

Here you can obviously see that starting up httpd normally no flags are necessary. So a line like: " **httpd_flags=**""" would be necessary. But to start httpd with ssl enabled. (Refer to the SSL FAQ or ssl(8)) You should start with a line like: "httpd_flags="-DSSL"".

A good approach is to never touch */etc/rc.conf* itself. Instead, create the file */etc/rc.conf.local*, copy just the lines you are about to change from */etc/rc.conf* and adjust them as you like. This may make future upgrading easier -- all the changes are in the one file.

Starting up local daemons and configuration

For other daemons that you might install with the system via ports or other ways, you should use the */etc/rc.local* file. For example, I've installed a daemon which lies at /usr/local/sbin/daemonx. I want this to start at boot time. I would put an entry into */etc/rc.local* like this:

```
if [ -x /usr/local/sbin/daemonx ]; then
            echo -n ' daemonx';       /usr/local/sbin/daemonx
fi
```

(If the daemon does not automatically detach on startup, remember to add a "&" at the end of the command line.)

From now on, this daemon will be run at boot. You will be able to see any errors on boot, a normal boot with no errors would show a line like this:

```
Starting local daemons: daemonx.
```

rc.shutdown

/etc/rc.shutdown is a script that is run at shutdown. Anything you want done before the system shuts down should be added to this file. If you have apm, you can also set "powerdown=YES". Which will give you the equivalent of "shutdown -p".

10.4 - Why do users get "relaying denied" when they are remotely sending mail through my OpenBSD system?

Try this:

```
# cat /etc/mail/sendmail.cf | grep relay-domains
```

The output may look something like this:

```
FR-o /etc/mail/relay-domains
```

If this file doesn't exist, create it. You will need to enter the hosts who are sending mail remotely with the following syntax:

```
.domain.com    #Allow relaying for/to any host in domain.com
sub.domain.com #Allow relaying for/to sub.domain.com and any host in that domain
10.2           #Allow relaying from all hosts in the IP net 10.2.*.*
```

Don't forget send a 'HangUP' signal to sendmail, (a signal which causes most daemons to re-read their configuration file):

```
# kill -HUP `head -1 /var/run/sendmail.pid`
```

Further Reading

- http://www.sendmail.org/~ca/email/relayingdenied.html
- http://www.sendmail.org/tips/relaying.html
- http://www.sendmail.org/antispam.html

10.5 - I've set up POP, but users have trouble accessing mail through POP. What can I do?

Most issues dealing with POP are problems with temporary files and lock files. If your pop server sends an error message such as:

```
-ERR Couldn't open temporary file, do you own it?
```

Try setting up your permissions as such:

```
permission in  /var
drwxrwxr-x   2 bin     mail     512 May 26 20:08 mail


permissions in  /var/mail
-rw-------   1 username   username        0 May 26 20:08 username
```

Another thing to check is that the user actually owns their own /var/mail file. Of course this should be the case (as in, /var/mail/joe should be owned by joe) but if it isn't set correctly it could be the problem!

Of course, making /var/mail writable by group mail opens up some vague and obscure security problems. It is likely that you will never have problems with it. But it could (especially if you are a high profile site, ISP,...)! There are several POP servers you can install right away from the ports collection. If possible, use popa3d which is available in the OpenBSD base install. Or, you could just have the wrong options selected for your pop daemon (like dot locking). Or, you may just need to change the directory that it locks in (although then the locking would only be valuable for the POP daemon.)

PS: Notice, OpenBSD does not have a group name of "mail". You need to create this in your */etc/group* file if you need it. An entry like:

```
mail:*:6:
```

would be sufficient.

10.6 - Why does Sendmail ignore `/etc/hosts file`?

By default, Sendmail uses DNS for name resolution, not the `/etc/hosts` file. The behavior can be changed through the use of the `/etc/mail/service.switch` file.

If you wish to query the hosts file before DNS servers, create a `/etc/mail/service.switch` file which contains the following line:

```
hosts       files dns
```

If you wish to query ONLY the hosts file, use the following:

```
hosts       files
```

Send Sendmail a HUP signal:

```
# kill -HUP `head -1 /var/run/sendmail.pid`
```

and the changes will take effect.

10.7 - Setting up a Secure HTTP server with SSL(8)

OpenBSD ships with an SSL-ready httpd and RSA libraries. For use with httpd(8), you must first have a certificate created. This will be kept in */etc/ssl/* with the corresponding key in */etc/ssl/private/*. The steps shown here are taken in part from the ssl(8) man page. Refer to it for further information. This FAQ entry only outlines how to create an RSA certificate for web servers, not a DSA server certificate. To find out how to do so, please refer to the ssl(8) man page.

To start off, you need to create your server key and certificate using OpenSSL:

```
# openssl genrsa -out /etc/ssl/private/server.key 1024
```

Or, if you wish the key to be encrypted with a passphrase that you will have to type in when starting servers

```
# openssl genrsa -des3 -out /etc/ssl/private/server.key 1024
```

The next step is to generate a Certificate Signing Request which is used to get a Certifying Authority (CA) to sign your certificate. To do this use the command:

```
# openssl req -new -key /etc/ssl/private/server.key -out /etc/ssl/private/server.csr
```

This *server.csr* file can then be given to Certifying Authority who will sign the key. One such CA is **Thawte Certification** which you can reach at http://www.thawte.com/.

If you cannot afford this, or just want to sign the certificate yourself, you can use the following.

```
# openssl x509 -req -days 365 -in /etc/ssl/private/server.csr \
       -signkey /etc/ssl/private/server.key -out /etc/ssl/server.crt
```

With */etc/ssl/server.crt* and */etc/ssl/private/server.key* in place, you should be able to start httpd(8) with the **-DSSL** flag (see the section about rc(8) in this faq), enabling https transactions with your machine on port 443.

10.8 - I edited /etc/passwd, but the changes didn't seem to take place. Why?

If you edit */etc/passwd* directly, your changes will be lost. OpenBSD generates */etc/passwd* dynamically with pwd_mkdb(8). The main password file in OpenBSD is */etc/master.passwd*. According to pwd_mkdb(8),

```
FILES
     /etc/master.passwd  current password file
     /etc/passwd         a Version 7 format password file
     /etc/pwd.db         insecure password database file
     /etc/pwd.db.tmp     temporary file
     /etc/spwd.db        secure password database file
```

```
/etc/spwd.db.tmp    temporary file
```

In a traditional Unix password file, such as /etc/passwd, everything including the user's encrypted password is available to anyone on the system (and is a prime target for programs such as Crack). 4.4BSD introduced the master.passwd file, which has an extended format (with additional options beyond those provided by /etc/passwd) and is only readable by root. For faster access to data, the library calls which access this data normally read /etc/pwd.db and /etc/spwd.db.

OpenBSD does come with a tool with which you should edit your password file. It is called vipw(8). Vipw will use vi (or your favourite editor defined per $EDITOR) to edit /etc/master.passwd. After you are done editing, it will re-create /etc/passwd, /etc/pwd.db, and /etc/spwd.db as per your changes. Vipw also takes care of locking these files, so that if anyone else attempts to change them at the same time, they will be denied access.

10.9 - What is the best way to add and delete users?

OpenBSD provides two commands for easily adding users to the system:

- adduser(8)
- user(8)

You can also add users by hand, using vipw(8), but this is more difficult for most operations.

The easiest way to add a user in OpenBSD is to use the adduser(8) script. You can configure adduser(8) by editing */etc/adduser.conf*. adduser(8) allows for consistency checks on */etc/passwd*, */etc/group*, and shell databases. It will create the entries and $HOME directories for you. It can even send a message to the user welcoming them. Here is an example user, **testuser**, being added to a system. He/she will be given the $HOME directory */home/testuser*, made a member of the group **guest**, and given the shell */bin/ksh*.

```
# adduser
Use option ``-silent'' if you don't want to see all warnings and questions.

Reading /etc/shells
Reading /etc/login.conf
Check /etc/master.passwd
Check /etc/group

Ok, let's go.
Don't worry about mistakes. I will give you the chance later to correct any input.
Enter username []: testuser
Enter full name []: Test FAQ User
Enter shell csh ksh nologin sh [sh]: ksh
Uid [1002]: Enter
Login group testuser [testuser]: guest
Login group is ``guest''. Invite testuser into other groups: guest no
[no]: no
Login class auth-defaults auth-ftp-defaults daemon default staff
[default]: Enter
Enter password []: Type password, then Enter
Enter password again []: Type password, then Enter

Name:        testuser
Password:    ****
Fullname:    Test FAQ User
Uid:         1002
Gid:         31 (guest)
Groups:      guest
Login Class: default
HOME:        /home/testuser
Shell:i      /bin/ksh
OK? (y/n) [y]: y
Added user ``testuser''
Copy files from /etc/skel to /home/testuser
Add another user? (y/n) [y]: n
Goodbye!
```

To delete users you should use the rmuser(8) utility. This will remove all existence of a user. It will remove any crontab(1) entries, their $HOME dir (if it is owned by the user), and their mail. Of course it will also remove their */etc/passwd* and */etc/group* entries. Next is an example of removing the user that was added above. Notice you are prompted for the name, and whether or not to remove the users home directory.

```
# rmuser
Enter login name for user to remove: testuser
Matching password entry:

testuser:$2a$07$ZWnBOsbqMJ.ducQBfsTKUe3PL97Ve1AHWJ0A4uLamniLNXLeYrEie:1002
:31::0:0:Test FAQ User:/home/testuser:/bin/ksh

Is this the entry you wish to remove? y
Remove user's home directory (/home/testuser)? y
Updating password file, updating databases, done.
Updating group file: done.
Removing user's home directory (/home/testuser): done.
```

Adding users via user(8)

These tools are less interactive than the adduser(8) command, which makes them easier to use in scripts.

The full set of tools is:

- group(8)
- groupadd(8)
- groupdel(8)
- groupinfo(8)
- groupmod(8)
- user(8)
- useradd(8)
- userdel(8)
- userinfo(8)
- usermod(8)

Actually adding users

Being that user(8) is not interactive, the easiest way to add users efficiently is to use the adduser(8) command. The actual command */usr/sbin/user* is just a frontend to the rest of the */usr/sbin/user** commands. Therefore, the following commands can be added by using **user add** or **useradd**, its your choice as to what you want, and doesn't change the use of the commands at all.

In this example, we are adding the same user with the same specifications as the user that was added above. useradd(8) is much easier to use if you know the default setting before adding a user. These settings are located in */etc/usermgmt.conf* and can be viewed by doing so:

```
$ user add -D
group           users
base_dir        /home
skel_dir        /etc/skel
shell           /bin/csh
inactive        0
expire          Null (unset)
range           1000..60000
```

The above settings are what will be set unless you specify different with command line options. For example, in our case, we want the user to go to the group **guest**, not **users**. One more little hurdle with adding users, is that passwords must be specified on the commandline. This is, the encrypted passwords, so you must first use the encrypt(1) utility to create the password. For example: OpenBSD's passwords by default use the Blowfish algorithm for 6 rounds. Here is an example line to create an encrypted password to specify to useradd(8).

```
$ encrypt -p -b 6
Enter string:
$2a$06$YOdOZM3.4m6MObBXjeZtBOWArqC2.uRJZXUkOghbieIvSWXVJRzlq
```

Now that we have our encrypted password, we are ready to add the user.

```
# user add -p '$2a$06$YOdOZM3.4m6MObBXjeZtBOWArqC2.uRJZXUkOghbieIvSWXVJRzlq' -u 1002
\
-s /bin/ksh -c "Test FAQ User" -m -g guest testuser
```

Note: Make sure to use ' ' (single quotes) around the password string, not " " (double quotes) as the shell will interpret these before

sending it to user(8). In addition to that, make sure you specify the **-m** option if you want the user's home directory created and the files from */etc/skel* copied over.

To see that the user was created correctly, we can use many different utilities. Below are a few commands you can use to quickly check that everything was created correctly.

```
$ ls -la /home
total 14
drwxr-xr-x   5 root      wheel   512 May 12 14:29 .
drwxr-xr-x  15 root      wheel   512 Apr 25 20:52 ..
drwxr-xr-x  24 ericj     wheel  2560 May 12 13:38 ericj
drwxr-xr-x   2 testuser  guest   512 May 12 14:28 testuser
$ id testuser
uid=1002(testuser) gid=31(guest) groups=31(guest)
$ finger testuser
Login: testuser                         Name: Test FAQ User
Directory: /home/testuser               Shell: /bin/ksh
Last login Sat Apr 22 16:05 (EDT) on ttyC2
No Mail.
No Plan.
```

In addition to these commands, user(8) provides its own utility to show user characteristics, called userinfo(8).

```
$ userinfo testuser
login   testuser
passwd  *
uid     1002
groups  guest
change  Wed Dec 31 19:00:00 1969
class
gecos   Test FAQ User
dir     /home/testuser
shell   /bin/ksh
expire  Wed Dec 31 19:00:00 1969
```

Removing users

To remove users with the user(8) hierarchy of commands, you will use userdel(8). This is a very simple, yet usable command. To remove the user created in the last example, simply:

```
# userdel -r testuser
```

Notice the **-r** option, which must be specified if you want the users home directory to be deleted as well. Alternatively, you can specify **-p** and not **-r** and this will lock the user's account, but not remove any information.

10.10 - How do I create an ftp-only account (not anonymous FTP!)?

There are a few ways to do this, but a very common way to do such is to add */usr/bin/false* into */etc/shells*. Then when you set a users shell to /usr/bin/false, they will not be able log in interactively, but will be able to use ftp capabilities. adduser(8) will give them a home dir by default of */home/<user>*. If this is what you desire it doesn't need to be changed, however you can set this to whatever directory you wish. You can force this user to only be able to see files in their home directory by adding their username to */etc/ftpchroot*. Using the **-A** option to ftpd(8), you can allow only ftpchroot logins!

10.11 - Setting up Quotas

Quotas are used to limit user's space that they have available to them on your disk drives. It can be very helpful in situations where you have limited resources. Quotas can be set by user and/or by group.

The first step to setting up quotas is to make sure that "`option QUOTA`" is in your Kernel Configuration. This option is in the GENERIC kernel. After this, you need to mark in `/etc/fstab` the filesystems which will have quotas enabled. The keywords `userquota` and `groupquota` should be used to mark each filesystem that you will be using quotas on. By default, the files `quota.user` and `quota.group` will be created at the root of that filesystem to hold the quota information. This default can be overridden by specifying the file name with the quota option in `/etc/fstab`, such as "`userquota=/var/quotas/quota.user`". Here is an example `/etc/fstab` that has one filesystem with userquotas enabled, and the quota file in a non-standard location:

```
/dev/wd0a / ffs rw,userquota=/var/quotas/quota.user 1 1
```

Now it's time to set the user's quotas. To do so you use the utility edquota(8). A simple use is just "`edquota <user>`". edquota(8) will use vi(1) to edit the quotas unless the environmental variable EDITOR is set to a different editor. For example:

```
# edquota ericj
```

This will give you output similar to this:

```
Quotas for user ericj:
/: blocks in use: 62, limits (soft = 0, hard = 0)
        inodes in use: 25, limits (soft = 0, hard = 0)
```

To add limits, edit it to give results like this:

```
Quotas for user ericj:
/: blocks in use: 62, limits (soft = 1000, hard = 1050)
        inodes in use: 25, limits (soft = 0, hard = 0)
```

Note that the quota allocation is in 1k blocks. In this case, the softlimit is set to 1000k, and the hardlimit is set to 1050k. A softlimit is a limit where the user is just warned when they cross it and have until their grace period is up to get their disk usage below their limit. Grace periods can be set by using the **-t** option on edquota(8). After the grace period is over the softlimit is handled as a hardlimit. This usually results in an allocation failure.

Now that the quotas are set, you need to turn the quotas on. To do this use quotaon(8). For example:

```
# quotaon -a
```

This will go through `/etc/fstab` to turn on the filesystems with quota options. Now that quotas are up and running, you can view them using quota(1). Using a command of "`quota <user>`" will give that user's information. When called with no arguments, the quota(1) command will give your quota statistics. For example:

```
# quota ericj
```

Will result in output similar to this:

```
Disk quotas for user ericj (uid 1001):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
              /      62    1000    1050              27       0       0
```

By default quotas set in `/etc/fstab` will be started on boot. To turn them off use

```
# quotaoff -a
```

10.12 - Setting up KerberosV Clients and Servers

OpenBSD includes KerberosV as a pre-installed component of the default system.

For more information on KerberosV, from your OpenBSD system, use the command:

```
 # info heimdal
```

10.13 - Setting up Anonymous FTP Services

Anonymous FTP allows users without accounts to access files on your computer via the File Transfer Protocol. This will give an overview of setting up the anonymous FTP server, and its logging, etc.

Adding the FTP account

To start off, you need to have an account on your system of "ftp". This account shouldn't have a usable password. Here we will set the login directory to /home/ftp, but you can put it wherever you want. When using anonymous ftp, the ftp daemon will chroot itself to the home directory of the 'ftp' user. To read up more on that, read the ftp(8) and chroot(2) man pages. Here is an example of adding the *ftp* user. I will do this using adduser(8). We also need to add /usr/bin/false to our */etc/shells*, this is the "shell" that we will be giving to the ftp user. This won't allow them to login, even though we will give them an empty password. To do this you can simply *echo /usr/bin/false >> /etc/shells*. Also if you wish for that shell to show up during the adduser questions, you need to modify */etc/adduser.conf*.

```
# adduser
Use option ``-silent'' if you don't want to see all warnings and questions.

Reading /etc/shells
Reading /etc/login.conf
Check /etc/master.passwd
Check /etc/group

Ok, let's go.
Don't worry about mistakes. I will give you the chance later to correct any input.
Enter username []: ftp
Enter full name []: anonymous ftp
Enter shell csh false ksh nologin sh tcsh zsh [sh]: false
Uid [1002]: Enter
Login group ftp [ftp]: Enter
Login group is ``ftp''. Invite ftp into other groups: guest no
[no]: no
Login class auth-defaults auth-ftp-defaults daemon default staff
[default]: Enter
Enter password []: Enter
Set the password so that user cannot logon? (y/n) [n]: y

Name:        ftp
Password:    ****
Fullname:    anonymous ftp
Uid:         1002
Gid:         1002 (ftp)
Groups:      ftp
Login Class: default
HOME:        /home/ftp
Shell:       /usr/bin/false
OK? (y/n) [y]: y
Added user ``ftp''
Copy files from /etc/skel to /home/ftp
Add another user? (y/n) [y]: n
Goodbye!
```

Directory Setup

Along with the user, this created the directory */home/ftp*. This is what we want, but there are some changes that we will have to make to get it ready for anonymous ftp. Again these changes are explained in the ftp(8) man page.

You **do not** need to make a /home/ftp/usr or /home/ftp/bin directory.

- */home/ftp* - This is the main directory. It should be owned by root and have permissions of 555.
- */home/ftp/etc* - This is entirely optional and not recommended, as it only serves to give out information on users which exist on your box. If you want your anonymous ftp directory to appear to have real users attached to your files, you should copy /etc/pwd.db and /etc/group to this directory. This directory should be mode 511, and the two files should be mode 444. These are used to give owner names as opposed to numbers. There are no passwords stored in pwd.db, they are all in spwd.db, so don't copy that over.
- */home/ftp/pub* - This is a standard directory to place files in which you wish to share. This directory should also be mode 555.

Note that all these directories should be owned by "root". Here is a listing of what the directories should look like after their creation.

```
# pwd
/home
# ls -laR ftp
total 5
dr-xr-xr-x  5 root  ftp    512 Jul  6 11:33 .
drwxr-xr-x  7 root  wheel  512 Jul  6 10:58 ..
dr-x--x--x  2 root  ftp    512 Jul  6 11:34 etc
dr-xr-xr-x  2 root  ftp    512 Jul  6 11:33 pub

ftp/etc:
total 43
dr-x--x--x  2 root  ftp    512 Jul  6 11:34 .
dr-xr-xr-x  5 root  ftp    512 Jul  6 11:33 ..
```

```
-r--r--r--  1 root  ftp    316 Jul  6 11:34 group
-r--r--r--  1 root  ftp  40960 Jul  6 11:34 pwd.db

ftp/pub:
total 2
dr-xr-xr-x  2 root  ftp  512 Jul  6 11:33 .
dr-xr-xr-x  5 root  ftp  512 Jul  6 11:33 ..
```

Starting up the server and logging

With ftpd you can choose to either run it from inetd or the rc scripts can kick it off. These examples will show our daemon being started from inetd.conf. First we must become familiar with some of the options to ftpd. The default line from */etc/inetd.conf* is:

```
ftp             stream  tcp     nowait  root    /usr/libexec/ftpd       ftpd -US
```

Here ftpd is invoked with *-US*. This will log anonymous connections to */var/log/ftpd* and concurrent sessions to */var/run/utmp*. That will allow for these sessions to be seen via who(1). For some, you might want to run only an anonymous server, and disallow ftp for users. To do so you should invoke ftpd with the *-A* option. Here is a line that starts ftpd up for anonymous connections only. It also uses *-ll* which logs each connection to syslog, along with the get, retrieve, etc, ftp commands.

```
ftp             stream  tcp     nowait  root    /usr/libexec/tcpd       ftpd -llUSA
```

Note - For people using HIGH traffic ftp servers, you might want to not invoke ftpd from inetd.conf. The best option is to comment the ftpd line from inetd.conf and start ftpd from rc.conf along with the *-D* option. This will start ftpd as a daemon, and has much less overhead as starting it from inetd. Here is an example line to start it from rc.conf.

```
ftpd_flags="-DllUSA"            # for non-inetd use: ftpd_flags="-D"
```

This of course only works if you have ftpd taken out of */etc/inetd.conf* and made inetd re-read its configuration file.

Other relevant files

- */etc/ftpwelcome* - This holds the Welcome message for people once they have connected to your ftp server.
- */etc/motd* - This holds the message for people once they have successfully logged into your ftp server.
- *.message* - This file can be placed in any directory. It will be shown once a user enters that directory.

10.14 - Confining users to their home dir's in ftpd(8)

OpenBSD's ftpd(8) is setup by default to be able to handle this very easily. This is accomplished via the file **/etc/ftpchroot**. Since users cannot always be trusted, it might be necessary to restrain them to their home directories. This behavior is NOT on by default. Here is an example of what the default behavior is like.

```
$ ftp localhost
Connected to localhost.
220 oshibana FTP server (Version 6.4/OpenBSD) ready.
Name (localhost:ericj): ericj
331 Password required for ericj.
Password: *********
230- OpenBSD 3.4 (GENERIC) #18: Wed Sep 17 03:34:47 MDT 2003
230-
230- Welcome to OpenBSD: The proactively secure Unix-like operating system.
230-
230- Please use the sendbug(1) utility to report bugs in the system.
230- Before reporting a bug, please try to reproduce it with the latest
230- version of the code.  With bug reports, please try to ensure that
230- enough information to reproduce the problem is enclosed, and if a
230- known fix for it exists, include that as well.
230-
230 User ericj logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /
250 CWD command successful.
ftp> ls
227 Entering Passive Mode (127,0,0,1,60,7)
150 Opening ASCII mode data connection for 'file list'.
altroot
```

```
bin
dev
etc
home
mnt
root
sbin
stand
tmp
usr
var
bsd
sys
boot
226 Transfer complete.
ftp> quit
221 Goodbye.
```

As you can see here, access is granted to the whole server. In a perfect world this is ok, where all users can be trusted, but this isn't so. To limit a user, simply add their name to the file **/etc/ftpchroot**. Here is an example showing user "ericj" being restricted.

```
$ cat /etc/ftpchroot
#       $ OpenBSD: ftpchroot,v 1.3 1996/07/18 12:12:47 deraadt Exp $
#
# list of users (one per line) given ftp access to a chrooted area.
# read by ftpd(8).
ericj
```

This is enough to keep the user "ericj" from escaping from his own directory. As you can see in the next example. The / directory has suddenly changed to his home dir!

```
$ ftp localhost
Connected to localhost.
220 oshibana FTP server (Version 6.4/OpenBSD) ready.
Name (localhost:ericj): ericj
331 Password required for ericj.
Password: *********
230 User ericj logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /
250 CWD command successful.
ftp> ls
227 Entering Passive Mode (127,0,0,1,92,171)
150 Opening ASCII mode data connection for 'file list'.
.login
.mailrc
.profile
.rhosts
.ssh
.cshrc
work
mail
src
226 Transfer complete.
ftp> quit
221 Goodbye.
```

10.15 - Applying patches in OpenBSD

The OpenBSD source tree is constantly changing and improving, along with this fixes to common problems are often made and patches released to the public. These patches appear on the errata web page located at http://www.openbsd.org/errata.html, and are separated into categories. These categories correspond to patches that should be applied to different architectures or architecture independent patches.

Note, however, that patches aren't made for new additions to OpenBSD, and are only done for important reliability fixes or security problems that should be addressed right away, although the choice to do so is, as always, up to the administrator.

For the examples I will be patching talkd(8) with a security fix from the patch obtained from errata.html.

How are these patches different from what I would find in the CVS tree?

All patches posted at http://www.openbsd.org/errata.html are patches directly against the latest release's source tree. Patches against the latest CVS tree might also include other changes that wouldn't be wanted on a release system.

Getting your system ready to be patched.

Patches for the OpenBSD Operating System are distributed as diffs, which are text files that hold differences to the original source code. They are **NOT** distributed in binary form. This means that to patch your system you must have the source code from the **RELEASE** version of OpenBSD readily available. This does not mean that you must have ALL source code to the OpenBSD operating system to patch your system, but must have all code for the program which you are patching. For instance, if you are patching the kernel you must have all source for the kernel on hand.

cvs(1) is a very handy tool that can be used to grab only the source that you need via any of the anonymous cvs servers located around the world. You can get a listing of these servers at http://www.openbsd.org/anoncvs.html.

To retrieve the source code for talkd(8) from *3.4-release* using cvs(1), you would use the following lines:

```
$ export CVSROOT=anoncvs@anoncvs5.usa.openbsd.org:/cvs
$ cvs co -rOPENBSD_3_4_BASE src/libexec/talkd/
cvs server: Updating src/libexec/talkd
U src/libexec/talkd/announce.c
U src/libexec/talkd/talkd.c
U src/libexec/talkd/talkd.h
```

To find the CVS path to the code that you need, you can find this in the patch on the *Index:* line. In this case, the CVS path was *src/libexec/talkd/*. Always check out the revision of OPENBSD_version_number_BASE. Without "_BASE" you will be checking out the stable branch, which might contain other changes that will interfere. If you are already tracking the patch branch, the patches should already be in that source, however you should always check and make sure. You can always look at http://www.openbsd.org/plus.html to see which patches have been applied to the patch branch. If the patches haven't been applied yet, you will need to grab the latest release source using the commands above.

Also, for those users that bought official OpenBSD CDs, you can get the source code directly off of the CD. Refer to the CD insert on how to extract the source from the CD. In which case you won't need to obtain the source via anoncvs.

```
    Apply by doing:
            cd /usr/src
            patch -p0 < 026_talkd.patch
            cd libexec/talkd
            make obj && make depend && make && make install

    Index: libexec/talkd/announce.c <------ Path to sources
    ===================================================================
    RCS file: /cvs/src/libexec/talkd/announce.c,v
    retrieving revision 1.8
    retrieving revision 1.9
    diff -u -r1.8 -r1.9
    --- libexec/talkd/announce.c    1998/08/18 03:42:10     1.8
    +++ libexec/talkd/announce.c    2000/07/06 00:01:45     1.9
    @@ -160,6 +160,6 @@
                    *(bptr++) = '\n';
            }
            *bptr = '\0';
    -       fprintf(tf, big_buf);
    +       fprintf(tf, "%s", big_buf);
            fflush(tf);
     }
```

Once you've obtained the proper sources, you can obtain the patch and place it in *src/*

Applying Patches

```
$ cd /usr/src
$ patch -p0</path/to/026_talkd.patch
Hmm...  Looks like a unified diff to me...
The text leading up to this was:
```

```
--------------------------
|Apply by doing:
|       cd /usr/src
|       patch -p0 < 026_talkd.patch
|       cd libexec/talkd
|       make obj && make depend && make && make install
|
|Index: libexec/talkd/announce.c
|===================================================================
|RCS file: /cvs/src/libexec/talkd/announce.c,v
|retrieving revision 1.8
|retrieving revision 1.9
|diff -u -r1.8 -r1.9
|--- libexec/talkd/announce.c   1998/08/18 03:42:10     1.8
|+++ libexec/talkd/announce.c   2000/07/06 00:01:45     1.9
--------------------------
Patching file libexec/talkd/announce.c using Plan A...
Hunk #1 succeeded at 160. <------------ Patch Succeeded
done
$ cd /usr/src/libexec/talkd/
$ ls
CVS             announce.c      print.c         table.c         talkd.c
Makefile        announce.c.orig process.c       talkd.8         talkd.h
$ make obj && make depend && make
making /home/ericj/lsrc/src/libexec/talkd/obj
mkdep -a /home/ericj/lsrc/src/libexec/talkd/talkd.c /home/ericj/lsrc/sr
c/libexec/talkd/announce.c /home/ericj/lsrc/src/libexec/talkd/process.c
/home/ericj/lsrc/src/libexec/talkd/table.c /home/ericj/lsrc/src/libexec
/talkd/print.c
cc -O2     -c /home/ericj/lsrc/src/libexec/talkd/talkd.c
cc -O2     -c /home/ericj/lsrc/src/libexec/talkd/announce.c
cc -O2     -c /home/ericj/lsrc/src/libexec/talkd/process.c
cc -O2     -c /home/ericj/lsrc/src/libexec/talkd/table.c
cc -O2     -c /home/ericj/lsrc/src/libexec/talkd/print.c
cc   -o ntalkd talkd.o announce.o process.o table.o print.o
nroff -Tascii -mandoc /home/ericj/lsrc/src/libexec/talkd/talkd.8 > talk
d.cat8
$ sudo make install
install -c -s -o root -g bin  -m 555 ntalkd /usr/libexec
install -c -o root -g bin -m 444 talkd.cat8 /usr/share/man/cat8/talkd.0
/usr/share/man/cat8/ntalkd.0 -> /usr/share/man/cat8/talkd.0
```

Once you have done that, you should restart that service.

10.16 - Tell me about this chroot() Apache?

In OpenBSD, the Apache httpd(8) server has been chroot(2)ed by default. While this is a tremendous boost to security, it can create issues, if you are not prepared.

What is a chroot?

A chroot(2)ed application is locked into a particular directory and unable to wander around the rest of the directory tree, and sees that directory as its "/" (root) directory. In the case of httpd(8), the program starts, opens its log files, binds to its TCP ports (though, it doesn't accept data yet), and reads its configuration. Next, it locks itself into */var/www* and drops privileges, then starts to accept requests. This means all files served and used by Apache must be in the */var/www* directory. This helps security tremendously -- should there be a security issue with Apache, the damage will be confined to a single directory with only "read only" permissions and no resources to cause mischief with.

What does this mean to the user?

Put bluntly, chroot(2)ing Apache is something new, and many older applications and system configurations will not work as before.

- **Historic file system layouts:** Servers upgraded from older versions of OpenBSD may have web files located in user's directories, which clearly won't work in a chroot(2)ed environment, as httpd(8) can't reach the */home* directory. Administrators may also discover their existing /var/www partition is too small to hold all web files. Your options are to restructure or do not use the chroot(2) feature. You can, of course, use symbolic links in the user's home directories pointing to subdirectories in */var/www*, but you can NOT use links in */var/www* pointing to other part of the file system -- that is prevented from working by the chroot(2)ing. Note that if you want your users to have chroot(2)ed FTP access, this will not

work, as the FTP chroot will (again) prevent you from accessing the targets of the symbolic links. A solution to this is to not use */home* as your home directories for these users, rather use something similar to */var/www/home*.

- **Log Rotation:** Normally, logs are rotated by renaming the old files, then sending httpd(8) a SIGUSR1 signal to cause Apache to close its old log files and open new ones. This is no longer possible, as httpd(8) has no ability to open its own log files once privileges are dropped. httpd(8) must be stopped and restarted:

```
# apachectl stop && apachectl start
```

  There are also other strategies available, including logging to a pipe(2), and using an external log rotator at the other end of the pipe(2).
- **Existing Apache modules:** Virtually all will load, however some may not work properly in chroot(2), and many have issues on "`apachectl restart`", generating an error, which causes httpd(8) to exit.
- **Existing CGIs:** Most will NOT work as is. They may need programs or libraries outside */var/www*. Some can be fixed by compiling so they are statically linked (not needing libraries in other directories), most may be fixed by populating the */var/www* directory with the files required by the application, though this is non-trivial and requires considerable programming knowledge -- most users will find it easier to just disable the chroot(2) feature until they are updated.

In some cases, the application or configuration can be altered to run within the chroot. In other cases, you will simply have to disable this feature using the `-u` option for httpd(8) in */etc/rc.conf*.

10.17 - I don't like the standard root shell!

The default shell for *root* on OpenBSD is csh, due primarily to tradition. There is no requirement that OpenBSD have csh(1) for the root login (though keep reading before changing it).

Some users who come from other Unix-like operating systems find csh(1) unfamiliar, and ask if and how they can change it. There are a few options:

- **Don't login as *root!*** Between su and sudo, there should be few reasons for users to log in as *root* for most applications after initial setup.
- **Invoke your favorite shell after login:** If you like ksh(1) or any other shell, just invoke it from the default shell.
- **Change the root shell:** This can be done using chsh or vipw.

A traditional Unix guideline is to only use statically compiled shells for root, because if your system comes up in single user mode, non-root partitions won't be mounted and dynamically linked shells won't be able to access libraries located in the `/usr` partition. This isn't actually a significant issue for OpenBSD, as the system will prompt you for a shell when it comes up in single user mode, and the default is sh. The three standard shells in OpenBSD (csh, sh and ksh) are all statically linked, and thus usable in single user mode.

It is sometimes said that one should never change the root shell, though there is no reason not to in OpenBSD. But again, this shouldn't be an issue -- just don't log in as root.

10.18 - What else can I do with *ksh*?

In OpenBSD, ksh is pdksh, the Public Domain Korn Shell, and is the same binary as sh.

Users comfortable with *bash*, often used on Linux systems, will probably find ksh very familiar. Ksh(1) provides most of the commonly used features in *bash*, including tab completion, command line editing and history via the arrow keys, and CTRL-A/CTRL-E to jump to beginning/end of the command line. If other features of *bash* are desired, *bash* itself can be loaded via either ports or packages.

The command prompt of *ksh* can easily be changed to something providing more information than the default "$ " by setting the `PS1` variable. For example, inserting the following line:

```
export PS1='$PWD $ '
```

in your `/etc/profile` produces the following command prompt:

```
/home/nick $
```

See the file `/etc/ksh.kshrc`, which includes many useful features and examples, and may be invoked in your user's `.profile`.

[FAQ Index] [To Section 9 - Migrating from Linux] [To Section 11 - Performance Tuning]

---

www@openbsd.org

$OpenBSD: faq10.html,v 1.96 2004/01/01 04:13:34 nick Exp $

OpenBSD

11.0 - Performance Tuning

Inhaltsverzeichnis

Wenn du einen vielbesuchten Server, ein Gateway oder eine Firewall administrierst, möchtest oder musst du vielleicht einige der standardmässigen Parameter anpassen, um eine optimale Performance zu erhalten. Die options(4) man page berichtet über die angebotenen Kerneloptionen. Diese Optionen werden in der Kernel-Konfigurationsdatei plaziert, bevor du einen eigenen Kernel kompilierst. Mehr Details dazu erhälst du in der FAQ 5.

11.1 - Netzwerk

Ein Parameter, der bei einem besonders belasteten Server, Gateway oder Firewall vielleicht angepasst werden muss, ist NMBCLUSTERS. Er kontrolliert die Grösse der kernel mbuf cluster map. Wenn du Meldungen wie "mb_map full" auf deinem Computer bekommst, musst du die Werte für diesen Parameter vergrössern. Wenn ohne ersichtlichen Grund der Traffic auf einem Netzwerk aufhört, kann das ebenfalls ein Zeichen für zu kleine NMBCLUSTERS sein. Ein sinnvoller Wert beim i386 port mit mindestens 100Mbps ethernet Interfaces (egal wieviele davon die Maschine hat) ist 8192.

option NMBCLUSTERS=8192

Die Standard-Anzahl von NMBCLUSTERS variiert von Plattform zu Plattform, und reicht von 256 bis 2048. Sie werden in einer plattform-abhängigen Header-Datei gesetzt, es sei denn, sie werden von einer "option"-Zeile in einer Kernel-Konfigurations-Datei überschrieben.

11.2 - Festplatten I/O

Festplatten I/O 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

11.2.1 - CCD

Die erste Option ist die Benutzung des ccd(4), des Concatenated Disk Driver. Das 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 es 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 die 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 Performance 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 Disc weg ist.

11.2.2 - RAID

Eine weitere Lösung ist raid(4), wofür du raidctl(8) nutzen musst, um deine RAID Geräte zu kontrollieren. OpenBSD's RAID basiert auf Greg Oster's 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) man pages für die kompletten Details. Es gibt dafür viele Optionen und mögliche Konfiguationen, und ein detaillierter Überblick sprengt den Rahmen dieses Dokumentes.

11.2.3 - Filesystem Buffer

Fileserver, die noch Speicher überhaben, können die BUFCACHEPERCENT erhöhen. Das heisst, welcher Prozentsatz deines RAM als "file system buffer" (Dateisystem-Puffer) genutzt werden soll. Diese Option wird vielleicht geändert, wenn der Unified Buffer Cache komplett und ein Teil von OpenBSD ist. In der Zwischenzeit solltest du eine Zeile wie diese zu deiner Kernel-Konfiguration hinzufügen, um BUFCACHEPERCENT zu erhöhen:

option BUFCACHEPERCENT=30

Natürlich kannst du ihn auch auf 5 Prozent setzen (dem Standard) oder auch so hoch wie 50 Prozent (oder auch mehr.)

11.2.4 - Soft Updates

Ein weiteres Tool zum Erhöhen der Systemgeschwindigkeit sind Soft Updates. Eine der langsamsten Operationen im traditionellen BSD Dateisysteme 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.

11.2.5 - Grösse des namei() cache

Hinweis: Vorher hat die options(4) manual page 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. Standardmässig hat dieser Cache `NPROC * (80 + NPROC / 8)` Einträge. `NPROC` ist auf `20 + 16 * MAXUSERS` gesetzt; in der config(8) manual page steht eine Erklärugn der `maxusers` Kernel-Konfigurations-Parameter. Ein sinnvoller Weg zum Herausfinden der passenden Grösse des Cache wäre, eine große Anzahl von namei() cache misses vorrausgesetzt, die man mit einem Tool wie systat(1) messen könnte, wäre eine Untersuchung des momentanen berechneten Wertes mittels sysctl(8), (das diesen Parameter "`kern.maxvnodes`" nennt) und diesen Wert zu vergössern, 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 gestgestellt wurde, kannst du ihn für die nächsten Systemstarts mit sysctl.conf(5) setzen.

11.4 - 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 Performance 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 Performance deiner Anwendungen hängt aber auch stark von deiner Hardware ab. Für viele Leute ist das Preis-Leistungs-Verhältnis eines brandneuen PC 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 Qualitält verlieren!

Es gibt ein paar Dinge, die dir helfen können, die maximale Performance aus deiner Hardware zu holen:

- Benutze mehrere Festplatten.

  Statt nur eine 20GB Platte zu kaufen, kaufe mehrere 9GB 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-IO-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 baust, und mehr als 20 GB Plattenplatz 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 I/O Performance 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 trift 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 Performance 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 Netzwerk-Karte 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 Sound-Karten oder solche für serielle Ports.
- Vermeide billige PCI Netzwerk-Karten.

  OpenBSD unterstützt eine ganze Menge von billigen PCI Netzwerk-Karten. 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-Netzwerk-Karte besser dran. Unglücklicherweise sind einige Serien von teuren Marken-Herstellern (wie die 3com XL Serie) nicht besser als die billigen Karten. Ein echter Favorit unter den 10/100Mbps Netzwerk-Karten ist dagegen die Intel EtherExpress PRO/100.

11.5 - 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 asynchrones mounting abgelehnt und ist nicht standardmässig aktiviert (wie in manchen anderen Unixen) ? Wäre das nicht ein einfacherer, und daher auch sichererer Weg, die Performance 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 man page 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.

11.6 - Tunen deiner Monitorauflösung unter XFree86

Es ist durchaus mit vielen multi-Sync-Monitoren möglich, einen X Server in einer azeptablen 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 page)

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 der 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 Limits 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 Dell's Website herausgefunden, das 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 dot clock (Bandbreite) 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/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 CTRL-ALT-KEYPAD MINUS oder CTRL-ALT-KEYPAD PLUS kannst du zwischen den hier aufgeführten Auflösungen hin- und herschalten. Gemäss der Angaben oben wird X versuchen im 32-Bit-Modus (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 ist, zu breit oder zu klein, 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 fixes 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 BW 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 in 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, dekodiere die Monitor-Spezifikationen, die du in xf86config/XF86Setup eingegeben hast (Hast du eigentlich die echten Spezifikationen für deinen Monitor eingegeben, oder die generischen benutzt?) und passe die existierenden ModeLines an.

[Back to Main Index] [To Section 10.0 - System Administration] [To Section 12.0 - Für fortgeschrittene Users]


www@openbsd.org

Originally [OpenBSD: faq11.html,v 1.29]
$Translation: faq11.html,v 1.5 2003/01/17 14:02:59 jufi Exp $
$OpenBSD: faq11.html,v 1.5 2003/01/17 18:47:48 jufi Exp $

12 - Für fortgeschrittene User

Inhaltsverzeichnis

Mit *fortgeschrittenen Usern* meinen wir Leute, die ein Unix-System administrieren können und wissen, wie es funktioniert. Wenn du den Anweisungen in dieser Sektion folgst, ohne zu wissen, was du tust, kannst du deinem System Schaden zufügen!

12.1 - DMA Zugriff für IDE Festplatten erzwingen

Dem PCI IDE Code ist dein Chipset vielleicht nicht bekannt. Wenn das so ist, wirst du eine Meldung beim Booten bekommen, die etwa so aussieht:

```
pciide0: DMA, (unused)
```

Wenn du so eine Meldung erhältst, kannst du versuchen, den DMA Modus mittels 'flags 0x0001' in deinem pciide Eintrag in deiner Kernel-Konfigurationsdatei zu erzwingen. Das würde ungefähr so aussehen:

```
pciide* at pci ? dev ? function ? flags 0x0001
```

Wenn man das gemacht hat, wird der pciide Code versuchen, den DMA Modus zu benutzen, unabhängig davon, ob er weiss, wie er das mit deinem Chipsatz anstellen soll. Wenn das funktioniert, und es dein System durch 'fsck' und die restliche Startsequenz schafft, ist es wahrscheinlich, dass das Ganze weiter so funktioniert. Wenn es nicht klappt, und das System hängt oder in 'panic' verfällt, kannst du den DMA-Modus schlicht und einfach nicht benutzten (solange keine Unterstützung für deinen Chipsatz hinzugefügt wurde, natürlich). Wenn du die volle Dokumentation für deinen Chipsatz findest, ist das ein guter Anfang für volle Unterstützung im PCI-IDE Code des Kernels. Du kannst auf der Website des Herstellers nachsehen, oder sie auch anrufen. Wenn dein PCI-IDE Controller Teil deines Motherboards ist, finde heraus, wer der Hersteller des Chipsatzes ist, und wende dich an sie!

Wenn du diese Meldung beim Booten bekommst, weisst du dass dein DMA aktiviert wurde:

```
cd0(pciide0:1:0): using PIO mode 3, DMA mode 1
```

Das bedeutet, dass pciide0, channel 1, drive 0 (was hier ein ATAPI CD-ROM ist) DMA Datentransfers benutzt.

[FAQ Index] [Zur Sektion 11 - Performance Tuning] [Zur Sektion 13 - IPsec]


www@openbsd.org

Originally [OpenBSD: faq12.html,v 1.35]
$Translation: faq12.html,v 1.15 2003/07/13 13:53:13 jufi Exp $
$OpenBSD: faq12.html,v 1.15 2003/07/13 14:04:21 jufi Exp $

OpenBSD

14.0 - Disk Setup

Inhaltsverzeichnis

Benutzung von OpenBSD's disklabel

Inhaltsangabe

Was ist disklabel(8)?

Lese zunächst die disklabel(8) man page.

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 dein Dateisystem, wie z.B. die 'drive geometry' 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) man page.

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 BS 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 während der OpenBSD Installation

Einer der Hauptteile der OpenBSD Installation ist das erste Erzeugen der 'labels'. Das kommt (für i386 Benutzer) direkt nach der Benutzung von fdisk(1). Während der Installation benutzt du 'disklabel' um deine separaten 'label' zu erzeugen, die deine separaten 'mountpoints' enthalten. Während der Installation kannst du mittels disklabel(8) deine Mountpoints 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 erzeugt, ist das Label 'a'. Das sollte das Label sein, auf dem dann '/' gemountet wird. Die empfohlenen Partitionen und ihren Grössen kannst du dir auf faq4.3 ansehen. Für Server wird empfohlen zumindest diese 'label' 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 deine Swap-Partition 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 volle/komplette Installation gemacht wird.

```
Wenn die Festplatte mit anderen Betriebssystemen geteilt wird, sollten diese BS einen
BIOS Partitionseintrag haben, der den kompletten Platz umfasst, den sie beanspruchen.
Aus Sicherheitsgründen solltest du auch sicherstellen, dass alle OpenBSD Dateisysteme
innerhalb
des 'offset' und der Grösse sind, die in der 'A6' BIOS Partitionstabelle angegeben
sind.
(Standardmässig wird der disklabel Editor das versuchen 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), dann belasse es jetzt erstmal bei einer root
und einer
```

```
swap Partition.

# 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 64MB 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 nächsten Zylinder runden. Im obigen Beispiel kann man auch sehen, dass Disklabel annimmt, dass Label 'b' eine Swap-Partition 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 Ganze 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 automatisch für dich errrechnet 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
```

HINWEIS - Für User mit grossen Festplatten: Wenn dein BIOS nicht in der Lage ist, eine so grosse Festplatte zu unterstützen, kann OpenBSD das auch nicht. Ansonsten sollte OpenBSD keine Schwierigkeiten mit deiner Festplatte haben. Wenn dein BIOS das nicht kann, könntest du es mit Maxtor EZ-Drive oder einem anderen 'Overlay' Produkt ausprobieren.

Gebräuchliche Benutzung 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. 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 mit angeben:

```
# disklabel -E wd0
```

Das wird dich einfach an einen Prompt bringen, und zwar den selben, den du schon während der OpenBSD Installation benutzt hast. Das wahrscheinlich wichtigste Kommando ist '?'. Das erzeugt nämlich eine Liste mit möglichen Optionen für Disklabel. Mit Hilfe von 'M' kannst du dir sogar die gesamte disklabel(8) man page ansehen. Von diesem Prompt aus wirst du dein gesamtes Hinzufügen, Löschen und Ändern der Partitionen vornehmen. Zusätzliche Informationen gibt es in der disklabel(8) man page.

14.2 - Benutzung von OpenBSD's fdisk

Um sicher zu sein, prüfe zuerst die fdisk man page: fdisk(8)

Fdisk ist ein Programm, dass bei der Pflege und 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 setzen. Dieses Programm wird für gewöhnlich im 'Single User Mode' benutzt werden (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:

```
# fdisk fd0
```

Was dann eine ähnliche Ausgabe wie diese hier erzeugt:

```
         Disk: fd0       geometry: 80/2/18 [2880 sectors]
         Offset: 0       Signatures: 0xAA55,0x0
                    Starting        Ending
          #: id  cyl  hd sec -  cyl  hd sec [     start -       size]
-------------------------------------------------------------------------
         *0: A6    0   0   1 -   79   1  18 [         0 -       2880] OpenBSD
          1: 00    0   0   0 -    0   0   0 [         0 -          0] unused
          2: A7    0   0   2 -   79   1  18 [         1 -       2879] NEXTSTEP
          3: 00    0   0   0 -    0   0   0 [         0 -          0] unused
```

In diesem Beispiel betrachten wir die Ausgabe der Floppy Disk. Wir können die OpenBSD Partition (A6) und ihre Grösse sehen. Der * sagt uns, dass die OpenBSD Partition bootbar ist.

Im vorigen Beispiel haben wir uns die Informationen nur angesehen. Was aber, wenn wir unsere Partitionstabelle verändern wollen? Nunja, 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)
        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 mode' versteht.
- **reinit** Initialisiere die momentane, im Speicher befindliche Kopie des Boot-Blocks.
- **disk** Zeige die momentane Platten-Geometrie an, die fdisk herausgefunden hat. Du bekommst eine Möglichkeit sie zu ändern, wenn du willst.
- **edit** Ändere eine ausgewählte Platten-Geometrie in Kopie des momentanen Bootblocks. Das geschieht entweder im BIOS geometry mode oder in sector offsets und Grössen.
- **flag** Den jetzigen Partitionstabelleneintrag bootbar machen. Nur ein Eintrag zur Zeit kann bootbar sein. Wenn du von einer extended Partition booten willst, musst du auch den entsprechenden Eintrag als bootbar markieren.
- **update** Bringe den Maschinencode in der Speicherkopie des momentanen Bootblocks auf aktuellen Stand.
- **select** Wähle und lade den Boot block, auf den der Eintrag der erweiterten Partitionstabelle im momentanen Boot-Block zeigt.
- **print** Gebe die momentan im RAM befindlichen und gewählte Kopie des Boot Blocks und seinen MBR auf dem Bildschirm aus.
- **write** Schreibe die RAM-Version des Boot Blocks 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 zurücke 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 zurücke 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 - Zusätzliche Festplatten unter OpenBSD installieren

Nun, nachdem du deine Festplatte **SAUBER** eingebaut hast, musst du fdisk(8) (*nur bei i386*) und auch disklabel(8), um deine Festplatte unter OpenBSD benutzen zu können.

Die i386 Leute starten mit fdisk. Andere Architekturen ignorieren das einfach.

```
# fdisk -i sd2
```

Das wird die "echte" Partitionstabelle der Festplatte für eine ausschliessliche Benutzung mit OpenBSD initialisieren. Als nächstes musst du einen disklabel dafür erzeugen. Das wird wohl etwas verwirrend sein.

```
# 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. Für den normalen Betrieb sollte die fsize immer 1024 sein, bsize immer 8192 und cpg immer 16. Der Fstype 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. Rechne also 6185088/3000 (benutze bc(1)). Du erhältst 2061. um jetzt Partitionsgrössen für a, d, e, f, g, ... zu erhalten, rechne einfach X*2061 um X Megabytes Platz auf dieser Partition zu erhalten. Der offset für deine erste Partition sollte derselbe sein, wie von "sectors/track" vorher in disklabel's 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 das ganze Ding nur zum Ablegen von Webpages oder einem Homedirectory oder etwas anderem nutzen willst, nimm einfach die gesamte Grösse der Platte und ziehe die Sektoren pro Track 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 Gigabyte!) Unglücklicherweise benutzt der -E Modus die BIOS Platten-Geometrie 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.

```
bsd# newfs wd1a
```

Oder wie deine Festplatte auch immer nach dem OpenBSD Plattennummerierungs-Schema heissen mag. (Guck einfach in der Ausgabe von dmesg(1) nach, da steht es drin.)

Nun überleg dir, wohin du deine gerade neu geschaffene Partition mounten willst. Sagen wir einfach mal /u. Daher erzeuge zunächst einmal /u mit 'mkdir'. Dann mounte sie.

```
mount /dev/wd1a /u
```

Zuletzt musst du sie noch zu /etc/fstab hinzufügen.

```
/dev/wd1a /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 so an, dass jetzt die /usr/local Partition auf /dev/wd1a zu finden ist. (Deine frisch formatierte Partition.) Beispiel:

/dev/wd1a /usr/local ffs rw 1 1

Reboote in den single user Mode.. **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 reboote das System, und voila! Die Dateien sind da!

14.4 - Wie man in eine Datei swapt

(Hinweis: wenn du deshalb 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 csh's unlimit(1), oder auch mit sh's ulimit(1).)

Nach der Veröffentlichung von OpenBSD 2.5 kam swapctl(8) heraus, was den Umgang mit swap devices viel leichter machte. Falls du auf einem OpenBSD 2.5 System arbeitest, tausche swapctl mit swapon, und benutze pstat -s, um deine "swap devices" aufzulisten. Das Swappen in eine Datei benötigt keinen selber gebauten Kernel, obwohl man das natürlich trotzdem machen kann, diese FAQ zeigt dir beide Wege den Swap zu erhöhen.

In eine Datei swappen.

In eine Datei zu swappen ist der einfachste und schnellste Weg, um zusätzlichen Swap zu bekommen. Das gilt aber nicht für Benutzer von Soft Updates (was ja standardmässig nicht aktiviert 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:

```
ericj@oshibana> 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. 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 'disklabel' 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 würden.

Der erste Schritt um eine Datei als Swap-Bereich zu nutzen, ist, die Datei zu erzeugen. Am besten macht man das mit Hilfe von dd(1) Hier ist ein Beispiel, das eine Datei */var/swap* mit der Grösse von 32MB erzeugt.

```
ericj@oshibana> 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 diese Datei richten. Benutze einfach das folgende Kommando, um das Swappen auf diese Datei zu lenken

```
ericj@oshibana> sudo chmod 600 /var/swap
ericj@oshibana> sudo swapctl -a /var/swap
```

Jetzt müssen wir noch prüfen, ob sie auch korrekt zu unserer Liste der Swap-Geräte hinzugefügt wurde.

```
ericj@oshibana> 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, wo 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-Gerät eben nicht konfiguriert.

```
ericj@oshibana> cat /etc/fstab
```

```
/dev/wd0a / ffs rw 1 1
/var/swap /var/swap swap sw 0 0
```

Swappen über ein vnode Gerät

Dies ist eine dauerhaftere Lösung, um mehr Swap-Speicher 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 Konfigurations-Datei (wenn du dir nicht sicher bist, sieh dir die Sektion "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, in die geswappt werden soll, erzeugt werden. Du solltest mit dem selben Kommando wie im Beispiel oben machen.

```
ericj@oshibana> 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 die */etc/fstab* eintragen. Hier ist eine Beispielzeile, mit der man dieses Gerät beim booten als Swap benutzt.

```
ericj@oshibana> 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.

```
ericj@oshibana> 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 Tabelle eingetragen wurde.

```
ericj@oshibana> sudo swapctl -a /dev/vnd0c
ericj@oshibana> 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

In den letzten paar Jahren hat Kirk McKusick an etwas gearbeitet, was "Soft Updates" genannt wird. Es basiert auf einer Idee von Greg Ganger und Yale Patt, die besagt, dass ein teilweises Ordnen der "buffer cache" Operationen die Notwendigkeit des synchronen Schreibens von Verzeichnis Einträgen im FFS Code überflüssig machen würde. Ergo ein großer Performance-Gewinn beim Schreiben auf Festplatten.
Da Soft Updates sich im Ganzen noch in Entwicklung befinden, wird ein fsck nach einem abrupten Abschalten des Computers ohne saubere Shutdown Sequenz immer noch notwendig sein, das wird aber in zukünftigen Versionen entfallen.

Mehr Interna und Details über Soft Updates kann man in den Untersuchungen von Ganger und Patt und auch von McKusick finden.

Um Softupdates aktivieren zu können, muss dein Kernel folgende Option haben:

option FFS_SOFTUPDATES

Diese Option ist beginnend mit OpenBSD Version 2.9 im GENERIC Kernel enthalten. Trotzdem musst du sie pro mount-Point mit Hilfe einer mount-Option aktivieren.

Beginnend mit 2.9 werden die soft updates durch eine "mount time" Option anstelle von tunefs(8) aktiviert. Wenn du jetzt eine Partition mit dem mount(8) Utility mountest, kannst du angeben, dass soft updates auf dieser Partition aktiviert sein sollen. Unten ein beispielhafter */etc/fstab* Eintrag, der eine Partition *sd0a* hat, die wir mit soft updates gemountet haben wollen.

```
/dev/sd0a / ffs rw,softdep 1 1
```

Wenn du eine ältere OpenBSD Version als 2.9 verwendest, boote in den single-user mode:

```
boot> boot -s
[snip]
bsd# tunefs -s enable <raw device>
bsd# reboot -n
```

Hinweis für Sparc User: Auf sun4 und sun4c Maschinen sollten soft updates nicht eingeschaltet werden. Diese Archiktekturen unterstützen nur eine sehr begrenzte Menge an Kernel Speicher und können dieses Feature nicht nutzen.

14.6 - Wenn ich nach der Installation von OpenBSD/i386 boote, stoppt der Rechner bei "Using partition 3 id 0".

Das bedeutet, dass dein MBR (Master Boot Record) nicht sauber installiert wurde oder dass dein BIOS eine andere Idee hat, was die Geometrie deiner Fesplatte angeht, und zwar eine die nicht kompatibel ist mit deinem jetzigen MBR. Um das Problem zu lösen, solltest du zuerst versuchen, den OpenBSD boot block erneut zu installieren. Dazu lese fdisk(8) und installboot(8).

Um es zunächst zum Laufen zu kriegen, musst du deine boot disk als bootstrap benutzen. Danach tust du deine Install Disk rein, und bevor sie den Kernel und die ramdisk lädt, bekommst du einen `boot>` Prompt von der Floppy Disk. Benutze ihn, um OpenBSD von deiner Festplatte zu booten.

```
booting...
OpenBSD boot 1.2.3
probing hd0 fd0...
boot> boot hd0a:/bsd
```

Jetzt, da du gebootet hast, und unter der Vorraussetzung, dass du deine ganze Festplatte für OpenBSD reserviert hast, kannst du den Master Boot Record mit fdisk(8) reinitialisieren. (Falls du auf deiner Festplatte Partitionen für andere Betriebssystem hast, kannst du auf keinen Fall `installboot` verwenden, stattdessen musst du dir eine andere Option wie z.B. OS-BS ansehen, siehe weiter unten)

```
# fdisk -i wd0
```

Jetzt musst du die boot blocks wieder schreiben.

```
# cp /usr/mdec/boot /boot
# /usr/mdec/installboot -v /boot /usr/mdec/biosboot wd0
```

Und zum Schluss musst du rebooten und es testen.

Wenn das nicht funktioniert hat, hast du immer noch ein paar Optionen. Deine Glückssträhne ist also noch nicht zu Ende. Die erste ist, einen Bootloader wie OS-BS zu verwenden. Auf der OpenBSD CD-ROM ist der os-bs Bootloader im 'tools' Verzeichnis. Wenn du keine CD-ROM gekauft hast, kannst du os-bs von jedem OpenBSD ftp-Mirror herunterladen. Die benötigte Datei ist `pub/OpenBSD/2.8/tools/osbs135.exe`

Nimm dir auch die Zeit, dir die OS-BS Webseiten anzusehen: http://www.prz.tu-berlin.de/~wolf/os-bs.html

Es gibt auch einige andere kommerzielle Bootloader oder auch lilo, die du für das multi-booting nehmen kannst.

Hier ist eine kleine Anweisung, wie du lilo auf dein System bekommst.

- Boote mit einer DOS floppy und gib ein "fdisk /MBR" ein. Stelle sicher, dass du auch auf der Festplatte bist, von der du booten willst.
- Boote von einer Linux Disk, installiere LILO & und verknüpfe es mit deinem OpenBSD Bootblock.

Ausführlichere Anweisungen gibt es in INSTALL.linux

14.7 - Wie man ein dmesg von einer Boot-Floppy bekommt

RAMDISK Images (boot floppies) enthalten das *dmesg* Werkzeug leider nicht. Sie mounten aber das `/kern` Dateisystem. Um die dmesg Informationen in eine Datei zu kopieren kannst du z.B. folgendes eingeben:

```
# cat /kern/msgbuf >mydmesg
```

Boot disks enthalten aber 'more', um seitenweise durch die Ausgabe zu scrollen :

```
# more /kern/msgbuf
```

Prüfe auch die Sektion 4.5

14.8 - Bootblocks installieren - 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 BIOSse (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 boot loader' zu laden (und kannst daher den Kernel auch nicht laden) (Das liegt daran, dass der 'first stage boot loader' 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 BIOSse bieten 3 verschiedene Übersetzungen.) Um deinen Bootblock zu reparieren, damit du normal booten kannst, stecke einfach eine Boot floppy in dein Floppy-Laufwerk 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 Boot block schreiben). Unser Beispiel geht davon aus, dass deine boot disk sd0 ist (bei IDE wäre es wd0, etc..) :

```
# cd /usr/mdec; ./installboot /boot biosboot sd0
```

Wenn installboot sich darüber beschwert, dass es die BIOS Geometrie nicht lesen kann, kannst du am boot> Prompt das "machine diskinfo" (oder kürzer "ma di") Kommando eingeben, damit es die Informationen ausgibt, die du brauchst. Füttere die "heads" und "secs" Werte in installboot's -h und -s Flags, so dass das modifizierte installboot Kommando wie folgt aussieht:

```
# cd /usr/mdec; ./installboot -h <heads> -s <secs> /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 - Auf die Katastrophe vorbereiten: Backups machen und Wiederherstellen mit Bändern (tapes)

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 multiple Server auf ein Tape-Drive sichern kann. In den meisten Umgebungen sind dump(8)/restore(8) aber ausreichend. Wenn du aber multiple 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 Tapes. 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 Tape bringen:

Um sein Backup auf ein Band zu bringen, muss man wissen, wo seine Dateisysteme gemountet sind. Das findet man mit dem "mount"-Kommando am Shell-Prompt heraus. Dabei sollte eine Ausgabe wie diese herauskommen:

```
shell# 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:

```
shell# 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 /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
```

Sieh dir einfach die dump-man-page für genauere Angaben über jedes Befehlsargument an.

- **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 Kommando wird am Ende benutzt, um das Band zurückzuspulen. Sieh dir die mt man page 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 Beispiel-Eintrag von dmesg für ein Bandlaufwerk könnte so aussehen:

```
st0 at scsibus0 targ 5 lun 0:
```

Du hast vielleicht bemerkt, dass bei der Sicherung das Bandlaufwerk als "nrst0" anstatt von "sto" bezeichnet wird, wie man es in dmesg sieht. Wenn du auf st0 als nrst0 zugreifst, benutzt du das selbe physikalische Gerät, aber sagst dem Gerät, es solle nicht zurückspulen, nachdem der Job im raw mode beendet wurde. Um multiple 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 man page 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-Mode 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 zweiten 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 wieder herstellen willst, sieh dir die restore man page 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 bootbar zu machen. In den meisten Fällen sind diese Floppy 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 packe es in dein Laufwerk.
- Benutze das fdisk(8) Kommando, um eine primäre OpenBSD Partition auf dieser neu installierten Festplatte zu erzeugen. Z.B.:

  ```
  shell# 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. Z.B.:

  ```
  shell# disklabel -E sd0
  ```

  (Vergiss den swap nicht, siehe dazu die disklabel FAQ)
- Benutze das newfs Kommando, um ein neues sauberes Dateisystem auf jeder Partition zu erstellen, die du mit dem oben aufgeführten Schritt

erstellt hast. Z.B.:

```
shell# newfs /dev/rsd0a
shell# newfs /dev/rsd0h
```

- Mounte dein neu vorbereitetes root (/) Dateisystem auf /mnt. Beispiel:

```
shell# mount /dev/sd0a /mnt
```

- Gehe in das gemountete root Dateisystem und beginne mit dem restore Prozess. Beispiel:

```
shell# cd /mnt
shell# restore -rs 1 -f /dev/rst0
```

- Wenn die Platte bootbar sein soll, schreibe mit dem folgenden Befehl einen neuen MBR auf deine Festplatte:

```
shell# fdisk -i sd0
```

- Zusätzlich zum Schreiben eines neuen MBR musst du boot blocks installieren, um davon booten zu können. Das folgende ist ein kurzes Beispiel:

```
shell# cp /usr/mdec/boot /mnt/boot
shell# /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 kannst 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 shell prompt benutze die folgende Kommandos, um deine Festplatten "abzumelden" (umount) und das System anzuhalten:

```
shell# umount /mnt
shell# 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 ausgführt 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:

```
shell# mount -u -w /
```

- Sobald du im r/w Modus remountet hast, kannst du fortfahren deine restlichen Dateisysteme wiederherzustellen. Beispiel:

```
(einfache mount Tabelle)
shell# mount /dev/sd0h /usr; cd /usr; restore -rs 2 -f /dev/rst0

(etwas umfassendere mount Tabelle)
shell#  mount /dev/sd0d /var; cd /var; restore -rs 2 -f /dev/rst0
shell#  mount /dev/sd0e /home; cd /home; restore -rs 3 -f /dev/rst0
shell#  mount /dev/sd0h /usr; cd /usr; restore -rs 4 -f /dev/rst0
```

  Benutze "**restore rvsf**" anstatt 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 wieder normal zu benutzen.

14.10 - Disk Images in OpenBSD mounten

Um ein Disk Image (ISO images, Disk images, die mit dd erstellt wurden, etc) in OpenBSD zu moutnen, musst du ein vnd(4) Device/Gerät 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 disk images angeben.

Um das Image wieder 'unzumounten' benutze die folgenden Kommandos:

```
# umount /mnt
# vnconfig -u svnd0
```

Mehr Informationen gibt es in der vnconfig(8) man page.

14.11 - Hilfe! Ich erhalte Fehler mit PCIIDE!

PCI IDE DMA ist unzuverlässig. Darum schaltet Microsoft es auch bei seinen Betriebssystemen grundsätzlich ab.

OpenBSD ist aggressiv und versucht den höchsten DMA Modus zu benutzen, den es kriegen kann. Dies führt in einigen Konfigurationen zu Datenkorruptionen aufgrund von defekten Motherboard Chipsets, Treibern, die buggy sind und/oder Lärm auf den Kabeln. Glücklicherweise schützt Ultra-DMA die Daten Transfers 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.

Falls OpenBSD nicht erfolgreich herunterschalten kann, oder der Prozess zu einem "Hard-Lock" deiner Maschine führt, schicke uns bitte einen bug report.

[Zurück zu Haupt-Index] [Zu Sektion 13.0 - IPsec]

---

www@openbsd.org

Originally [OpenBSD: faq14.html,v 1.48]
$Translation: faq14.html,v 1.21 2003/01/17 14:02:59 jufi Exp $
$OpenBSD: faq14.html,v 1.19 2003/01/17 18:47:48 jufi Exp $