NAME

INSTALL — Installation procedure for NetBSD/macppc.

CONTENTS

About this Document....3
Quick install notes for the impatient....3
What is NetBSD?....5
Changes Between The NetBSD 5.0 and 5.0.1 Releases....5
  Security Advisory Fixes....5
  Kernel....6
  Networking....6
  Drivers....6
  Platform specific....6
  Userland....7
  Miscellaneous....7
Changes Between The NetBSD 4.0 and 5.0 Releases....7
  General kernel....8
  Networking....9
  File systems....9
  Security....10
  Drivers....10
  Platforms....14
  Userland....20
  Components removed from NetBSD....23
  Known Problems....24
Features to be removed in a later release....24
The NetBSD Foundation....24
Sources of NetBSD....24
NetBSD 5.0.1 Release Contents....25
  NetBSD/macppc subdirectory structure....26
  Binary distribution sets....27
NetBSD/macppc System Requirements and Supported Devices....28
  Supported models....28
  Unsupported models....30
  Supported devices....30
  Unsupported devices....32
  Supported boot devices and media....33
Getting the NetBSD System on to Useful Media....33
Preparing your System for NetBSD installation....35
  Prepare yourself....35
Preparing your Open Firmware 3 System for NetBSD....36
  Updating your BootROM....36
  Getting to Open Firmware 3....36
  Setting up Open Firmware 3 to boot NetBSD....37
  Available Boot Media....38
  Partitioning your hard drive for NetBSD....39
  Preparing the Open Firmware 3 Bootable Media....40
    Creating the NetBSD/macppc CD-R....41
    Preparing an HFS or HFS+ partition....41
    Preparing a custom hybrid HFS/ISO9660 CD-R....41

Creating an MS-DOS disk....42
Preparing the netboot server....42
Preparing a SCSI or IDE drive with the CD-R image....46
Preparing your Open Firmware 1.x or 2.x System for NetBSD....47
Getting to Open Firmware on Apple Network Servers....47
Open Firmware 1 and 2 System Preparation....48
Getting to Open Firmware (MacOS X or Darwin)....49
Getting to Open Firmware (Best, MacOS 8 or 9)....49
Getting to Open Firmware (Harder, MacOS 7, 8, or....50
Getting to Open Firmware (Without using MacOS)....51
Setting up Open Firmware 1 and 2 to boot....51
Available Boot Media....53
Partitioning your hard drive for NetBSD....54
Preparing the Open Firmware 1 or 2 Bootable Media....55
Creating the NetBSD/macppc CD-R....55
Creating the NetBSD install floppies....56
Creating a custom ISO9660 CD-R....56
Creating an MS-DOS disk....57
Preparing the netboot server....57
Preparing a SCSI or IDE drive with the CD-R image....62
Installing the NetBSD System....63
Open Firmware boot syntax....63
Examples of Open Firmware boot commands....66
Booting the NetBSD/macppc install CD-R....66
Booting the NetBSD install floppies....66
Booting an IDE or SCSI drive with an HFS partition....66
Booting a custom CD-ROM....67
Booting an MS-DOS floppy....67
Booting over the ethernet....67
Booting an IDE or SCSI drive with 'partition zero'....68
Example of a normal boot....68
Common Problems and Error Messages....69
Black screen....69
Grey screen with flashing question mark....69
Information on your screen seems garbled or out of sync....70
DEFAULT CATCH!....70
CLAIM failed....70
can't OPEN....70
unrecognized Client Program formatstate not valid....70
bad partition number, using 0no bootable HFS partition....71
READ TIMEOUT@....71
TFTP timeout....71
enet:,/netbsd.ram.gz: Inappropriate file type or format....71
Bootloader hangs before the copyright notice....71
Hang after configuring devices....71
Milestone....72
Running the sysinst installation program....72
Introduction....72
General....72
Quick install....72
Booting NetBSD....73
Network configuration....73

Preparing a disk for Open Firmware 3 systems.....73
Installation drive selection and parameters.....75
Selecting which sets to install.....75
Partitioning the disk.....75
Preparing your hard disk.....76
Getting the distribution sets.....76
Installation from CD-ROM.....76
Installation using ftp.....76
Installation using NFS.....76
Installation from an unmounted file system.....77
Installation from a local directory.....77
Extracting the distribution sets.....77
Making the device nodes.....77
Finalizing your installation.....77
Finalizing Open Firmware settings 78
Booting NetBSD exclusively.....78
Additional Open Firmware tips.....78
Booting NetBSD and MacOS X or Darwin.....78
Booting NetBSD and MacOS 9 or earlier.....79
Other boot techniques.....79
Post installation steps.....79
Upgrading a previously-installed NetBSD System.....82
Compatibility Issues With Previous NetBSD Releases.....82
Issues affecting an upgrade from NetBSD 3.x releases 83
Issues affecting an upgrade from NetBSD 4.x releases 84
Using online NetBSD documentation.....84
Administrivia.....85
Thanks go to.....85
We are.....86
Legal Mumbo-Jumbo.....95
The End.....99

DESCRIPTION

About this Document

This document describes the installation procedure for NetBSD 5.0.1 on the *macppc* platform. It is available in four different formats titled `INSTALL.`*`ext`*, where *`.ext`* is one of `.ps`, `.html`, `.more`, or `.txt`:

`.ps` PostScript.

`.html` Standard Internet HTML.

`.more` The enhanced text format used on UNIX-like systems by the `more`(1) and `less`(1) pager utility programs. This is the format in which the on-line *man* pages are generally presented.

`.txt` Plain old ASCII.

You are reading the *PostScript* version.

Quick install notes for the impatient

This section contains some brief notes describing what you need to install NetBSD 5.0.1 on a machine of the macppc architecture.

- Fetch files needed to install NetBSD. The files depend on which model you are using and how you plan to boot your machine. For systems with built-in floppy drives (Open Firmware 1 or 2), fetch the pair of boot floppy images `macppc/installation/floppy/boot1.fs` and `macppc/installation/floppy/boot2.fs`, which include the bootloader and installation kernel. For systems without floppy drives (most are Open Firmware 3), fetch the bootloader `macppc/installation/ofwboot.xcf` and the installation kernel `macppc/binary/kernel/netbsd-GENERIC_MD.gz`. If you have a CD-R, you can fetch the CD image, `macppccd-5.0.1.iso`. Alternatively, you may netboot the installation kernel. This process is covered below, in detail.

- The actual binary distribution is in the `macppc/binary/sets/` directory. When you boot the install kernel from floppies, hard drive, or CD-ROM, the installation program can fetch these files for you (using e.g. ftp) if you have a network connection. There are several other methods to get the binary sets onto your machine.

  You will at a minimum need the following sets: `kern-GENERIC.tgz`, `base.tgz` and `etc.tgz`. In a typical workstation installation you will probably want all the installation sets.

- If your macppc has a floppy drive, create the pair of boot floppies using **`suntar`** (MacOS 9), **`rawrite`** (Windows), or **`dd`** (any UNIX-like system with floppy support). If your system has Open Firmware 3, drag `ofwboot.xcf` and `netbsd-GENERIC_MD.gz` to your hard drive icon (the top level of the drive, not the desktop). If you are using the CD image, burn it now.

  The disk(s) you just prepared will be used to boot the installation kernel, which contains all the tools required to install NetBSD.

- Determine your machine's model, quirks, and Open Firmware version from the NetBSD/macppc Model Support webpage.

  http://www.NetBSD.org/ports/macppc/models.html

  At present, NetBSD/macppc cannot exist on the same hard drive as Mac OS unless you partition your disk before running the installer. Open Firmware versions prior to 3 require a dedicated NetBSD drive — you must use the entire disk, partitioned with the installation tools. Open Firmware version 3 cannot boot into NetBSD on a drive partitioned with the installation tools, you must partition your disk before running the installer, then select the "**Re-install sets or install additional sets**" option in the installer (selecting the "**Install NetBSD to hard disk**" or "**Upgrade NetBSD on a hard disk**" options will render your drive unbootable). If you are unsure, you may want to read the section below on **Partitioning your hard drive for NetBSD**

- For systems with Open Firmware versions prior to 3, you may need to use Apple's System Disk utility to enter Open Firmware and use your screen and keyboard. To enter Open Firmware, hold down the `COMMAND-OPTION-O-F` keys after the boot chime starts, but before the chime ends. Entering Open Firmware versions prior to 3 is usually the most frustrating part of installation — you may want to read the section below on **Older Open Firmware System Preparation**

  You should have the Open Firmware "`0 >`" prompt on your screen before attempting to boot NetBSD/macppc.

- At the Open Firmware prompt, type the command to boot. To boot from the installation floppies, the command is "**`boot fd:0`**". For the install kernel and bootloader on your hard drive (Open Firmware 3), the command is "**`boot hd:,\ofwboot.xcf netbsd-GENERIC_MD.gz`**".

  For boot CDs, the command is something like "**`boot cd:,\ofwboot.xcf netbsd.macppc`**" (for Open Firmware 3) or "**`boot scsi-int/sd@3:0 NETBSD.MACPPC`**" (for earlier Open Firmware versions). You will need to use the correct case for **`OFWBOOT.XCF`** and **`NETBSD.MACPPC`** depending on how your version of Open Firmware interprets the ISO file system. You may need to replace **`cd`** with **`scsi/sd@3 , scsi-int/sd@3 , ata/atapi-disk ,`** or some other device

alias. You should also use the Open Firmware **dir** command to confirm that the NetBSD/macppc kernel is called NETBSD.MACPPC. You may want to read the section below on **Open Firmware boot syntax**

- For third-party programs which are not part of the base NetBSD distribution, you will want to explore the **pkgsrc** system with its many thousands of third party software applications.

What is NetBSD?

The NetBSD Operating System is a fully functional Open Source UNIX-like operating system derived from the University of California, Berkeley Networking Release 2 (Net/2), 4.4BSD-Lite, and 4.4BSD-Lite2 sources. NetBSD runs on 57 different system architectures (ports) across 15 distinct CPU families, and is being ported to more. The NetBSD 5.0.1 release contains complete binary releases for many different system architectures. (A few ports are not fully supported at this time and are thus not part of the binary distribution. Please see the NetBSD web site at **http://www.NetBSD.org/** for information on them.)

NetBSD is a completely integrated system. In addition to its highly portable, high performance kernel, NetBSD features a complete set of user utilities, compilers for several languages, the X Window System, firewall software and numerous other tools, all accompanied by full source code.

NetBSD is a creation of the members of the Internet community. Without the unique cooperation and coordination the net makes possible, it's likely that NetBSD wouldn't exist.

Changes Between The NetBSD 5.0 and 5.0.1 Releases

The NetBSD 5.0.1 release is the first security/critical update of the NetBSD 5.0 release branch. This represents a selected subset of fixes deemed critical in nature for stability or security reasons.

Please note that all fixes in security/critical updates (i.e., NetBSD 5.0.1, 5.0.2, etc.) are cumulative, so the latest update contains all such fixes since the corresponding minor release. These fixes will also appear in future minor releases (i.e., NetBSD 5.1, 5.2, etc.), together with other less-critical fixes and feature enhancements.

The complete list of changes can be found in the CHANGES-5.0.1:
http://ftp.NetBSD.org/pub/NetBSD/NetBSD-5.0.1/CHANGES-5.0.1
file in the top level directory of the NetBSD 5.0.1 release tree. An abbreviated list is as follows:

Security Advisory Fixes

- NetBSD-SA2009-004 (NetBSD OpenPAM passwd(1) changing weakness):
  http://ftp.NetBSD.org/pub/NetBSD/security/advisories/NetBSD-SA2009-004.txt.asc
- NetBSD-SA2009-005 (Plaintext Recovery Attack Against SSH):
  http://ftp.NetBSD.org/pub/NetBSD/security/advisories/NetBSD-SA2009-005.txt.asc
- NetBSD-SA2009-006 (Buffer overflows in ntp):
  http://ftp.NetBSD.org/pub/NetBSD/security/advisories/NetBSD-SA2009-006.txt.asc
- NetBSD-SA2009-007 (Buffer overflows in hack(6)):
  http://ftp.NetBSD.org/pub/NetBSD/security/advisories/NetBSD-SA2009-007.txt.asc
- NetBSD-SA2009-008 (OpenSSL ASN1 parsing denial of service and CMS signature verification weakness):
  http://ftp.NetBSD.org/pub/NetBSD/security/advisories/NetBSD-SA2009-008.txt.asc
- NetBSD-SA2009-009 (OpenSSL DTLS Memory Exhaustion and DSA signature verification vulnerabilities):
  http://ftp.NetBSD.org/pub/NetBSD/security/advisories/NetBSD-SA2009-009.txt.asc

- NetBSD-SA2009-010 (ISC dhclient subnet-mask flag stack overflow):
  http://ftp.NetBSD.org/pub/NetBSD/security/advisories/NetBSD-SA2009-010.txt.asc
- NetBSD-SA2009-011 (ISC DHCP server Denial of Service vulnerability):
  http://ftp.NetBSD.org/pub/NetBSD/security/advisories/NetBSD-SA2009-011.txt.asc
- NetBSD-SA2009-012 (SHA2 implementation potential buffer overflow):
  http://ftp.NetBSD.org/pub/NetBSD/security/advisories/NetBSD-SA2009-012.txt.asc
- NetBSD-SA2009-013 (BIND named dynamic update Denial of Service vulnerability):
  http://ftp.NetBSD.org/pub/NetBSD/security/advisories/NetBSD-SA2009-013.txt.asc

Advisories prior to NetBSD-SA2009-004 do not affect NetBSD 5.0:
http://www.NetBSD.org/support/security/patches-5.0.html

Kernel

- Fix random "filesystem full" messages on large FFS file systems.
- Fix a regression in the 4.4BSD scheduler, improving interactive performance under load.
- Remove a race where physio_done() may use memory already freed. Fixes PR kern/39536.
- Fix a crash observed when trying to load a corrupted ELF kernel module.
- Fix PR kern/41566, where writes on the controlling tty were not being awoken from blocks.
- Various fixes for POSIX message queues.
- Fix a possible deadlock in the VFS subsystem.
- Fixes for POSIX advisory locks.
- A number of other stability fixes.

Networking

- Follow exactly the recommendation of draft-ietf-tcpm-tcpsecure-11.txt: Don't check gainst the last ack received, but the expected sequence number. This makes RST handling independent of delayed ACK.
- Fix a panic when trying to disable IPFilter before enabling it. Fixes PR kern/41364.

Drivers

- `ehci`(4): Add a workaround for ATI SB600 and SB700 revisions A12 and A13 to avoid a USB subsystem hang when the system has multiple USB devices connected to it or one device is re-connected often.
- `wm`(4):
  - On i82563, FreeBSD's em driver says that the ready bit in the MDIC register may be incorrectly set. Insert delay(200) like the em driver. Fixes PR kern/41014.
  - Add workaround for 82543GC. We need to force speed and duplex on the MAC equal to what the PHY speed and duplex configuration is. Fixes PR kern/36430.
  - Fix many problems and panic on TBI's cards (PR kern/32009).

Platform specific

- x86 (amd64 and i386): Add a workaround for a bug with some Opteron revisions where locked operations sometimes do not serve as memory barriers, allowing memory references to bleed outside of critical sections.

- amd64: Handle protection faults properly, returning SIGSEGV instead of SIGBUS.
- hp300: Make install.md probe cd(4) devices properly.
- pmax: Make ksyms(4) actually work.
- sh3: Fix logic error in copyinstr() when deciding whether to return EFAULT or ENAMETOOLONG.
- sparc64:
  - Fix long double support in 32bit libc. Fixes PR port-sparc64/41406.
  - When preparing the initial trap frame for a new forked lwp, explicitly clear condition code. Otherwise we might catch a signal before we ever return to userland. Fixes PR port-sparc64/41302.
- vax: binutils: Allocate relocation section using bfd_zalloc() to ensure no garbage relocations when not all the entries are used. Fixes PR port-vax/39182.

Userland

- Update pkg_install to 20090724.

  Note: pkg_install now depends on the pkgdb cache for automatic conflict detection. It is recommended to rebuild the cache with

  ```
  # pkg_admin rebuild
  ```

  audit-packages.conf(5) has been superseded by pkg_install.conf(5). The default configuration is the same.

  Support for pkg_view(1) has been retired.

  The functionality of audit-packages(1) and download-vulnerability-list(1) has moved into pkg_admin(1). However, wrapper scripts that handle the common use cases are provided.
- Update libfetch to 2.23.
- racoonctl(8): Adjust ADMINPORTDIR to match that of racoon (/var/run). Fixes PR bin/41376.
- schedctl(8): Skip LSIDL and LSZOMB threads when retrieving info.
- postinstall(8) now knows about /etc/dhcpcd.conf.

Miscellaneous

- The X.Org s3 driver now works.
- Install the Xvidtune app-defaults file.
- Fixes to Linux compat:
  - In sendmsg(2), do copy the msghdr structure before trying to use it.
  - In linux_sys_sched_getaffinity(), do not leak memory on error.
- Various METALOG fixes, including sorting entries. Addresses PR toolchain/24457 and PR bin/41155.

Changes Between The NetBSD 4.0 and 5.0 Releases

The NetBSD 5.0 release provides numerous significant functional enhancements, including support for many new devices, integration of hundreds of bug fixes, new and updated kernel subsystems, and many user-land enhancements. The result of these improvements is a stable operating system fit for production use that rivals most commercially available systems.

It is impossible to completely summarize the massive development that went into the NetBSD 5.0 release. The complete list of changes can be found in the CHANGES:

ftp://ftp.NetBSD.org/pub/NetBSD/NetBSD-5.0/CHANGES

and CHANGES-5.0:

ftp://ftp.NetBSD.org/pub/NetBSD/NetBSD-5.0/CHANGES-5.0

files in the top level directory of the NetBSD 5.0 release tree.

Some highlights include:

General kernel

- Introduced new synchronization primitives and made almost all core kernel subsystems use fine-grained locking, resulting in much greater performance on MP systems. See `mutex`(9), `rwlock`(9), and `condvar`(9).
- The threading system was rewritten and is now based on a 1:1 model.
- Made the process scheduler modular, allowing for different scheduling algorithms to be selected.
- The default scheduler implementation was rewritten, giving large performance improvements on SMP systems, and bringing support for real-time and time-sharing classes.
- Added support for SCHED_M2, a second scheduling algorithm, as an alternative to the original SCHED_4BSD algorithm.
- Added support for POSIX real-time scheduling extensions.
- Added support for POSIX asynchronous I/O.
- Added support for POSIX message queues.
- Added support for processor sets, CPU sets, and thread affinity. See `pset`(3), `cpuset`(3), and `affinity`(3).
- Added MI code to support in-kernel preemption and made i386 and amd64 use it. See `kpreempt`(9).
- Added a new implementation of software interrupts in a machine-independent way to provide software interrupts with thread context. All ports were made to use it. See `softint`(9).
- Added support for per-system call statistics.
- Added support for per-CPU work-queues. See `workqueue`(9).
- Added support for kernel core dumps to software RAID 1 sets.
- Added `mremap`(2), to remap virtual memory addresses.
- Added support for dynamic changing of SysV message queues and semaphore limits.
- Improved pagedaemon behavior on memory shortage.
- Added TFTPROOT option for TFTPing root ramdisk at root mount time.
- Added `putter`(9) (Pass-to-Userspace Transporter), a generic request-response handler for kernel-attached userspace daemons.
- Added a new power management framework, `pmf`(9).
- By default, the kernel no longer enters `ddb`(4) if it panics. To get the old behavior, set ddb.onpanic to 1 in /etc/sysctl.conf.
- `ddb`(4): Added a "whatis" command, inspired by Solaris.

- All ports made to use generic `todr`(9).
- All ports made to use `timecounter`(9).
- Added Mersenne Twister PRNG implementation.
- Added `posix_madvise`(2).
- Numerous improvements to `compat_linux`(8) and compat_linux32.
- Device attach and detach notifications are now exposed via /dev/drvctl.
- Added `sockopt`(9), a new kernel API for passing socket options.
- If no usable `init`(8) program can be found, set the RB_ASKNAME flag and prompt users for the init path.

Networking

- Added ipv6 fast forward support, which increases forwarding speed by hashing/caching flows.
- Added per-socket keepalive timer settings and the ability to change connection timeouts.
- Added support for automatic sizing of TCP socket buffers.
- Added a new socket option SO_NOHEADER to tell the network stack to strip the protocol header from packets received on a raw socket.
- Added accept filters. See `accept_filter`(9), `accf_data`(9), and `accf_http`(9).
- `fast_ipsec`(4): Added support for IPsec NAT-T.
- `pf`(4): Made "nat" and "rdr" translation rules obey state policy flags. Extended `pf.conf`(5) syntax to express the translation-state policies.
- Network statistics were changed to be per-CPU.
- ALTQ: Added support for source address hashing in the WFQ scheduler.
- Pulled in IANA-generated services and protocols files.

File systems

- Added `wapbl`(4), a preview of metadata journaling for FFS. Contributed by Wasabi Systems.
- Added read-only support for Apple HFS+ file systems. See `mount_hfs`(8).
- Added read-only support for Silicon Graphics EFS file systems. See `mount_efs`(8).
- Added write support for UDF file systems. See `mount_udf`(8).
- Added `mount_psshfs`(8), to support `puffs`(3) sshfs.
- Added `mount_9p`(8), to support 9P file services with `puffs`(3).
- Added `rump_nfs`(8), a userspace NFS client.
- Added `mount_sysctlfs`(8), to support browsing, querying, and modifying the `sysctl`(3) hierarchy.
- Added `refuse`(3) for FUSE compatibility functionality, layered on top of `puffs`(3).
- Added the user kernel file system library, `ukfs`(3), for standalone file system access.
- Added `p2k`(3), a `puffs`(3) to kernel vfs adaption library.

- puffs(4): Added support for NFS exporting puffs file servers.
- Added 32 bit UID/GID support for ext2fs.
- Added a newfs_ext2fs(8) utility to create Ext2 file systems.
- /proc is now mounted by default. Extended the Linux emulation of /proc to include /proc/stat, /proc/loadavg, and /proc/<pid>/statm. Linux compatibility mode is now the default. Use -o nolinux if Linux compatibility is not desired.
- Write speed to FAT file systems was significantly increased and the 255 head limit was removed.
- Imported FreeBSD's unionfs.
- fsck_ext2fs(8), fsck_ffs(8), fsck_lfs(8): Disable userid to username lookups by default and add -U flag to perform them.
- scan_ffs(8): Added -b option to search a partition for valid alternate superblocks.
- fsck_ffs(8): Added -x/-X options, which allow running fsck_ffs -n on a snapshot of a live filesystem.

Security

- Added support for per-user /tmp. See security(8).
- Added support for ASLR (Address Space Layout Randomization) in the kernel and dynamic linker. See security(8).
- Added an opencrypto(9) provider for VIA ACE (AES encryption instructions).
- Added nsp(4), a driver for NetOctave NSP2000, contributed by NBMK Encryption Technologies, ported from vendor FreeBSD SDK and integrated with opencrypto by Coyote Point Systems.
- opencrypto(9): Improved performance by adding asynchronous operation and batched submit/retrieve of requests/results. Contributed by Coyote Point Systems.
- cgd(4): Changed the default IV to encblkno1, which is faster without a real loss of security.
- openssl(1): Enabled support for Camellia.

Drivers

- Audio:
  - Added pad(4), a pseudo-audio device driver for feeding back raw PCM data to userland.
  - Added sgsmix(4), a driver for the SGS 7433 mixer found in some G3 Macs.
  - umidi(4): Added support for Roland UA25, UA4FX, and SonicCell devices.
  - dbri(4): Added support for audio input.
  - auvia(4) now works on big endian machines.
  - azalia(4): AD1984 support was greatly improved. Added support for Realtek ALC662-GR and ALC269 codecs, which are found in EeePCs. Added support for ALC268.
  - auich(4): Fixed a clock accounting problem that prevented detecting sample rates correctly.
- Hardware Monitoring:
  - Imported envsys2, which brings, among other things, the ability to send events to powerd(8). See envsys(4) and sysmon_envsys(9).

- Added support for hardware monitoring on Sun Ultra Enterprise 450.
- Added aiboost(4), a driver for the ASUS AI Booster ACPI Hardware monitor.
- Added smsc(4), a driver for the hardware monitoring portion of the SMSC LPC47B397.
- Added aps(4), a driver for the IBM Thinkpad Active Protection System.
- Added coretemp(4), a driver for Intel Core (and newer) on-die thermal sensors.
- Added finsio(4), a driver for various Fintek Super I/O chips.
- Added amdtemp(4), a driver for AMD CPU on-die thermal sensors.
- Added dbcool(4), a driver for Analog Devices dbCool chips including ADT7460, ADT7463, ADT7467, and ADM1030.
- Added alipm(4), a driver for the Acer Labs M7101 Power Management Controller.
- Added admtemp(4), a driver for the Analog Devices ADM1021 temperature sensor.
- viaenv(4): Added support for VIA VT8231.
- nsclpcsio(4): Added support for the VLM logical device.
- lm(4): Added an i2c attachment for the LM78 family of temp sensor and fan controllers.

• Networking:

- Added jme(4), a driver for the JMicron Technologies JME250 Gigabit Ethernet and JME260 Fast Ethernet controllers.
- Added u3g(4), a driver for many multi-port 3G datacards.
- Added iwn(4), a driver for the Intel Wireless LAN 4965AGN adapter.
- Added zyd(4), a driver for ZyDAS ZD1211/ZD1211B USB IEEE 802.11b/g wireless network devices.
- Added uhmodem(4), a driver for 3G wireless modems including Huawei E220 and E620, E-mobile D01HW and D02HW, and NTT DoCoMo a2502.
- Added lii(4), a driver for the Atheros L2 Fast Ethernet controller.
- Added btuart(4), a driver for Bluetooth HCI UART (H4).
- Firmware images are now shipped for ipw(4), iwi(4), wpi(4), and iwn(4). For ipw(4) and iwi(4), the Intel EULA has to be accepted via sysctl(8).
- fxp(4): Fixed some TX timeout and RX pool corruption problems. Added a workaround for a hardware ip4csum-tx bug.
- bge(4): Added support for BCM5786 and BCM5906(M). Fixed fiber card support.
- nfe(4): Added support for NVIDIA MCP67/73 Ethernet controllers. Fixed wakeup issues on some newer chips. Fixed a problem with receiving jumbo frames.
- btbc(4): Added support for AnyCom BlueCard devices.
- rum(4): Added support for MELCO WLI-U2-SG54HP, PLANEX GW-US54Mini2, COREGA CG-WLUSB2GL and K.K. CG-WLUSB2GPX, and ABOCOM WUG2700.
- makphy(4): Added support for Marvell 88E1116 Gigabit PHY.

- gem(4): Added support for Sun PCI SX fiber cards and Sun SBus SX fiber cards.
- wm(4): Added support for the Intel PRO/1000 PT Quad Port Server Adapter. Fixed a bug on receiving a jumbo frame which lead to a panic in sbcompress(). Added support for more ICH9 devices. Fixed an EEPROM-trashing bug on ICH8 and ICH9 chipsets.
- udav(4): Added support for Shantou ADM8515.
- brgphy(4): Added support for BCM5708C.
- re(4): Made hardware vlan(4) insertion/extraction work properly. Added support for the Realtek 8102E/8102EL PCIe 10/100 Ethernet adapters, as well as the 8111C chips that are found on many Intel-based motherboards.
- sk(4), msk(4): Fixed a lock panic on receiving jumbo packets.
- msk(4): Fixed a Yukon EC Ultra cold power up issue.
- bnx(4): Added support for SerDes controllers.
- vge(4): Added ifconfig down and ALTQ support.

• Storage:

- Added arcmsr(4), a driver for Areca Technology Corporation SATA RAID controllers.
- Added siisata(4), a driver for Silicon Image SteelVine SATA-II controllers (SiI3124, SiI3132, and SiI3531).
- isp(4): Major update, including 4Gb (24XX) card support and new firmware sets.
- piixide(4): Added support for ICH10.
- ahcisata(4): Added support for ATAPI devices.
- svwsata(4): Added support for ServerWorks HT-1000 SATA controller.
- njata(4): Added support for Workbit CF32A CF adapter.
- viaide(4): Added support for VIA CX700, CX700M2, NVIDIA MCP67, and MCP73/77 controllers.
- mfi(4): Added support for LSI SAS1078 and Dell PERC 6 controllers.
- mpt(4): Fixed performance problems for old revisions of the Symbios 53c1030.
- cac(4): Added initial bio(4) support; only volume status is handled at this time.
- siop(4): Added support for the non-PCI NCR 53c720/770 in big-endian mode.
- twa(4): Added support for 3ware 9650 and 9690, based on contributions from Wasabi Systems.
- ciss(4): Added bio(4) support.
- ataraid(4): Added NVIDIA MediaShield, JMicron RAID, and Intel MatrixRAID support. Added support for status reports through bio(4).
- ixpide(4): Added support for ATI SB700/SB800 controllers.
- aac(4): Added support for raw I/O mode and >2TB.
- umass(4): Added support for Sony GPS GPS-CS1. Fixed a panic on device removal.
- aic(4): Worked around an rbus resource allocation problem so that aic PCMCIA cards work again.

- Implemented DIOCACHESYNC for RAIDframe, vnd(4), and cgd(4).
- rccide: Added support for the ServerWorks HT-1000 IDE controller.
- wd33c93: Enabled Fast SCSI transfers for WD33C93B parts with appropriate input clocks.

- USB:
  - Added uvideo(4), a driver for USB Video Class capture devices, from Patrick Mahoney's Google Summer of Code 2008 project.
  - Added uslsa(4), a driver for CP210x USB-RS232 devices.
  - Added uchcom(4), a driver for WinChipHead CH341/340 and HL-340 USB-Serial adapters.
  - Added uberry(4), a driver to allow RIM BlackBerries to charge from the USB port.
  - Added appropriate bus_dmamap_sync(9) calls to uhci(4), ohci(4), and ehci(4) to prevent the CPU from reordering loads and stores against DMA descriptors. This fixes "host controller process error/host controller halted" errors.
  - ehci(4): Added isochronous transfer support, contributed by Jeremy Morse as part of his Google Summer of Code 2008 project.
  - ukbd(4): Added support for function keys F16 through F19.
  - uplcom(4): Added support for Willcom WS002IN PHS and SMART Technologies-badged devices. Recognize Corega CG-USBRS232R as a serial device.
  - ugensa(4): Added support for Novatel Wireless Merlin CMDA and Ovation U727.
  - ubsa(4): Added support for CDMA modems sold by Eurotel/O2.
  - uftdi(4): Added support for Sealevel SeaPORT+4 USB to Serial adapter.
  - slhci(4): Replaced with Matthew Orgass's driver.
- Graphics and Video:
  - Switched to X.Org on amd64, i386, macppc, sgimips, shark, and sparc64.
  - Added video(4), a video4linux2 compatible capture interface, part of Patrick Mahoney's Google Summer of Code 2008 project.
  - Added uvideo(4), a driver for USB Video Class capture devices, from Patrick Mahoney's Google Summer of Code 2008 project.
  - Added pseye(4), a driver for the Sony PLAYSTATION(R) Eye USB webcam.
  - Added genfb(4), a generic framebuffer console driver with PCI and SBus frontends.
  - Added AGPv3 support to VIA AGP driver.
  - Added support for Fujitsu AG-10e graphics cards.
  - Added r128fb, an accelerated console driver for ATI Rage 128 graphics controllers.
  - Added isv(4), a driver for the IDEC Supervision/16 image capture board.
  - wscons(4): Added scrollback support to vcons.
  - cgfourteen(4): Added support for wscons.
  - agp(4): Added support for ALI M1689, MB i965Q, Intel Q33/35/G33, Intel 945GME, and Intel 946GZ.

  - bktr(4) now works on amd64.

- Miscellaneous:
  - Added pud(4), a driver that makes it possible to implement block and character devices in userspace.
  - Added spdmem(4), a Serial Presence Detect driver that decodes technical specs stored in the eeprom on common types of memory modules.
  - Added bcsp(4) to support the BlueCore Serial Protocol.
  - Added thinkpad(4), a driver to support IBM/Lenovo Thinkpad hotkeys, brightness controls, and temperature and fan monitoring.
  - Added gcscpcib(4), a driver for the AMD CS5535 and CS5536 Companion Device with support for the timecounter, watchdog timer, and GPIO.
  - Added ichsmb(4), a driver for Intel ICH SMBus controllers.
  - Added asus(4), a driver for ASUS ACPI hotkeys as found in the EeePC.
  - Added acpidalb(4), a driver for PNP0C32 ACPI hotkeys, aka the Direct Application Launch Buttons.
  - Added hpqlb(4), a driver for hotkeys on some HP notebooks.
  - adb(4): New and simplified MI ADB drivers.
  - ichlpcib(4): Added support for the TCO (watchdog) on ICH6 or newer chipsets. Now runs on EM64T systems as well.
  - itesio(4): Added support for the watchdog timer.
  - ulpt(4): Implemented non-blocking read.
  - puc(4): Added support for the I-O DATA RSA-PCI 2 port serial board, Digi International 4 and 8 port boards, and B&B Electronics MIPort serial boards.
  - piixpm(4): Added support for ATI SB600, SB700, and SB800 SMBus controllers.
  - wscons(4): Added support for the Colemak keyboard layout.
  - com(4): Added support for PCMCIA Sierra Wireless Aircard 850.
  - nfsmb(4): Added support for numerous NVIDIA chipsets.

Platforms

- x86 (i386, amd64):
  - Removed GENERIC.MP and made MULTIPROCESSOR mandatory.
  - Enabled BUFQ_PRIOCSCAN in GENERIC for better responsiveness under heavy disk I/O.
  - Added fatboot, a sub-512 byte bootloader that can be written into the first sector of a FAT16 file system.
  - Added support for enhanced speedstep on Intel Atom CPUs.
  - Added a High Precision Event Timer driver.
  - Merged amd64 and i386 pmap. Large pages are always used if available.
  - Added support for the Intel 82G965 chipset.

  - Added `boot.cfg`(5) to configure the bootloader.
  - The boot menu now allows disabling ACPI and/or SMP.
  - Added support for ACPI S3 (suspend-to-RAM) on MP systems.
  - Added `mbr`(8) variants that directly access serial ports.
  - It is now possible on some systems to get to a `ddb`(4) session on a VGA console if the system crashed while X11 was running.
  - Added detection of errata for AMD Family 10h steppings A and 2.
  - Added checks for erratum 261 on AMD Family 10h stepping 3 processors.
  - `boot`(8): Added support for the multiboot protocol. This allows booting Xen without GRUB.
  - The kernel now recognizes CD-ROMs as booted devices.
  - Added an Intel On Demand Clock Modulation driver.
- i386:
  - Removed support for 80386 level CPUs.
  - Added support for Microsoft Xbox.
  - Added `cmos`(4), a driver for CMOS RAM.
  - Added support for VIA Esther.
  - i386 bootblocks are now able to boot amd64 kernels and support for a.out kernels has been removed.
  - Removed pccons.
  - Added support for sparse kernel core dumps.
  - Removed the MATH_EMULATE option.
  - Added the i386-specific COMPAT_30_PTHREAD option, which restores binary compatibility with netbsd-3 libpthread.
- amd64:
  - Added support for PCI_BUS_FIXUP and PCI_ADDR_FIXUP kernel options.
- acorn32:
  - Added support for Kinetic cards.
  - Fixed up boot32 to work on A7000+.
- algor:
  - Fixed a `delay`(9) issue and now the P5064 kernel works in gxemul.
- alpha:
  - Make dynamic executables work under `compat_osf1`(8) again.
- amiga:
  - Added a `wdc`(4) frontend for the buddha and catweazle Z2 hardware.
- arc:

  - Added support for booting from UFS2 partitions.
  - Added `sysinst`(8) support.
- arm:
  - Added armv6 kernel support.
  - Added arm32 support for kernel crashdumps.
  - Added VFP support.
  - Added AT91 support.
  - Added FA526 support.
  - Added the missing kernel bits that kept `eeprom`(8) from actually changing firmware settings.
- atari:
  - Fixed the sysinst miniroot image and retired the old miniroot.
  - Fixed a hang on loading md_root from floppy during installation.
  - Many pmap bugs were fixed and it was synced with amiga's pmap.
  - Fixed a lost interrupt problem on Falcon `wdc`(4).
  - Renamed the BOOT kernel to SMALL030 and removed BOOTX.
  - Added support for installation from GEMDOS partitions.
  - Fixed an ncr5380 SCSI freezing problem that occurred during boot.
- bebox:
  - Converted to generic powerpc PCI and ISA frameworks.
  - Added support for `genfb`(4).
- cobalt:
  - Added `sysinst`(8) support.
  - Added support for booting from UFS2 partitions.
  - Added support for booting from Ext2fs partitions.
  - Added netboot support for 21041 on Qube 2700.
  - Startup and shutdown messages, as well as a banner and kernel name are now printed on the LCD panel.
  - Added kernel and bootloader support for optional Z85C30 serial on Qube 2700.
  - Fixed a botched `spl`(9) bug which could cause a network freeze on traffic between two network interfaces.
- evbarm:
  - Added support for the Cortina Systems SL3516 evaluation board.
  - Added support for NPE Ethernet found on IXP425 boards.
  - Added a PXA2x0 RTC driver.

- evbmips:
  - Added support for Infineon ADM5120.
- evbppc:
  - Added support for Xilinx Virtex II-Pro/4-FX.
  - Converted PM/PPC to generic powerpc clock, PCI, and interrupt frameworks.
  - Fixed sysinst to refer to the proper WALNUT and OPENBLOCKS266 kernels.
- hp700:
  - Added support for booting from CD-ROM images.
  - Fixed kernel profiling.
  - Kernels are now compiled with -O2.
  - Added `gdb`(1) support.
  - Added bootinfo support.
  - Added support for NCR 53C720.
- hpcmips:
  - Fixed a silent hangup problem on TX3922 machines.
- ibmnws:
  - Converted to generic powerpc clock, PCI, and interrupt frameworks.
- landisk:
  - Added `sysinst`(8) support.
  - Fixed MD `bus_dmamap_load`(9) so that NFS write works with `re`(4).
- mac68k:
  - Added support for more NuBus video cards, including VillageTronic MacPicasso 340.
  - Added support for Creative Systems Inc Hurdler CPI NuBus parallel printer card.
  - Switched `sn`(4) to use the MI SONIC driver.
- macppc:
  - Switched to generic powerpc interrupt, clock, PCI, `bus_dma`(9), `bus_space`(9), SMP, and IPI frameworks.
  - `genfb`(4) is now the default framebuffer.
  - Switched to the new ADB subsystem.
  - Added support for bus speed control found in some Intrepid-based laptops like the 800MHz iBook G4.
  - Converted to shared ofw_autoconf.
  - Converted to shared powerpc major numbers.
  - Added an experimental Apple UniNorth AGP driver.

- mvmeppc:
  - Converted to generic powerpc clock/PCI/interrupt frameworks.
- m68k:
  - Fixed a panic in FPE code caused by 040/060 specific FDADD instructions on 020/030 machines.
  - `gcc`(1): Fix jump table addressing in the M68k codegen.
  - Fixed floating point handling regressions.
- ofppc:
  - Completely rewrote the ofppc port from scratch.
  - Added support for the bplan/Genesi Pegasos II.
  - Added support for the IBM 7044-270.
  - Added support for the IBM 7046-B50.
  - Added install support and booting from RAID.
  - Fixed ofwboot to work on an IBM CHRP-based RS/6000.
  - Added support for SMP.
- pmppc:
  - Removed pmppc as a port and moved it under evbppc.
- powerpc:
  - Added a generic interrupt handler framework.
  - Added generic shared timecounter-based clock routines.
  - Added shared PReP memory maps.
  - Added generic powerpc OpenFirmware code.
  - Added a generic powerpc IPI framework.
  - Added new generic powerpc SMP support framework.
  - Added tuned assembler from IBM for `memcpy`(3), `memmove`(3), and `memcmp`(3).
  - Added booted_kernel and booted_device sysctl.
  - Added the missing kernel bits that kept `eeprom`(8) from actually changing firmware settings.
- prep:
  - Switched to generic powerpc PCI, ISA, interrupt, clock, IPI, and SMP frameworks.
  - Added support for the MTX604 RAVEN based machine. Other RAVEN based machines are also likely to work now.
  - Added support for floppy controller/drive.
- sandpoint:
  - Switched to generic powerpc interrupt, clock, PCI, and ISA frameworks.
  - Added a DHCP/NFS bootloader.

  - Converted to shared powerpc major numbers.

- sgimips:
  - Added support for the SGI O2 PS/2 controller `macekbc`(4), onboard display adapter `crmfb`(4), and audio driver `mavb`(4).
  - Added support for Set Engineering's GIO Fast Ethernet board: `tl`(4).
  - Added support for the SGI Light/Entry/Starter LG1/LG2 framebuffers found in Indigo systems: `light`(4).
  - Added support for the `sq`(4) interface on the Challenge S's IOPLUS mezzanine.
  - Added support for booting from UFS2 partitions.
  - Many bug fixes and improvements to the MAC-110 Ethernet driver found in the O2.
- sh3:
  - Implemented fast path TLB miss handling.
  - Fixed compilation of native sh3 gcc on 64-bit build machines.
  - Added support for single-stepping in `ddb`(4).
- shark:
  - Added `dhclient`(8) to the install disk image.
  - The installation disk now uses wscons.
  - Removed pccons, which has been obsoleted by `wscons`(4).
  - X works again on revision 4 machines.
- sparc:
  - Added support for booting from UFS2 partitions.
  - Added support for Sun PGX32 / TechSource Raptor GFX 8P to `genfb`(4).
- sparc64:
  - SMP now works.
  - Fixed boot device matching on U5/U10 machines to properly distinguish cd0 from wd0.
- vax:
  - Turned on generation of PIC code.
  - Added an accelerated driver for SPX graphics cards found in some VAXstations.
  - Fixed Unibus support on VAX11/780 and 8600 models.
- xen:
  - Added support for amd64, both as dom0 and domU.
  - Added support for the i386 PAE extension to Xen3 domU.
  - Fixed PCI_BUS_FIXUP/PCI_ADDR_FIXUP support.
- zaurus:

- Added the zaurus port, with support for Sharp C3x00 PDAs.
- Allow Zaurus screen to rotate 90 degrees to a usable state with the keyboard.
- Added support for detecting the AC adapter and charging the battery.
- Added `sysinst`(8) support.

Userland

- 3rd party software updates:
  - ACPICA 20080321
  - am-utils 6.2a3
  - BIND 9.5.0-P2
  - file 4.26
  - GNU gcc 4.1.3-20080831
  - GNU gdb 6.5 on all ports
  - IPFilter 4.1.29
  - (n)awk 20070501
  - NTP 4.2.4p6
  - nvi 1.81
  - openpam 20071221
  - OpenSSH 5.0p1
  - OpenSSL SNAP-20080509
  - PF from OpenBSD 4.2
  - Postfix 2.5.4
  - texinfo 4.8
- Libraries:
  - `pthread`(3):
    - Major overhaul, resulting in many performance and stability improvements.
    - Added POSIX real-time extensions.
    - Diagnostic assertions are now disabled by default. Application errors will be reported only by error return from pthread APIs. The assertions can be re-enabled by setting the PTHREAD_DIAGASSERT environment variable as desired.
    - Added a PTHREAD__COMPAT build flag which creates a libpthread that can be dropped into a NetBSD 2/3/4 chroot and used on a kernel without scheduler activations support.
  - Replaced the default `malloc`(3) with jemalloc, bringing a significant performance boost for many threaded workloads that make heavy use of malloc.
  - `curses`(3):
    - Merged wide curses work done by Ruibiao Qui as a Google Summer of Code project.

    - Added termattrs(3) and term_attrs(3).
    - Added getwin(3) and putwin(3).
  - util(3):
    - Added estrndup(3).
    - Added raise_default_signal(3).
  - Added C99 complex support (float/double) to math(3).
  - proplib(3):
    - Implemented prop_dictionary_make_immutable(3).
    - Added prop_array_util(3) functions.
  - libc:
    - Added dehumanize_number(3).
    - Added posix_memalign(3).
    - Replaced O(nm) versions of strspn(3), strcspn(3), and strpbrk(3) with O(n+m) implementations.
    - Added getlogin_r(2).
    - Added the C99 functions imaxabs(3) and imaxdiv(3).
    - Enabled atomic_ops(3) in userspace.
    - queue(3): Added TAILQ_CONCAT() and STAILQ_CONCAT().
  - Added libgcov.
- Imported bozohttpd, a small HTTP server. See httpd(8).
- Imported dhcpcd 4.0.10.
- Imported openldap 2.4.11.
- Added rump(3), the Runnable Userspace Meta Program framework. Allows running kernel code in userspace applications.
- Added cpuctl(8), a utility that allows placing CPUs online/offline.
- Added schedctl(8), a program to control scheduling of processes and threads.
- Added psrset(8), a utility to control processor sets.
- Added atf(7), the Automated Testing Framework, Julio M. Merino Vidal's 2007 Google Summer of Code project.
- Added newgrp(1), a utility to change effective group ID.
- Added tcpdrop(8), a utility to drop tcp(4) connections.
- Added acpitools from FreeBSD: acpidump(8) and amldb(8).
- Added dkscan_bsdlabel(8) to scan disks for BSD disklabels.
- Added btkey(1), a utility to manage Bluetooth link keys in OS and device storage.

- Added svhlabel(8), a tool to update disklabel(5) from SGI Volume Header, like mbrlabel(8) for MBR labels.
- Added pcc(1) as an alternative compiler.
- Added btpand(8), a Bluetooth Personal Area Networking profile daemon.
- Added c99(1) as a wrapper to run cc(1) in C99 mode.
- Added support for Solaris $ORIGIN etc. expansions in paths.
- Added BUILDSEED support to build.sh to yield reproducible C++ builds.
- Added support for "cc -m32" on amd64 and sparc64 to generate 32 bit binaries. 32bit libraries are now provided on these two ports.
- ld(1) can now link 32bit objects on amd64.
- vi(1) has been updated to nvi 1.81, which supports internationalization. It also grew a new NetBSD-specific expandtab option.
- pkill(1): Added the -l (long format) option.
- find(1): Added the -delete and -E (extended regex) options.
- xargs(1): Replaced with FreeBSD's while keeping our GNU compatible exit values.
- sdiff(1): Replaced by OpenBSD's sdiff(1).
- pax(1): Added a -V flag for verbose summary without listing.
- top(1): Allow a single process to be selected by pid. Added a thread mode that displays LWPs.
- scsictl(8): Added a setspeed command.
- split(1): Added a new option "-n chunk_count" that splits the input into chunk_count smaller files.
- df(1): Fixed the -P option and added the -g (gigabytes) option.
- wtf(6) now searches pkgsrc's help database when called inside a package directory.
- atactl(8): Improved SATA support.
- wlanctl(8): Added a -p flag that only prints public nodes.
- btconfig(8): Added a new "rssi" option to toggle inquiry results with RSSI.
- ifconfig(8): Added "list scan" to ifconfig, which lists access points in the neighborhood.
- newsyslog.conf(5) gained a "J" flag to bzip2 logfiles.
- fdisk(8) now reports the first active partition.
- bioctl(8) was rewritten to handle new features like creating and removing hot-spares, pass-through disks and RAID volumes, start/stop consistency checks in volumes.
- savecore(8) now uses the raw device to read crashdumps.
- make(1): Implemented "-dl" (aka LOUD) to override "@" at the start of script lines.
- monop(6): The save and restore format changed, breaking compatibility with already broken previous save files.
- iconv(1) now allows SUSv3 syntax.

- `lint`(1): Added _Complex support.
- `ftp`(1): Added epsv6 and epsv to disable extended passive mode.
- `getent`(1): Added support for "netgroup" databases.
- `ypserv`(8): Disabled libwrap address to hostname lookups to avoid the chance of ypserv blocking for an extended period of time due to a long DNS timeout.
- `postfix`(1): Enabled LDAP support for tables.
- `amd`(8): Enabled LDAP support for maps.
- `newfs`(8): Added support for the "t" (terabytes) suffix.
- `grep`(1): A warning is now printed if "-r" is used without specifying an argument.
- `db`(1): Added support for encoding or decoding VIS_HTTPSTYLE, and for tuning the page size of the database.
- `daily.conf`(5): Added run_fsck_flags to allow passing extra options to the daily fsck -n.
- `sysinst`(8) now supports the Colemak and Dvorak keyboard layouts.
- `od`(1) Added support for the "-A addressformat" flag.
- `etcupdate`(8): Removed the "-b binarydir" and "-s srcdir/etc" options which were deprecated in NetBSD-4.0. Deprecate the "-s tgz1:tgz2" option; please use "-s tgz1 -s tgz2" instead.
- `postinstall`(8): Deprecated the "-s tgz1:tgz2" option; please use "-s tgz1 -s tgz2" instead.
- `sed`(1): Added the -r flag, which is an alias for -E, to be compatible with GNU sed.
- `patch`(1): Merged improved version from DragonFly. patch -b now behaves as specified by POSIX.
- `rc.conf`(5): ifconfig_xxN variables may now have multi-line values, just like /etc/ifconfig.xxN files, and semicolons may be used instead of line breaks.
- `ls`(1): -n now implies -l.
- `ps`(1): Added the -A option, to display information about all processes. Use "O" for LSONPROC like Solaris instead of bundling LSIDL, LSRUN, and LSONPROC to "R".
- `ksh`(1): Fixed POSIX mode interpretation of backslashes inside backquotes inside double quotes.
- `makefs`(8): Made the allow-multidot option for cd9660 useful.
- `restore`(8) now works on Linux dump volumes, by ignoring extended attribute records on these volumes.
- Added an `rc.d`(8) script for `rndctl`(8).
- `MAKEDEV`(8) now creates /dev rather than `init`(8).
- `MAKEDEV`(8) now uses `mtree`(8) in preference to `pax`(1) and `mknod`(8), making node creation more efficient, and `mount_tmpfs`(8) in preference to `mount_mfs`(8) when creating a memory file system.
- `MAKEDEV.local`(8) can now use functions defined in `MAKEDEV`(8).

Besides this list, there have also been innumerable bug fixes and miscellaneous enhancements.

Components removed from NetBSD

In NetBSD 5.0, the following software components were removed from the system. Some were not useful anymore, or their utility did not justify the maintenance overhead. Others were not working properly and there was a lack of interest in fixing them.

- The pc532 port.
- Support for systrace.
- HP-UX binary compatibility.
- The esl driver.
- Support for NQNFS.
- The TPCONS option in netiso.

Known Problems

Using block device nodes directly for I/O may cause a kernel crash when the file system containing `/dev` is FFS and is mounted with -o log. Workaround: use raw disk devices, or remount the file system without -o log.

Occassionally, gdb may cause a process that is being debugged to hang when "single stepped". Workaround: kill and restart the affected process.

gdb cannot debug running threaded programs correctly. Workaround: generate a core file from the program using `gcore`(1) and pass the core to gdb, instead of debugging the running program.

Statically linked binaries using pthreads are currently broken.

The sparc port does not have functional SMP support in this release.

Features to be removed in a later release

The following features are to be removed from NetBSD in the future:

- Support for soft dependencies, also knows as soft updates (see "softdep" in `mount`(8)) will be removed in the next major release.

  NetBSD 5.0.1 includes a preview of WAPBL (Write Ahead Physical Block Logging), which will replace soft dependencies in the next major release. See `wapbl`(4) and
  http://mail-index.netbsd.org/netbsd-announce/2008/12/14/msg000051.html
  for details.

- Support for Xen 2.0.x. The Xen-3 and hypervisor interface is diverging from Xen-2 as development is ongoing, increasing the maintenance cost for NetBSD.

  It should be considered as deprecated. Users are expected to not rely on it any more beyond this major release.

  Further, at least version 3.1 of Xen will be required to run NetBSD as Dom0 or DomU.

The NetBSD Foundation

The NetBSD Foundation is a tax exempt, not-for-profit 501(c)(3) corporation that devotes itself to the traditional goals and Spirit of the NetBSD Project and owns the trademark of the word "NetBSD". It supports the design, development, and adoption of NetBSD worldwide. More information on the NetBSD Foundation, its composition, aims, and work can be found at: **http://www.NetBSD.org/foundation/**

Sources of NetBSD

Refer to **http://www.NetBSD.org/mirrors/**

NetBSD 5.0.1 Release Contents

The root directory of the NetBSD 5.0.1 release is organized as follows:

`.../NetBSD-5.0.1/`

`CHANGES` Changes between the 4.0 and 5.0 releases.

`CHANGES-5.0` Changes between the initial 5.0 branch and final release of 5.0.

`CHANGES.prev` Changes in previous NetBSD releases.

`LAST_MINUTE` Last minute changes and notes about the release.

`README.files` README describing the distribution's contents.

`source/` Source distribution sets; see below.

In addition to the files and directories listed above, there is one directory per architecture, for each of the architectures for which NetBSD 5.0.1 has a binary distribution.

The source distribution sets can be found in subdirectories of the `source` subdirectory of the distribution tree. They contain the complete sources to the system. The source distribution sets are as follows:

gnusrc This set contains the "gnu" sources, including the source for the compiler, assembler, groff, and the other GNU utilities in the binary distribution sets.
82 MB gzipped, 422 MB uncompressed

sharesrc This set contains the "share" sources, which include the sources for the man pages not associated with any particular program; the sources for the typesettable document set; the dictionaries; and more.
7 MB gzipped, 29 MB uncompressed

src This set contains all of the base NetBSD 5.0.1 sources which are not in **gnusrc**, **sharesrc**, or **syssrc**.
58 MB gzipped, 284 MB uncompressed

syssrc This set contains the sources to the NetBSD 5.0.1 kernel for all architectures as well as the `config`(1) utility.
36 MB gzipped, 180 MB uncompressed

xsrc This set contains the sources to the X Window System.
133 MB gzipped, 699 MB uncompressed

All the above source sets are located in the `source/sets` subdirectory of the distribution tree.

The source sets are distributed as compressed tar files. Except for the **pkgsrc** set, which is traditionally unpacked into `/usr/pkgsrc`, all sets may be unpacked into `/usr/src` with the command:

```
# cd / ; tar -zxpf set_name.tgz
```

In each of the source distribution set directories, there are files which contain the checksums of the files in the directory:

`BSDSUM` Historic BSD checksums for the various files in that directory, in the format produced by the command:
cksum -o 1 *file*.

`CKSUM` POSIX checksums for the various files in that directory, in the format produced by the command:
cksum *file*.

`MD5` MD5 digests for the various files in that directory, in the format produced by the command:
cksum -a MD5 *file*.

`SHA512` SHA512 digests for the various files in that directory, in the format produced by the command:
cksum -a SHA512 *file*.

`SYSVSUM` Historic AT&T System V UNIX checksums for the various files in that directory, in the format produced by the command:
cksum -o 2 *file*.

The SHA512 digest is the safest checksum, followed by the MD5 digest, and finally the POSIX checksum. The other two checksums are provided only to ensure that the widest possible range of systems can check the integrity of the release files.

NetBSD/macppc subdirectory structure

The macppc-specific portion of the NetBSD 5.0.1 release is found in the `macppc` subdirectory of the distribution: `.../NetBSD-5.0.1/macppc/`. It contains the following files and directories:

`INSTALL.html`
`INSTALL.ps`
`INSTALL.txt`
`INSTALL.more` Installation notes in various file formats, including this file. The `.more` file contains underlined text using the `more`(1) conventions for indicating italic and bold display.

`binary/`
`kernel/`
`netbsd-GENERIC.gz`
A gzipped NetBSD kernel containing code for everything supported in this release.
`netbsd-GENERIC.MP.gz`
A gzipped NetBSD kernel containing code for everything supported in this release. This kernel supports SMP on systems with more than one CPU.
`netbsd-GENERIC_MD.gz`
A gzipped NetBSD kernel containing code for everything supported in this release. It also has a RAM-disk installer. This is the kernel you should use when you want to install via netboot, from a CD-R, or an HFS partition on Open Firmware 3 systems.
`netbsd-INSTALL.gz`
A gzipped NetBSD kernel containing code only for Open Firmware 1.0.5, 1.1.22, 2.0.x, and 2.4 systems (i.e. those that have floppy drives). It has a RAM-disk installer.
`sets/` macppc binary distribution sets; see below.

`installation/`
`floppy/boot1.fs`
`floppy/boot2.fs`
macppc boot and installation floppy images. The first is the bootloader and kernel. The second image has the installation tools. Only usable on mod-

els that ship with a floppy drive (pre-Open Firmware 3); see below.

`ofwboot.xcf`
macppc bootloader; see below.

Binary distribution sets

The NetBSD macppc binary distribution sets contain the binaries which comprise the NetBSD 5.0.1 release for macppc. The binary distribution sets can be found in the `macppc/binary/sets` subdirectory of the NetBSD 5.0.1 distribution tree, and are as follows:

base The NetBSD 5.0.1 macppc **base** binary distribution. You *must* install this distribution set. It contains the base NetBSD utilities that are necessary for the system to run and be minimally functional.
29 MB gzipped, 83 MB uncompressed

comp Things needed for compiling programs. This set includes the system include files (`/usr/include`) and the various system libraries (except the shared libraries, which are included as part of the **base** set). This set also includes the manual pages for all of the utilities it contains, as well as the system call and library manual pages.
44 MB gzipped, 153 MB uncompressed

etc This distribution set contains the system configuration files that reside in `/etc` and in several other places. This set *must* be installed if you are installing the system from scratch, but should *not* be used if you are upgrading.
1 MB gzipped, 1 MB uncompressed

games This set includes the games and their manual pages.
4 MB gzipped, 8 MB uncompressed

kern-GENERIC
This set contains a NetBSD/macppc 5.0.1 `GENERIC` kernel named `/netbsd`. You *must* install this distribution set.
4 MB gzipped, 8 MB uncompressed

kern-GENERIC.MP
This set contains a NetBSD/macppc 5.0.1 `GENERIC.MP` kernel, which will use multiple processors (if present), named `/netbsd`.
4 MB gzipped, 8 MB uncompressed

man This set includes all of the manual pages for the binaries and other software contained in the **base** set. Note that it does not include any of the manual pages that are included in the other sets.
12 MB gzipped, 50 MB uncompressed

misc This set includes the system dictionaries, the typesettable document set, and other files from `/usr/share`.
4 MB gzipped, 13 MB uncompressed

text This set includes NetBSD's text processing tools, including `groff`(1), all related programs, and their manual pages.
4 MB gzipped, 11 MB uncompressed

NetBSD maintains its own set of sources for the X Window System in order to assure tight integration and compatibility. These sources are based on X.Org. Binary sets for the X Window System are distributed with NetBSD. The sets are:

xbase The basic files needed for a complete X client environment. This does not include the X servers.
8 MB gzipped, 23 MB uncompressed

xcomp The extra libraries and include files needed to compile X source code.
16 MB gzipped, 55 MB uncompressed

xfont Fonts needed by the X server and by X clients.
32 MB gzipped, 74 MB uncompressed

xetc Configuration files for X which could be locally modified.
1 MB gzipped, 1 MB uncompressed

xserver The X server. This includes the modular Xorg server.
12 MB gzipped, 34 MB uncompressed

The macppc binary distribution sets are distributed as gzipped tar files named with the extension **.tgz**, e.g. `base.tgz`.

The instructions given for extracting the source sets work equally well for the binary sets, but it is worth noting that if you use that method, the filenames stored in the sets are relative and therefore the files are extracted *below the current directory*. Therefore, if you want to extract the binaries into your system, i.e. replace the system binaries with them, you have to run the **`tar -xzpf`** command from the root directory (`/`) of your system.

Note: Each directory in the macppc binary distribution also has its own checksum files, just as the source distribution does.

NetBSD/macppc System Requirements and Supported Devices

Currently, NetBSD/macppc requires the use of Open Firmware to boot. Open Firmware is a command environment using the FORTH language. The NetBSD kernel uses Open Firmware to gather information about your system and to control some of your devices. It is part of the boot ROMs in most PowerPC-based Macintosh systems. Until late 1996, Apple never intended to use Open Firmware for anything other than internal debugging and hardware support. It was not intended to be used to boot an operating system. This is why earlier machines have so much trouble with Open Firmware. This also means that PowerMacs and clones that lack Open Firmware cannot boot NetBSD on the macppc platform. Most machines introduced by Apple and the clone-makers after August 17, 1995 have Open Firmware and are supported.

Apple made several revisions of this Open Firmware environment, and each has various quirks and problems that we must work around. The single hardest step of installing NetBSD/macppc is to set up Open Firmware properly. Open Firmware versions 1.0.5 and 2.0.x act similarly and the same set of instructions applies to them. Open Firmware version 2.4 is slightly different with regards to booting. Open Firmware version 3 is altogether different, but easier to set up for NetBSD.

At present, NetBSD/macppc does not support the PPC 601 microprocessor, which means that the PowerMacintosh 7200 and 7500 models are not supported. The PowerMacintosh 7500 may be upgraded to a PPC 604, G3 or G4 microprocessor via a daughtercard replacement, in which case NetBSD will run on this system.

The minimal configuration requires 16 MB of RAM and 100 MB of disk space. To install the entire system requires considerably more, plus space for the swap partition (usually the RAM size, unless you've got a lot of RAM). To use X, 32MB of RAM is required. NetBSD with 16 MB of RAM is very slow. Until you have around 32 MB of RAM, getting more RAM is more important than getting a faster CPU.

Supported models

Find your model from the list below and take note of its Open Firmware version:

Open Firmware 1.0.5

- Apple PowerMacintosh (7300, 7500 (with CPU upgrade), 7600, 8500, 8600, 9500, and 9600)

- Apple Workgroup Server 8550
- Power Computing (PowerCenter, PowerCenter Pro, PowerCurve, PowerTower, PowerTower Pro, and PowerWave)
- UMAX (J700, S900)

Open Firmware 1.1.22

- Apple Network Server (500 and 700)

  Note: Users have reported that the NetBSD kernel does not work well with the on-board MACE-based ethernet (the one that requires an Apple AAUI dongle). If your system does not have the Apple PCI Ethernet Card, you may need to purchase and install another ethernet card in your Apple Network Server.

Open Firmware 2.0.x

- Apple PowerBook (2400, 3400, G3, and G3 Series)
- Apple PowerMacintosh/Performa (4400, 54xx, 5500, 6300/160, 6360, 6400, and 6500)
- Apple PowerMacintosh (G3 "Beige" models with ATI RageII+: Desktop, Mini Tower, and All-in-One)
- APS Tech (M∗Power 604e/200)
- Motorola StarMax (3000, 4000, 5000, and 5500)
- Power Computing (PowerBase)
- UMAX (Apus 2000, Apus 3000, C500, and C600)

Open Firmware 2.4

- Apple PowerMacintosh (G3 "Beige" models with ATI Rage Pro: Desktop, Mini Tower, and All-in-One)

Open Firmware 3

- Apple original iBook, iBook SE, iBook (Dual USB), iBook (Late 2001), iBook (16VRAM), iBook (Opaque 16 VRAM), iBook (32 VRAM), iBook (14.1 LCD 32 VRAM), and iBook G4
- Apple iMac; Bondi Blue (Rev A and Rev B, 233 MHz), 5 Flavors (Rev C and Rev D, 266 MHz and 333 MHz), iMac (Slot Loading), iMac (Summer 2000), iMac (Early 2001), iMac (Summer 2001), and iMac (USB 2.0)
- Apple original eMac and eMac (ATI Graphics)
- Apple PowerBook (G3 Series (bronze keyboard) and G3 (FireWire))
- Apple PowerBook (G4 (Titanium), G4 (Gigabit Ethernet), G4 (DVI), G4 (12-inch), G4 (17-inch), G4 (12-inch DVI), PowerBook G4 (15-inch FW 800), PowerBook G4 (17-inch 1.33GHz/1.5GHz))
- Apple PowerMacintosh G3 (Blue and White)
- Apple PowerMacintosh (G4 (PCI), G4 (AGP), G4 (Gigabit Ethernet), G4 (Digital Audio), G4 (Quicksilver), G4 (Quicksilver 2002), G4 (Mirrored Drive Doors), G4 (FW 800), and G4 Cube)
- Apple Mac Mini G4 (1.25GHz/1.42GHz/1.5GHz)
- Apple original Xserve and Xserve (Slot Load)

  Note: Some newer G4 models may not run at their full CPU speed, due to the fact Open Firmware starts the CPU at a reduced frequency.
- Apple iMac G5, PowerMacintosh G5, Xserve G5

  Note: G5 models are supported as a 32-bit port in bridge mode only.

Unsupported models

- Systems with Open Firmware, but using a PPC 601 microprocessor (PowerMacintosh 7200 and 7500 (without CPU upgrade))
- Systems with a PowerPC microprocessor, but lacking Open Firmware
  - Apple PowerBook (1400, 2300, and 5300)
  - Apple PowerMacintosh/Performa (52xx, 53xx, 62xx, and 63xx (except 6300/160 and 6360 which are supported))
  - Apple PowerMacintosh/Performa (61xx, 71xx, and 81xx)
  - PowerComputing (Power 100 and Power 120)
- Systems with broken Open Firmware (Twentieth Anniversary Macintosh)
- Systems with a Motorola 68k microprocessor (these systems are supported by NetBSD/mac68k)
- Systems upgraded from any of the above (unless the motherboard is replaced as part of the upgrade)

Supported devices

- CPU upgrades
  - Only one CPU in a multiprocessor system is supported with the GENERIC kernel. All CPUs are supported with the GENERIC.MP kernel.
  - Most CPU upgrades are supported. Some older models will not automatically enable the L2 and/or L3 caches on a G3 or G4 processor. See the FAQ entry
    http://www.NetBSD.org/ports/macppc/faq.html#cpu-upgrade
- Ethernet
  - On-board 79C950-based MACE Ethernet interface (*mc*)
  - On-board bmac Ethernet interface (*bm*)
  - On-board gmac Ethernet interface (*gem*)
  - Apple PCI Ethernet Card (option for Apple Network Server) (*tlp*)
  - Asante Mac 10/100 PCI Rev A, part number 09-00169-01 (*tlp*)
  - Farallon Fast EtherTX 10/100, part number PN996L-TX (*tlp*)
  - SMC Etherpower II (9432TX) (*epic*)
  - SMC 83c170 (*epic*)
  - 3Com 3c905 (*ex*)
  - Intel EtherExpress PRO/10+ PCI LAN Adapter (*fxp*)
  - Realtek 8029 Ethernet (*ne*)
  - VIA Technologies VT86C926 (*ne*)
  - D-Link DFE-530TX+ (*rtk*)
  - Realtek 8139 (*rtk*)
  - Netgear FA-311 (*sip*)
  - Lite-On PNIC (*tlp*)
  - D-Link DFE-530TX (*vr*)
  - Many other PCI and Cardbus Ethernet interfaces, such as Tulip-compatible (*tlp*), 3Com (*ep*), SMC (*epic*), Intel (*fxp*), NE2000-compatible (*ne*), and Realtek (*rtk*)
  - Many USB Ethernet interfaces (*aue*, *cue* and *kue*)
- Wireless
  - On-board AirPort or AirMac 802.11b (*wi*)
  - AT&T/Lucent WaveLan IEEE PCMCIA card (*wi*)
- SCSI
  - On-board NCR 53c94 SCSI controller (*esp*)

- On-board MESH SCSI controller (*mesh*)
- Adaptec PCI controllers 291x, 2920, 2930C, 294x, 295x, 39xx, 19160, 29160 and AIC-78xx (*ahc*)

  Note: The 294x models are not bootable in Open Firmware, even though they can boot MacOS 9 and earlier.

- AdvanSys 1200[A,B], 9xx[U,UA] SCSI controller (*adv*)
- AMD 53c974 (*pcscp*)
- Apple 53c875 Ultra Wide SCSI (shipped in some Beige G3 models) (*ncr* or *siop*)

  Note: This card may need a firmware update to boot NetBSD. Use the Mac OS X SCSI Card Updater from
  http://docs.info.apple.com/article.html?artnum=25176
- NCR/Symbios 53C8xx (*siop* or *esiop*)
- Many other PCI SCSI controllers should work, but no one has tried them
- Most SCSI disk/tape/CD-ROM devices should work

- IDE
  - On-board IDE controllers
  - Promise Ultra66 (*pciide*) (this, and other PC-based cards are not bootable on macppc)
  - Acard ATP-850/860 based IDE controllers (*pciide*) (including the Acard AEC-6210/6260 and the Sonnet Tempo ATA/66 cards)
  - Some other PCI IDE controllers should work, although no one has had much success
  - Most IDE disk/CD-ROM/ATAPI devices should work

- Input devices
  - Most ADB keyboards, mice, trackballs, and trackpads
  - Most USB keyboards, mice, trackballs, and trackpads
  - Most PS/2 keyboards, mice, and trackballs (middle button on 3-button mice may not work)

- Video
  - On-board video on most models (*ofb*)

    Note: Several models have been reported to not work with NetBSD if the on-board video is in use, such as the Performa 54xx, 6360, 6400, PowerComputing PowerCenter and PowerTower, and UMAX C600 and Apus 3000. Also, the video card that shipped with the PowerMacintosh 9500/150, 9500/180, and 9500/200 systems is reported to have the same problem.

  - PCI frame buffers which have Open Firmware support (ATI, IMS, Matrox, and NVIDIA have several models which work)
  - Full wscons capabilities, such as virtual terminals and color text are available. (Note however that switching between virtual terminals while running an X session is not supported at present.)

- Audio
  - On-board audio on most models (*awacs* or *snapper*)

    Note: The (*awacs*) driver is considered "experimental ", meaning that it may cause your system to crash and/or is lacking some features you may expect. Due to interrupt conflicts with some G4 models, this driver is disabled by default and you will need to build a kernel with it enabled.

- USB audio devices (*uaudio*)
- PCI audio cards, although only the Ensoniq AudioPCI (*eap*) has been thoroughly tested

- Serial ports
  NetBSD/macppc refers to the on-board serial ports as (*tty00*) and (*tty01*). Firmware uses the names listed below.
  - On-board serial ports (the modem and printer ports) (*ttya* and *ttyb*)
    ttya is the mini-DIN 8 serial port with the 'Modem' icon, and *ttyb* is has the 'Printer' icon. Some models with internal modems have the serial port with the 'Modem' icon covered over, and the modem is *ttya*.

    Note: The on-board serial ports can be used for console (38400 bps, 8 bits, no parity, 1 stop bit, no handshaking), although many users have reported problems trying to run ppp or other high speed serial applications
  - The Xserve serial port.

    Note: This serial port can be used for console (57600 bps, 8 bits, no parity, 1 stop bit, no handshaking). Use (*scca*) for the Open Firmware 'input-device' and 'output-device' variables. Alternatively, if you remove your video card, Open Firmware will default to using the serial port for the console.
  - The Gee Three Stealth Serial Port, possibly the Griffin Technology gPort

    Note: These serial ports can be used for console (57600 bps, 8 bits, no parity, 1 stop bit, no handshaking). Use (*scca*) for the Open Firmware 'input-device' and 'output-device' variables.
  - Some USB, PCI, and Cardbus serial ports should work, but no one has tried them
- USB devices
  - Most MI USB devices should work (such as disks, printers, input devices, SCSI adapters, and ethernet interfaces)
    http://www.NetBSD.org/support/hardware/usb.html
- PCMCIA and Cardbus cards
  - Most MI PCMCIA and Cardbus cards should work, although very few have been tested with NetBSD/macppc
    http://www.NetBSD.org/support/hardware/cardbus.html
    http://www.NetBSD.org/support/hardware/pcmcia.html
- PCI cards
  - Most MI PCI cards should work, although very few have been tested with NetBSD/macppc
    http://www.NetBSD.org/support/hardware/pci.html

  Note: While the NetBSD kernel may support the various Cardbus, PCI, PCMCIA, and USB devices you may have, Open Firmware does *not* unless it has a specific Open Firmware ROM. This means you cannot boot from these devices. Some Adaptec SCSI controllers have bootable Open Firmware ROMs.

Unsupported devices

- Floppy disk

  Note: Though NetBSD/macppc can boot from a floppy, there is no kernel support yet

- Advanced power management (cannot put system to 'sleep')
- On-board video capture
- AirPort Extreme
- On-board SATA controller on G5 systems (these machines must be netbooted)

Supported boot devices and media

Each version of Open Firmware supports different devices and media that you may boot from. We define the terms "bootable media" as the media (hard drive, floppy, CD-R, ethernet) that will be used to bootstrap your macppc system into NetBSD, and "distribution sets" or "distribution media" as the media (hard drive, CD-R, ethernet) that contains the files that will be installed to generate a working NetBSD system onto your destination media.

Go to the NetBSD/macppc Model Support webpage and look up your system. Take note of the comments about your model and keep these in mind during the rest of this installation procedure.

http://www.NetBSD.org/ports/macppc/models.html

Getting the NetBSD System on to Useful Media

You should wait to decide where to put the NetBSD distribution sets until you have figured out how you are going to boot your system. Refer back to this section after you have done so.

Note: Some Mac OS **ftp** clients default to downloading files in 'ASCII' mode. This will render the NetBSD files useless. Make sure to set your ftp program to download in 'binary' mode.

Note that if you are installing or upgrading from writable media, it can be write-protected if you wish. These systems mount a root image from inside the kernel, and will not need to write to the media. If you booted from a floppy, the floppy disk may be removed from the drive after the system has booted.

Installation is supported from several media types, including:

- CD-ROM / DVD
- MS-DOS floppy
- FTP
- Remote NFS partition
- Tape
- Existing NetBSD or MacOS X UFS partitions, if performing an upgrade

The steps necessary to prepare the distribution sets for installation depend upon which installation medium you choose. The steps for the various media are outlined below.

CD-ROM / DVD Find out where the distribution set files are on the CD-ROM or DVD. Likely locations are `binary/sets` and `macppc/binary/sets`.

Proceed to the instructions on installation.

MS-DOS floppy NetBSD does not include split distribution sets for installation by floppy. However, they can be created on a separate machine using the `split`(1) command, running e.g. **`split -b 235k base.tgz base.`** to split the `base.tgz` file from `macppc/binary/sets` into files named `base.aa`, `base.ab`, and so on. Repeat this for all `set_name.tgz` files, splitting them into `set_name.`*xx* files. Count the number of `set_name.`*xx* files that make up the distribution sets you want to install or upgrade. You will need one sixth that number of 1.44 MB floppies.

Format all of the floppies with MS-DOS. Do *not* make any of them bootable MS-DOS floppies, i.e. don't use `format /s` to format them. (If the floppies are bootable, then the MS-DOS system files that make them bootable will take up some space, and you

won't be able to fit the distribution set parts on the disks.) If you're using floppies that are formatted for MS-DOS by their manufacturers, they probably aren't bootable, and you can use them out of the box.

Place all of the `set_name.`*`xx`* files on the MS-DOS disks.

Once you have the files on MS-DOS disks, you can proceed to the next step in the installation or upgrade process. If you're installing NetBSD from scratch, go to the section on preparing your hard disk, below. If you're upgrading an existing installation, go directly to the section on upgrading.

FTP

The preparations for this installation/upgrade method are easy; all you need to do is make sure that there's an FTP site from which you can retrieve the NetBSD distribution when you're about to install or upgrade. If you don't have DHCP available on your network, you will need to know the numeric IP address of that site, and, if it's not on a network directly connected to the machine on which you're installing or upgrading NetBSD, you need to know the numeric IP address of the router closest to the NetBSD machine. Finally, you need to know the numeric IP address of the NetBSD machine itself. If you don't have access to a functioning nameserver during installation, the IPv4 address of **ftp.NetBSD.org** is `204.152.190.13` and the IPv6 address is `2001:4f8:3:7:230:48ff:fe31:43f2` (as of April, 2009).

Once you have this information, you can proceed to the next step in the installation or upgrade process. If you're installing NetBSD from scratch, go to the section on preparing your hard disk, below. If you're upgrading an existing installation, go directly to the section on upgrading.

NFS

Place the NetBSD distribution sets you wish to install into a directory on an NFS server, and make that directory mountable by the machine on which you are installing or upgrading NetBSD. This will probably require modifying the `/etc/exports` file on the NFS server and resetting its mount daemon (mountd). (Both of these actions will probably require superuser privileges on the server.)

You need to know the numeric IP address of the NFS server, and, if you don't have DHCP available on your network and the server is not on a network directly connected to the machine on which you're installing or upgrading NetBSD, you need to know the numeric IP address of the router closest to the NetBSD machine. Finally, you need to know the numeric IP address of the NetBSD machine itself.

Once the NFS server is set up properly and you have the information mentioned above, you can proceed to the next step in the installation or upgrade process. If you're installing NetBSD from scratch, go to the section on preparing your hard disk, below. If you're upgrading an existing installation, go directly to the section on upgrading.

Tape

To install NetBSD from a tape, you need to make a tape that contains the distribution set files, in 'tar' format.

If you're making the tape on a UNIX-like system, the easiest way to do so is probably something like:

```
# tar -cf tape_device dist_directories
```

where *`tape_device`* is the name of the tape device that describes the tape drive you're using; possibly `/dev/rst0`, or something similar, but it will vary from system to system. (If you can't figure it out, ask your system administrator.) In the above example, *`dist_directories`* are the distribution sets' directories, for the distribution sets you wish to place on the tape. For instance, to put the **kern-GENERIC, base,**

and **etc** distributions on tape (in order to do the absolute minimum installation to a new disk), you would do the following:

```
# cd .../NetBSD-5.0.1
# cd macppc/binary
# tar -cf tape_device kern-GENERIC base etc
```

Note: You still need to fill in *tape_device* in the example.

Once you have the files on the tape, you can proceed to the next step in the installation or upgrade process. If you're installing NetBSD from scratch, go to the section on preparing your hard disk, below. If you're upgrading an existing installation, go directly to the section on upgrading.

Preparing your System for NetBSD installation

Prepare yourself

Take a deep breath.

Good. Now, make sure you are reading the PostScript or HTML version of this document, as the `.txt` and `.more` versions lack important formatting information that will prevent you from following the twisted path documented below. OK, good. Now, print out this document.

While it's printing, get some coffee, relax a bit, and mentally prepare yourself for something that promises to be confusing, frustrating, and annoying. If you assume the worst, you'll be pleasantly surprised when everything works easier than you expected. Also, forget everything you've been told about installing NetBSD/macppc. That's right, flush your knowledge cache -- some of it is almost certainly dirty.

Some of this document assumes familiarity with MacOS, e.g. how to download BinHex files and extract things with StuffIt Expander. If you've never heard of those terms before, it is possible to install NetBSD/macppc without booting or knowing how to use MacOS, but depending on your model it may be almost as painful as learning a little MacOS. See the FAQ for more help:

http://www.NetBSD.org/ports/macppc/faq.html#macos-newbie

It's done printing? Fine, time to get started.

The recommended installation procedure is as follows:

1. Go to the NetBSD/macppc model support page and look up your model information and issues (I can't stress this enough times).
   http://www.NetBSD.org/ports/macppc/models.html
2. Spend about 15 minutes and read through this document from this point onward. There will be many pages of instructions that apply to Open Firmware versions that you do not have. Throw them out. Do not read them — at best you will be confused, at worst you may damage your system, requiring repairs.
3. Now create your bootable media and media for the distribution sets.
4. Prepare your machine, depending on the instructions for your model (for example: partition your drive, download and run System Disk, or hook up a serial console).
5. Boot into Open Firmware and verify that it has the version of Open Firmware you think it does. Also make sure that the other variables are set correctly.
6. Figure out the correct boot command and boot your machine from the bootable media you just created. If you're having trouble, be sure to read the section on **Common Problems and Error Messages**

7. Celebrate! The worst is over, but you've still got some work to do. Take a break, maybe more coffee, maybe a quick walk around the block, whatever turns your fancy.
8. Now use the installer to install the distribution sets onto your system and do some initial configuration.
9. Figure out how to boot from the installed partition. Boot into NetBSD for the first time.
10. You may then boot into Open Firmware and set it to always boot your favorite operating system.
11. Configure to your preferences, install your favorite packages, and have fun with your new NetBSD/macppc machine!

Note: You really actually truly do need to follow the procedure listed in this document in the order that we describe. These systems are rather tricky to boot for the novice and expert alike. Once you cross off the sections that don't apply to you, it will make more sense.

Note: If the instructions in a subsection below do not apply to *all* versions of Open Firmware. There will be a line listing which versions of Open Firmware they apply to, such as:
(Open Firmware 1.0.5, Open Firmware 1.1.22, Open Firmware 2.0.x, Open Firmware 2.4, Open Firmware 3)

If you do not have an Open Firmware 3 system, skip down to **Preparing your Open Firmware 1.x or 2.x System for NetBSD**

Preparing your Open Firmware 3 System for NetBSD

Updating your BootROM

Open Firmware 3 systems have a rewritable "firmware", also called the BootROM. When you use an Apple firmware updater, it updates the BootROM. This will *not* change the version of Open Firmware in your machine — it will still be Open Firmware 3. The BootROM is what is first executed when you power on or reset your system. The BootROM then loads Open Firmware, which boots your operating system.

Go to the 'Apple Support Downloads' web site at **http://support.apple.com/downloads/** and search for 'firmware' and install the most recent version for your model. For most G3 and G4 models, you will need to run the FirmWare updater from MacOS 9.

Note: If you accidentally change the **`load-base`** or **`real-base`** Open Firmware variables and reset your machine you will, in effect, rewrite the BootROM with garbage. This will permanently damage your machine. We recommend not doing this.

Note: The most recent BootROMs available (4.1.7 and later) are a little picky about RAM. Initially, some PowerMacintosh G3 users found that their third-party RAM had been disabled, but the RAM vendors brought their RAM up to spec and it hasn't been much of an issue since then.

There is one report that FirmWare Update 4.1.9 on iMac (Summer 2000) will prevent the CD-ROM and the hard drive from operating together. You may get `wdc0:0:1: lost interrupt` problems.

Getting to Open Firmware 3

Hold down a special four-key combination when your system boots.

After the chime starts, but before it stops, hold down the four `COMMAND-OPTION-O-F` keys (the `COMMAND` key looks like a four-leaf clover or an open apple, and the `OPTION` key may look like a two-way switch with four straight line segments or say `ALT`) until you see the Open Firmware command prompt on your screen:

```
Apple PowerBook3,1 2.1f1 BootROM built on 01/29/00 at 22:38:07
Copyright 1994-2000 Apple Computer, Inc.
All Rights Reserved
```

```
Welcome to Open Firmware.
To continue booting, type "mac-boot" and press return.
To shut down, type "shut-down" and press return.
 ok
0 >
```

Now, set your system to always stop at the Open Firmware prompt.

```
0 > setenv auto-boot? false
```

Alternatively, if you are currently running MacOS X or Darwin, you can use the **nvram** command to set this variable before rebooting.

```
# nvram auto-boot\?=false
```

You will need to escape the question-mark or enclose the whole **nvram** argument in double-quotes to prevent your shell from trying to interpret it.

Setting up Open Firmware 3 to boot NetBSD

This section describes some steps you must take to prepare Open Firmware to boot NetBSD. Additional resources are available in the FAQ regarding how to use the Open Firmware command environment, and the Open Firmware variables you may be using:

http://www.NetBSD.org/ports/macppc/faq.html#ofw-access
http://www.NetBSD.org/ports/macppc/faq.html#ofw-variables

Double-check your Open Firmware version:

```
0 > dev /openprom
0 > .properties
name                    openprom
device_type             BootROM
model                   OpenFirmware 3
relative-addressing
supports-bootinfo
 ok
```

If you will be netbooting your system, you can look up your MAC address.

```
0 > dev enet
0 > .properties
[...]
local-mac-address   CCCCCCCC CCCC
[...]
```

Note: Some Open Firmware 3 machines have their MAC address stored incorrectly (little- vs. big-endian problem). If you look up your MAC address in MacOS, it will be different than what Open Firmware 3 uses to contact your netboot server. Your machine will still work, but its MAC address may conflict with another ethernet device on your network.

You can check your Open Firmware settings with the `printenv` command:

```
0 > printenv
-------------- Partition: common -------- Signature: 0x70 ---------------
little-endian?      false               false
real-mode?          false               false
auto-boot?          false               true
diag-switch?        false               false
[...]
```

```
use-nvramrc?        true                false
real-base           -1                  -1
[...]
input-device        keyboard            keyboard
output-device       screen              screen
```

Available Boot Media

Open Firmware is capable of booting from a variety of media (such as hard drives, CD-ROMs, and ethernet). Open Firmware is able to boot files from a variety of file systems (such as ISO9660, HFS, HFS+, and MS-DOS FAT). Unfortunately, Open Firmware is *not* able to directly boot from the NetBSD file system (FFS) or Apple's BSD-based file system (UFS), so we must put the bootloader in a location that Open Firmware is capable of understanding.

Therefore, to boot the NetBSD kernel, Open Firmware must first load a 'bootloader' (`ofwboot.xcf`) which knows how to load the NetBSD kernel.

Note: `ofwboot.elf` is obsoleted. All users should be using `ofwboot.xcf` instead of `ofwboot.elf` now.

The following bootable media are available for loading the bootloader:

- *HFS or HFS+ file system*

  This method loads `ofwboot.xcf` from an HFS or HFS+ partition which then loads the kernel from an acceptable location. The kernel (compressed or non-compressed) may be on an HFS or HFS+ partition. Due to Open Firmware pickiness, it must be on the same partition as `ofwboot.xcf`.

- *MS-DOS file system*

  In this method, Open Firmware loads the `ofwboot.xcf` bootloader from an MS-DOS file system. It may then load a NetBSD kernel from the same MS-DOS file system. This has only been thoroughly tested on floppy disks, but may work on Zip disks or FDISK-formatted hard drives. This does not work for MS-DOS partitions on a hard drive with an Apple Partition Map.

- *Ethernet (network boot)*

  You can run your entire system diskless or netboot only the files necessary to boot (i.e. the bootloader and the installation kernel). You must have root access for the UNIX-like netboot server, which must be on the same subnet as your NetBSD/macppc machine.

  Note: You must use your on-board ethernet device for netbooting. While there may be some PCI or Cardbus ethernet interfaces with Open Firmware ROMs, no one has reported success netbooting using these devices

- *Hybrid HFS/ISO9660 CD-ROM file system*

  Load the `ofwboot.xcf` bootloader from the CD-ROM's HFS file system. It can then load a NetBSD kernel from the HFS or ISO9660 file system.

  Make sure that the CD has an Apple Partition Map.

  Note: Open Firmware can only open files on the first session of a multi-session CD-R

Once the bootloader is loaded, it can open the kernel from one of the following sources:

- *Ethernet (NFS)*

- *NetBSD FFS file system*
- *MacOS X UFS file system*
- *HFS or HFS+ file system*
- *ISO9660 CD-ROM file system*
- *MS-DOS file system*

The boot CD image provided has both a 'partition zero' bootloader and `ofwboot.xcf` on a hybrid partition so it can be booted on all Open Firmware versions. It also has an ISO9660 file system with an installation kernel and the distribution sets.

Partitioning your hard drive for NetBSD

You must have at least one disk that was partitioned before running the NetBSD installer.

This is the drive that will have the bootloader, `ofwboot.xcf`. Your NetBSD partitions may either be on this same disk (using the method described in the rest of this section), or on a separate disk accessible only to NetBSD.

This section describes how to make a single disk usable by both MacOS and NetBSD — this is necessary for machines which have only one hard drive (such as the eMac, iBook, iMac, PowerBook, and PowerMacintosh Cube systems). If you do not want MacOS you must still follow this procedure, but create only a small HFS+ partition (large enough for the bootloader).

There are two partitioning tools available for NetBSD/macppc, `disklabel`(8) and `pdisk`(8). The former is used in the NetBSD **`sysinst`** installer, and will render your disk unusable by MacOS. It will also prevent Open Firmware 3 machines from booting from that disk. When running the installer, you will need to use the installer's "Re-install sets" option to skip the **`disklabel`** step.

Do not use **`disklabel`** or "Re-install sets" unless you will use one drive for NetBSD only and have another drive which will have the bootloader.

The process is more fully detailed in the Partitioning HOW-TO:
http://www.NetBSD.org/ports/macppc/partitioning.html

You can create a partition map with `pdisk`(8), but the disk will not be usable with MacOS 9 and earlier. If this is a concern, you will need to use Apple's Drive Setup or Disk Utility.

If you are using Apple's Drive Setup tool, make sure you have version 1.8.1 or later. This tool only runs under MacOS 9 and earlier. Drive Setup will erase the contents of your drive — it does not preserve data from any of your partitions.

Apple's Disk Utility only runs under MacOS X 10.0.0 and later. Make sure you click the "Install Mac OS 9 Disk Drivers" checkbox. Also, keep in mind that Disk Utility does *not* create the partitions that NetBSD/macppc requires. After creating the initial partition map with Disk Utility, you will need to use the NetBSD **`pdisk`** to change the partition types. Also, Disk Utility will erase the contents of your drive — it does not preserve data from any of your partitions.

`pdisk` is the most flexible (and most difficult to use) partitioning tool available. It runs on almost all OSes that macppc machines support. Download it:
ftp://ftp.NetBSD.org/pub/NetBSD/arch/macppc/netbsd-pdisk/
ftp://ftp.NetBSD.org/pub/NetBSD/arch/macppc/macos-utils/pdisk.sea.hqx

There is built-in help describing how it works. When it asks you to enter the "`Type of partition`", use `Apple_UNIX_SVR2` for NetBSD partitions, `Apple_HFS` for HFS and HFS+ partitions, and `Apple_UFS` for UFS partitions.

After you've written the partition map with **pdisk**, you will need to create the file systems. Use newfs(8) and mount(8) for NetBSD file systems, and **hfstools** to create and mount HFS file systems.

ftp://ftp.NetBSD.org/pub/NetBSD/packages/pkgsrc/sysutils/hfsutils/README.html

Make the following partitions:

HFS or *HFS+*	Must be large enough to hold the bootloader, over 100 KB. May be as large as you desire for MacOS usage.
A/UX Root	Must be at least 20 MB. Alternatively, you may decide to use one partition for your entire NetBSD installation, in which case it should be at least 200 MB. NetBSD interprets an *A/UX Root* partition as the first partition (*a*) on the disk. This partition is not readable from MacOS.
A/UX Swap	Any size. The recommendation is your RAM size, although this is not strictly necessary for machines with a lot of RAM. NetBSD interprets an *A/UX Swap* partition as the second partition (*b*) on the disk. This partition is not readable from MacOS.
A/UX User and *A/UX Free1*	Use these for any additional partitions you may want to use under NetBSD, such as /usr (at least 200 MB), /home, /usr/local, or /usr/pkg. NetBSD interprets these partitions as normal NetBSD-style partitions. These partitions are not readable from MacOS.
HFS	Any size. You may want to leave an additional partition available to transfer files between MacOS and NetBSD. If would like to create such a partition, then see the Partitioning HOW-TO. **http://www.NetBSD.org/ports/macppc/partitioning.html#msdos**
UFS	Any size. *UFS* partitions are not readable from MacOS versions prior to X 10.0.0. If you use an *UFS* partition as your root, then it may not be recognized by the NetBSD kernel as the first partition (*a*) on the disk. You will need to compile a new kernel with the root partition explicitly defined to be the *UFS* partition.

Now would be a good time to use **pdisk** to determine the partition numbers for your bootloader and kernel.

Preparing the Open Firmware 3 Bootable Media

The purpose of this section is to prepare the media from which your system will boot the installer. We'll describe how to put the files in the right places on your disk(s) or netboot server and prepare it for use on your system.

If you will be running your system diskless (i.e. entirely over NFS, not using any local hard drives), then you do not need to run the installer, you only need to extract the distribution sets on the diskless server.

To get the distribution sets onto appropriate media, see the above section entitled **Getting the NetBSD System on to Useful Media**. You may want to get the distribution sets when you create the bootable media.

Note: Some MacOS **ftp** clients default to downloading files in 'ASCII' mode. This will render the NetBSD files useless. Make sure to set your ftp program to download in 'binary' mode.

What follows are the steps to create different types of bootable media for the NetBSD install kernel. You should only need to create one of these to get your system to boot the installer

- *Creating the NetBSD/macppc CD-R*

  Go to one of the NetBSD mirror sites and download the CD-R image `macppccd-5.0.1.iso` from the `pub/NetBSD/iso` directory. **http://www.NetBSD.org/mirrors/#iso**

  – *From an UNIX-like machine*

    Get and install cdrecord. NetBSD users should install it from the package collection. Other UNIX-like systems should get it from the official website:
    ftp://ftp.NetBSD.org/pub/NetBSD/packages/pkgsrc/sysutils/cdrtools/README.html
    http://www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/cdrecord.html

    ```
    # cdrecord -v speed=4 dev=/dev/cd1c macppccd-5.0.1.iso
    ```

    You will need to substitute the correct name of the disk image file, speed for your CD writer, and the correct device for your system (for i386 it would be `/dev/cd1d`).

  – *From a MacOS machine (using Toast)*

    1. Click the 'Other' button in the main window.
    2. Open the contextual menu on the 'Other' button and select 'Disk Image'
    3. Click the 'Select' button and select the disk image you downloaded.
    4. Click the 'Record' button

  Skip forward to **Installing the NetBSD System**

- *Preparing an HFS or HFS+ partition*

  Place `ofwboot.xcf` and the installation kernel `netbsd-GENERIC_MD.gz` at the top level of your Macintosh file system. That is, drag the two icons onto your hard drive icon (not the desktop). Make sure they're both on the same partition.

  Skip forward to **Installing the NetBSD System**

- *Preparing a custom hybrid HFS/ISO9660 CD-R*

  This section describes how to create your own bootable NetBSD/macppc CD-R. We recommend that you use the official NetBSD/macppc−5.0.1 CD-R image, as described in the section above.

  Place `ofwboot.xcf` and the installation kernel `netbsd-GENERIC_MD.gz` at the top level of the CD. Make sure that the bootloader is present on the HFS partition, and the kernel is on both the ISO9660 and HFS partitions. Due to restrictions in the way that Open Firmware deals with ISO filenames, you may wish to name your kernel `netbsd.gz`. You may also place the NetBSD 5.0.1 distribution sets on the ISO9660 partition (not only on the HFS partition).

  – *From an UNIX-like machine*

    Get and install mkisofs. This is now part of the cdrecord package. NetBSD users should install it from the package collection. Other UNIX-like systems should get it from the official website:
    ftp://ftp.NetBSD.org/pub/NetBSD/packages/pkgsrc/sysutils/cdrtools/README.html
    http://www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/cdrecord.html

    ```
    # mkisofs -o output.iso -hfs -part -l -J -r -L -N /cdsources
    # cdrecord -v speed=4 dev=/dev/cd1c output.iso
    ```

    You will need to substitute the correct speed for your CD writer, and the correct device for your system (for i386 it would be `/dev/cd1d`).

See the NetBSD Bootable CD-ROM HOW-TO for more detail:
http://www.NetBSD.org/docs/bootcd.html

– *From a MacOS machine (using Toast)*

1. Click the 'Data' button in the main window
2. Pick 'Mac OS and PC (Hybrid) CD' from the popup menu between the four buttons and the file and folder area.
3. Drag `installation/ofwboot.xcf` and `binary/kernel/netbsd-GENERIC_MD.gz` into the window. You may also want to drag the NetBSD 5.0.1 distribution sets to this window as well.
4. Click the 'Record' button

Skip forward to **Installing the NetBSD System**

• *Creating an MS-DOS disk*

Open Firmware 3 cannot boot from a floppy disk, but you can still create a bootable Zip disk with this method.

1. First, make sure you have done a low-level format on the disk. A bad disk can produce Open Firmware errors similar to other problems.
2. Mount the disk on your computer.
   – Insert it into a Windows or DOS machine.
   – Use 'File Exchange or 'PC Exchange'' with MacOS versions prior to X 10.0.0
   – Use `mount_msdos`(8) on any UNIX-like machine.
3. Copy `netbsd-INSTALL.gz` and `ofwboot.xcf` to the disk.
4. Rename `netbsd-INSTALL.gz` to `netbsd.gz`, since OpenFirmware can't deal with MS-DOS filenames longer than eight characters.

Skip forward to **Installing the NetBSD System**

• *Preparing the netboot server*

1. Introduction

   To netboot a macppc, you must configure one or more servers to provide information and files to your macppc (the 'client'). If you are using NetBSD (any architecture) on your netboot server(s), the information provided here should be sufficient to configure everything. Additionally, you may wish to look at the `diskless`(8) manual page and the manual pages for each daemon you'll be configuring. If the server(s) are another operating system, you should consult the NetBSD Diskless HOW-TO, which will walk you through the steps necessary to configure the netboot services on a variety of platforms.
   http://www.NetBSD.org/docs/network/netboot/

   You may either netboot the installer so you can install onto a locally attached disk, or you may run your system entirely over the network.

   Briefly, the netboot process involves discovery, bootstrap, kernel and file system stages. In the first stage, the client discovers information about where to find the bootstrap program. Next, it downloads and executes the bootstrap program. The bootstrap program goes through another discovery phase to determine where the kernel is located. The bootstrap program tries to mount the NFS share containing the kernel. Once the kernel is loaded, it starts executing. For RAM disk kernels, it mounts the RAM disk file system and begins executing the installer from the RAM disk. For normal (non-RAM disk) kernels, the kernel tries to mount the NFS share that had the kernel and starts

executing the installation tools or `init`(8). All macppc systems use BOOTP for the discovery stage. TFTP is used in the bootstrap phase to download the bootstrap program, `ofwboot.xcf`. NFS is used in both the kernel and file system stages to download the kernel, and to access files on the file server.

We will use '`CC:CC:CC:CC:CC:CC`' as the MAC address (ethernet hardware address) of your netboot client machine. You should have determined this address in an earlier stage. In this example, we will use '`192.168.1.10`' as the IP address of your client and '`client.test.net`' as its name. We will assume you're providing all of your netboot services on one machine called '`server.test.net`' with the client's files exported from the directory `/export/client/root`. You should, of course, replace all of these with the names, addresses, and paths appropriate to your environment.

You should set up each netboot stage in order (i.e., discovery, bootstrap, kernel, and then file system) so that you can test them as you proceed.

2. `dhcpd`(8) in `bootpd`(8) compatible mode

   Put the following lines in your `/etc/dhcpd.conf` (see `dhcpd.conf`(5) and `dhcp-options`(5) for more information):

```
ddns-update-style none;
                # Do not use any dynamic DNS features
                #
allow bootp;    # Allow bootp requests, thus the dhcp server
                # will act as a bootp server.
                #
authoritative;  # master DHCP server for this subnet
                #
subnet 192.168.1.0 netmask 255.255.255.0 {
                # Which network interface to listen on.
                # The zeros indicate the range of addresses
                # that are allowed to connect.
}
group {
                # Set of parameters common to all clients
                # in this "group".
                #
        option broadcast-address        192.168.1.255;
        option domain-name              "test.net";
        option domain-name-servers      dns.test.net;
        option routers                  router.test.net;
        option subnet-mask              255.255.255.0;
                #
                # An individual client.
                #
        host client.test.net {
                hardware ethernet       CC:CC:CC:CC:CC:CC;
                fixed-address           192.168.1.10;
                #
                # Name of the host (if the fixed address
                # doesn't resolve to a simple name).
                #
                option host-name        "client";
```

```
                        #
                        # Name of the bootloader or kernel
                        # to download via tftp.
        #
                        filename                "ofwboot.xcf";

                        #
                        # The path on the NFS server.
                        #
                        option root-path "/export/client/root";

                        #
                        # If your DHCP server is not your NFS server, supply the
                        # address of the NFS server. Since we assume you run every
                        # on one server, this is not needed.
        #
                        # macppc machines will look for their bootloader,
                        # ofwboot.xcf, on the next-server as well.
                        #
                        # next-server server.test.net;
                }
         #you may paste another "host" entry here for additional
         #clients on this network
        }
```

You will need to make sure that the `dhcpd.leases` file exists.

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

You will need to start the **dhcpd**. If it's already running, you will need to restart it to force it to re-read its configuration file. If the server is running NetBSD, you can achieve this with:

```
# /etc/rc.d/dhcpd restart
```

3. `tftpd`(8)

   The default configuration of the TFTP server is to run in a `chroot`(8) environment in the `/tftpboot` directory. Thus, the first order of business is to create this directory:

   ```
   # mkdir -p /tftpboot
   ```

   Next, edit `/etc/inetd.conf` and uncomment the line with the TFTP daemon:

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

   Now, restart `inetd`(8). If the server is running NetBSD, you can achieve this with:

   ```
   # /etc/rc.d/inetd restart
   ```

   Now, you need to copy the bootloader for your macppc machine to `/tftpboot`. Get `ofwboot.xcf`(8) from the `installation` directory of the distribution.

   ```
   # cp ofwboot.xcf /tftpboot
   ```

   Just to be sure, let's make everything readable.

   ```
   # chmod -R a+rX /tftpboot
   ```

   Sometimes, the `arp`(8) table gets messed up, and the TFTP server can't communicate with the client. In this case, it will write a log message (via `syslogd`(8)) to `/var/log/messages` say-

ing: '`tftpd: write: Host is down`'. If this is the case, you may need to force the server to map your client's ethernet address to its IP address:

```
# arp -s client CC:CC:CC:CC:CC:CC
```

4. `nfsd`(8), `mountd`(8), and `rpcbind`(8)

   Now your system should be able to load the bootstrap program and start looking for the kernel. Let's set up the NFS server. Create the directory you are exporting for the netboot client:

   ```
   # mkdir -p /export/client/root
   ```

   Put the following line in `/etc/exports` to enable NFS sharing:

   /export/client/root -maproot=root client.test.net

   If your server is currently running an NFS server, you only need to restart `mountd`(8). Otherwise, you need to start `rpcbind`(8) and `nfsd`(8). If the server is running NetBSD, you can achieve this with:

   ```
   # /etc/rc.d/rpcbind start
   # /etc/rc.d/nfsd start
   # /etc/rc.d/mountd restart
   ```

5. NetBSD kernel and installation tools

   Now, if you place a kernel named `netbsd` in `/export/client/root` your client should boot the kernel. If you are netbooting the installer, use `binary/kernel/netbsd-GENERIC_MD.gz` (this has the installation tools in a RAM disk). Also, copy the distribution files to the client's root directory.

   Note: Some combinations of Open Firmware version and `ofwboot.xcf` version have trouble loading compressed kernels. If you have trouble loading a kernel, try uncompressing it with `gunzip`(1)

   ```
   # cp *tgz /export/client/root
   # gunzip netbsd-GENERIC_MD.gz
   # mv netbsd-GENERIC_MD /export/client/root/netbsd
   ```

   If you are running your macppc diskless, simply use `binary/kernel/netbsd-GENERIC.gz`.

6. Client file system

   You can skip this step if you do not plan to run your client diskless after installation. Otherwise, you need to extract and set up the client's installation of NetBSD. The Diskless HOW-TO describes how to provide better security and save space on the NFS server over the procedure listed here. See
   http://www.NetBSD.org/docs/network/netboot/nfs.html
   for details.

   – Extracting distribution sets

     ```
     # cd /export/client/root
     # tar -xpzf /path/to/files/base.tgz
     # tar -xpzf /path/to/files/etc.tgz
     ```

     Continue with the other non-essential distribution sets if desired.

   – Set up swap

     ```
     # mkdir /export/client/root/swap
     # dd if=/dev/zero of=/export/client/swap bs=4k count=4k
     # echo '/export/client/swap -maproot=root:wheel client.test.net' | cat >>
     ```

```
# /etc/rc.d/mountd restart
```

This creates a 16 MB swap file and exports it to the client.

– Create device nodes

```
# cd /export/client/root/dev
# ./MAKEDEV all
```

This procedure only works on NetBSD hosts.

– Set up the client's `fstab`(5)

Create a file in `/export/client/root/etc/fstab` with the following lines:

```
server:/export/client/swap   none  swap  sw,nfsmntpt=/swap
server:/export/client/root   /     nfs   rw 0 0
```

– Set up the client's `rc.conf`(5)

Edit `/export/client/root/etc/rc.conf`

```
rc_configured=YES
hostname="client"
defaultroute="192.168.1.1"
nfs_client=YES
auto_ifconfig=NO
net_interfaces=""
```

Make sure rc does not reconfigure the network device since it will lose its connection to the NFS server with your root file system.

– Set up the client's `hosts`(5) file.

Edit `/export/client/root/etc/hosts`

```
::1                     localhost
127.0.0.1               localhost
192.168.1.10 client.test.net client
192.168.1.5  server.test.net server
```

7. Setting up the server daemons

If you want these services to start up every time you boot your server, make sure the following lines are present in your `/etc/rc.conf`:

```
dhcpd=YES         dhcpd_flags="-q"
nfs_server=YES          # enable server daemons
mountd=YES
rpcbind=YES       rpcbind_flags="-l"   # -l logs libwrap
```

Also, you'll need to make sure the `tftpd` line in `/etc/inetd.conf` remains uncommented.

Skip forward to **Installing the NetBSD System**

- *Preparing a SCSI or IDE drive with the CD-R image*

If the NetBSD/macppc CD-R image is compressed, then you will need to uncompress it first.

Find a spare bootable drive (i.e. SCSI or IDE), and use some tool to write the disk image `macppccd-5.0.1.iso` to your spare drive, and boot from that drive. For example, you could use a Zip drive, a Jaz drive, a Compact Flash drive, or even a spare hard drive. The disk image has an HFS partition with `ofwboot.xcf` which loads the installation kernel from the ISO file system on the disk

image.

- *From an UNIX-like machine (including MacOS X)*

  ```
  # dd if=macppccd-5.0.1.iso of=/dev/rsd0c
  ```

  where `/dev/rsd0c` is the 'whole disk' partition for the drive you will be using. Be certain you have typed this correctly, as it will erase the disk.

- *From a Windows machine*

  Get **`rawrite32.exe`** from **http://www.duskware.com/rawrite32/**

  Be certain you have selected the correct disk, as it will erase the contents.

- *From a MacOS 9 (or earlier) machine*

  Get **`suntar`** from
  http://hyperarchive.lcs.mit.edu/HyperArchive/Archive/cmp/suntar-223.hqx

  1. Open the 'Preferences' menu and pick 'Expert Mode'
  2. Open the 'File' menu and pick 'Open Device ...'
  3. Click on the "scan SCSI" button to get a list of which drives are attached.
  4. Pick the correct drive. Be certain you have selected the correct disk, as it will erase the contents.
  5. Open the 'Special' menu and pick 'Overwrite Sectors ...'
  6. Hit return (i.e. start at sector number zero)
  7. Select `macppccd-5.0.1.iso` and click "Open"
  8. Open the 'File' menu and pick 'Close Device'

Skip forward to **Installing the NetBSD System**

Preparing your Open Firmware 1.x or 2.x System for NetBSD

Getting to Open Firmware on Apple Network Servers

(Open Firmware 1.1.22)

The version of Open Firmware in the Apple Network Servers can only use a serial console. You must first hook up a serial console (38400 bps, 8 bits, no parity, 1 stop bit, no handshaking) to 'Port 2' (the '`ttya`' device in Open Firmware).

Hold down a special four-key combination on the keyboard attached to the ADB port on your system (not the serial console) when your system boots.

After the chime starts, but before it stops, hold down the four `COMMAND-OPTION-O-F` keys (the `COMMAND` key looks like a four-leaf clover or an open apple, and the `OPTION` key may look like a two-way switch with four straight line segments or say `ALT`) until you see some introductory text and the Open Firmware command prompt on your serial terminal:

```
0 >
```

Your Apple Network Server's screen will remain black.

Now, set your system to always stop at the Open Firmware prompt.

```
0 > setenv auto-boot? false
```

Skip down to the section on **Setting up Open Firmware 1 and 2 to boot NetBSD** since the next several pages are instructions for MacOS models.

Open Firmware 1 and 2 System Preparation

Open Firmware has two variables, 'input-device' and 'output-device', which specify how it accepts commands and displays output. All Open Firmware 1.0.5 and most Open Firmware 2.0.x systems will default to using the 'Modem' serial port for the console instead of the ADB keyboard and the monitor attached to the on-board video.

Unless you use a MacOS-based utility to set these variables correctly, you *will* need to hook up a serial console temporarily to configure Open Firmware to use your keyboard and screen. Some models (such as the Performa 54xx, 6360, 6400, and 6500) have the 'Modem' serial port covered with a piece of plastic since the internal modem usurps that serial port. You will either need to use Boot Variables to set the 'input-device' and 'output-device' variables to 'ttyb' (which is the Printer serial port) or remove the internal modem.

Open Firmware seems to ignore the settings on most DB15 to VGA adapters. Depending on your model, it will default to either 640 x 480 at 60 Hz or to the resolution previously selected in MacOS. Make sure that your monitor can handle these resolutions.

Now would be a good time to look at the NetBSD/macppc Model Support webpage to determine the issues with your model.

http://www.NetBSD.org/ports/macppc/models.html

In particular, some models *must* use a serial console, or they will be unable to boot NetBSD at all. All models can be set to use a serial console, if you desire to bypass the keyboard and screen.

If, after re-reading the next several sections, you still need help figuring out your 'input-device' and 'output-device' settings, see the FAQ:

http://www.NetBSD.org/ports/macppc/faq.html#ofw-input-output-devices

If you need to use a serial console, you can use a normal 'printer' cable (mini-DIN 8 to mini-DIN 8) and a MacOS tool, such as ZTerm to connect a MacOS system to your NetBSD/macppc system.

http://hyperarchive.lcs.mit.edu/HyperArchive/Archive/comm/term/zterm-101.hqx
http://homepage.mac.com/dalverson/zterm/

See the NetBSD Serial Port Primer for additional help and references:

http://www.NetBSD.org/docs/Hardware/Misc/serial.html

All Open Firmware 1 and 2 macppc systems have Open Firmware bugs. Luckily, Open Firmware has a small Non-Volatile RAM variable (NVRAM) which is reserved for FORTH commands which will be run before booting an operating system. Apple has released a freeware MacOS 9 tool called System Disk, which patches most of these bugs. We *strongly* recommend that you use this tool to patch your Open Firmware, as several systems cannot boot without these patches. Instructions for using System Disk are covered in the next section.

Unfortunately, some models are broken by or are unsupported by System Disk. If you have one of the following models, then skip down to the section on **Getting to Open Firmware (Harder, MacOS 7 thru 9)**
Apple Performa 4400, 5500, 6500, 54xx, 6400, and 6360,
Motorola Starmax 3000, 4000, 5000, and 5500,
APS Tech M*Power 604e/200,
PowerComputing PowerBase,
Umax Apus 2000, Apus 3000, C500, and C600
Umax S900

Getting to Open Firmware (MacOS X or Darwin)

When you install MacOS X or Darwin, it will install the necessary NVRAM bug fixes. Use the **nvram** command to set your system to always stop at the Open Firmware prompt, then reboot.

```
# nvram auto-boot\?=false
```

You will need to escape the question-mark or enclose the whole **nvram** argument in double-quotes to prevent your shell from trying to interpret it.

You should now see the Open Firmware command prompt on your screen:

```
Open Firmware, 1.0.5
To continue booting the MacOS type:
BYE<return>
To continue booting from the default boot device type:
BOOT<return>
 ok
0 >
```

If your screen is black, then your model has defaulted to using a serial console. You must hook up a serial console (38400 bps, 8 bits, no parity, 1 stop bit, no handshaking).

Getting to Open Firmware (Best, MacOS 8 or 9)

Download System Disk:
ftp://ftp.apple.com/developer/macosxserver/utilities/SystemDisk2.3.1.smi.bin

For a brief tutorial on how to use System Disk, see:
http://www.NetBSD.org/ports/macppc/SystemDisk-tutorial/

Launch the MacOS System Disk tool. Click on "Power User (Open Firmware)" then click on the "Advanced Options" button. Now, click on the checkbox that says "Stop Boot at Open Firmware prompt" and select "OK". Click the "Save" button and reboot your system.

Note: NVRAM patches and Open Firmware settings will be erased if you 'zap your PRAM' by holding down COMMAND-OPTION-P-R keys during the boot chimes, or if you accidentally boot into MacOS 9 or earlier.

If your 'output-device' is '/chaos/control' (i.e. you have a PowerMacintosh 7300 — 8600 system), there is a chance that your monitor will not sync. See
http://www.NetBSD.org/ports/macppc/SystemDisk-tutorial/of105patch.html

You should now see the Open Firmware command prompt on your screen:

```
Open Firmware, 1.0.5
To continue booting the MacOS type:
BYE<return>
To continue booting from the default boot device type:
BOOT<return>
 ok
0 >
```

If your screen is black, then your model does not support using the on-board video in Open Firmware. You will need to connect a serial console to the 'Modem' port of your system (38400 bps, 8 bits, no parity, 1 stop bit, no handshaking).

Note: The "Stop Boot at Open Firmware prompt" setting is persistent. It is equivalent to the Open Firmware command

```
0 > setenv auto-boot? false
```

Note: Unfortunately, there are a few models that are better off without the System Disk patches. If you find that your machine doesn't boot, then try:

```
0 > setenv use-nvramrc? false
0 > reset-all
```

Getting to Open Firmware (Harder, MacOS 7, 8, or 9)

If System Disk doesn't work because your version of MacOS is too old or because System Disk says that it doesn't support your model, then you may try using the MacOS BootVars tool.

ftp://ftp.NetBSD.org/pub/NetBSD/arch/macppc/macos-utils/bootvars/bootvars.sit.hqx

Note: BootVars does *not* apply the (possibly critical) NVRAM patches that System Disk does. Expect some devices to not work (such as booting from hard drives and CD-ROMs).

Look up the proper 'output-device' for your model on the NetBSD/macppc Model Support webpage.

http://www.NetBSD.org/ports/macppc/models.html

If the NetBSD/macppc Model support webpage does not list an 'output-device' for your model, then your system will default to using the on-board video. You needn't fill in the 'output-device' and 'input-device' variables.

Launch the MacOS BootVars tool. Uncheck the "auto-boot?" checkbox, then check on the "All Variables" checkbox and type **kbd** into the 'input-device' box, and the proper device name into the 'output-device' box. Click on the "write" button, and then reboot your system.

If your 'output-device' is '`/chaos/control`' (i.e. you have a PowerMacintosh 7300 — 8600 system), there is a chance that your monitor will not sync. See

http://www.NetBSD.org/ports/macppc/SystemDisk-tutorial/of105patch.html

If you have a Performa 5500 or 6500, you may need to apply NVRAMRC patches to use your built-in video. See the information in the NetBSD/macppc Model Support webpage.

You should now see the Open Firmware command prompt on your screen:

```
Open Firmware, 1.0.5
To continue booting the MacOS type:
BYE<return>
To continue booting from the default boot device type:
BOOT<return>
 ok
0 >
```

If your screen is black, then your model has defaulted to using a serial console. This is fairly common on Open Firmware 1 and 2 models if you do not use the System Disk tool to set up Open Firmware. You must hook up a serial console (38400 bps, 8 bits, no parity, 1 stop bit, no handshaking).

Note: The "auto-boot?" setting is persistent. Your system will always stop at the Open Firmware prompt. It is equivalent to the Open Firmware command

```
0 > setenv auto-boot? false
```

Getting to Open Firmware (Without using MacOS)

(Open Firmware 1.0.5, Open Firmware 2.0.x, Open Firmware 2.4)

If you don't have MacOS, then you need to hold down a special four-key combination when your system boots. Do this on the keyboard attached to the ADB port on your system (not the serial console or PS/2 port) when your system boots.

Note: Your system will *not* have the (possibly critical) NVRAM patches that System Disk applies. Expect some devices to not work (such as booting from hard drives and CD-ROMs).

After the chime starts, but before it stops, hold down the four `COMMAND-OPTION-O-F` keys (the `COMMAND` key looks like a four-leaf clover or an open apple, and the `OPTION` key may look like a two-way switch with four straight line segments or say `ALT`) until you see the Open Firmware command prompt on your screen or serial console:

```
Open Firmware, 1.0.5
To continue booting the MacOS type:
BYE<return>
To continue booting from the default boot device type:
BOOT<return>
 ok
0 >
```

If your screen is black, then your system has defaulted to using a serial console. This is fairly common on Open Firmware 1 and 2 models if you do not use the System Disk tool to set up Open Firmware. You must hook up a serial console (38400 bps, 8 bits, no parity, 1 stop bit, no handshaking).

Now, set your system to always stop at the Open Firmware prompt.

```
0 > setenv auto-boot? false
```

To use your on-board video and keyboard, look up the proper 'output-device' for your model on the NetBSD/macppc webpage

http://www.NetBSD.org/ports/macppc/models.html

Run the following commands to use your screen instead of a serial console (replace '`screen`' with the correct 'output-device' for your model):

```
0 > setenv output-device screen
0 > setenv input-device kbd
0 > reset-all
```

Now you should see the Open Firmware prompt on your screen.

If your 'output-device' is '`/chaos/control`' (i.e. you have a PowerMacintosh 7300 — 8600 system), there is a chance that your monitor will not sync. See

http://www.NetBSD.org/ports/macppc/SystemDisk-tutorial/of105patch.html

Setting up Open Firmware 1 and 2 to boot NetBSD

This section describes some steps you must take to prepare Open Firmware to boot NetBSD. Additional resources are available in the FAQ regarding how to use the Open Firmware command environment, and the Open Firmware variables you may be using:

http://www.NetBSD.org/ports/macppc/faq.html#ofw-access
http://www.NetBSD.org/ports/macppc/faq.html#ofw-variables

Double-check your Open Firmware version:

```
0 > dev /openprom
0 > .properties
name                    openprom
model                   Open Firmware, 1.0.5
relative-addressing
 ok
```

If your system has Open Firmware prior to version 3, then you must set some Open Firmware variables before NetBSD can boot. Do *not* run these commands on Open Firmware 3 machines, as you may overwrite your firmware requiring a trip to Apple for repairs.

```
0 > setenv load-base 600000
0 > setenv real-base F00000
0 > reset-all
```

The last command reboots your machine so that the settings are stored.

If you will be netbooting your system, you can look up your MAC address.

```
0 > dev enet
0 > .properties
[...]
local-mac-address   CCCCCCCC CCCC
[...]
 ok
```

Note: Some early Open Firmware 1.0.5 machines had their MAC address stored incorrectly on the motherboard (little- vs. big-endian problems). The patches the System Disk installs will correct this. Without the patch, the machine will still work, but its MAC address may conflict with another ethernet device on your network.

You can check your Open Firmware settings with the `printenv` command:

```
0 > printenv
little-endian?      false               false
real-mode?          false               false
auto-boot?          false               true
diag-switch?        false               false
[...]
use-nvramrc?        true                false
real-base           F00000              -1
[...]
load-base           600000              4000
[...]
input-device        kbd                 ttya
output-device       /chaos/control      ttya
```

Note: All Open Firmware 1.0.5 settings and nvram patches will be erased if you boot into MacOS 9 or earlier. You will need to re-enter them before booting NetBSD again.

Note: Open Firmware 2.0.x and Open Firmware 2.4 systems will set the `real-base` environment variable to its default value (which prevents NetBSD from booting) if you boot into MacOS 9 or earlier.

Available Boot Media

Open Firmware is capable of booting from a variety of media (such as hard drives, floppy disks, CD-ROMs, and ethernet). Open Firmware is able to boot files from a variety of file systems (such as ISO9660, and MS-DOS FAT). Unfortunately, Open Firmware is *not* able to directly boot from the NetBSD file system (FFS) or Apple's file systems (HFS, HFS+, or UFS), so we must put the bootloader in a location that Open Firmware is capable of understanding.

Therefore, to boot the NetBSD kernel, Open Firmware must first load a 'bootloader' which knows how to load the NetBSD kernel. Open Firmware 1 and 2 take either a two or three stage approach, depending on the boot media. In the two step approach, Open Firmware loads `ofwboot.xcf` from the boot media, which then loads the kernel. In the three step approach (used in the 'partition zero' method), Open Firmware loads a primary bootloader `bootxx` from a disk which then loads the secondary bootloader `ofwboot` (which is functionally identical to `ofwboot.xcf`) which then loads the kernel.

Note: `ofwboot.elf` is obsoleted. All users should be using `ofwboot.xcf` instead of `ofwboot.elf` now.

The following bootable media are available for loading the bootloader:

- '*Partition zero*'

  This method loads the primary bootloader `bootxx` located in the Apple Partition Map, which then loads the secondary bootloader `ofwboot` which then loads the kernel. This is what **`sysinst`** will place on your hard drive during the installation procedure. Additionally, the NetBSD/macppc CD-R images and boot floppies use this method, loading a kernel from the floppy's FFS partition or the CD-R's ISO9660 partition.

- *MS-DOS file system*

  In this method, Open Firmware loads the `ofwboot.xcf` bootloader from an MS-DOS file system. It may then load a NetBSD kernel from the same MS-DOS file system. This has only been thoroughly tested on floppy disks, but may work on Zip disks or FDISK-formatted hard drives. This does not work for MS-DOS partitions on a hard drive with an Apple Partition Map.

- *Ethernet (network boot)*

  You can run your entire system diskless or netboot only the files necessary to boot (i.e. the bootloader and the installation kernel). You must have root access for the UNIX-like netboot server, which must be on the same subnet as your NetBSD/macppc machine.

  Note: You must use your on-board ethernet device for netbooting. While there may be some PCI or Cardbus ethernet interfaces with Open Firmware ROMs, no one has reported success netbooting using these devices

- *ISO9660 CD-ROM file system*

  Load the `ofwboot.xcf` bootloader from an ISO9660 CD-ROM. It can then load a NetBSD kernel from the ISO9660 CD-ROM.

  Open Firmware 1.0.5 and 2.0.x machines will *not* boot from the ISO9660 file system if you create a hybrid HFS/ISO format CD-R. Make sure that the CD *only* has ISO9660 data and does not have an Apple Partition Map.

  Note: Open Firmware can only open files on the first session of a multi-session CD-R

Once the bootloader is loaded, it can open the kernel from one of the following sources:

- *Ethernet (NFS)*
- *NetBSD FFS file system*

  Such as the boot floppy, or an installed NetBSD partition.
- *MacOS X UFS file system*
- *ISO9660 CD-ROM file system*
- *MS-DOS file system*

In theory, Open Firmware 2.4 systems should be able to load the bootloader from an HFS or HFS+ partition or a hybrid CD-R, but users have reported that it doesn't work.

The boot floppy images provided have a 'partition zero' bootloader and a NetBSD file system with an installation kernel. The boot CD image provided has both a 'partition zero' bootloader and `ofwboot.xcf` on a hybrid partition so it can be booted on all Open Firmware versions. It also has an ISO9660 file system with an installation kernel and the distribution sets.

Partitioning your hard drive for NetBSD

You must use the NetBSD installer to partition your disk if you want it to be bootable. With this release of NetBSD, there is no way to dual-boot MacOS and NetBSD on one hard drive.

You can use the instructions in this section to partition a disk that may also be used with MacOS, although a disk prepared in this way *will not boot* NetBSD. That means, your root partition (`/`) must be on a drive prepared with the NetBSD installer, but the partitions not necessary to boot (for example `/usr`, `/home`, or `/export`) may be on the same disk as MacOS.

Unless you are planning to use partitions on the same disk as MacOS, skip forward to **Preparing the Open Firmware 1 or 2 Bootable Media**

There are two partitioning tools available for NetBSD/macppc, `disklabel`(8) and `pdisk`(8). The former is used in the NetBSD **`sysinst`** installer, and will render your disk unusable by MacOS.

The process is more fully detailed in the Partitioning HOW-TO:
http://www.NetBSD.org/ports/macppc/partitioning.html

You can create a partition map with `pdisk`(8), but the disk will not be usable with MacOS 9 and earlier. If this is a concern, you will need to use Apple's Drive Setup or Disk Utility.

If you are using Apple's Drive Setup tool, make sure you have version 1.8.1 or later. This tool only runs under MacOS 9 and earlier. Drive Setup will erase the contents of your drive — it does not preserve data from any of your partitions.

Apple's Disk Utility only runs under MacOS X 10.0.0 and later. Make sure you click the "Install Mac OS 9 Disk Drivers" checkbox. Also, keep in mind that Disk Utility does *not* create the partitions that NetBSD/macppc requires. After creating the initial partition map with Disk Utility, you will need to use the NetBSD **`pdisk`** to change the partition types. Also, Disk Utility will erase the contents of your drive — it does not preserve data from any of your partitions.

`pdisk` is the most flexible (and most difficult to use) partitioning tool available. It runs on almost all OSes that macppc machines support. Download it:
ftp://ftp.NetBSD.org/pub/NetBSD/arch/macppc/netbsd-pdisk/
ftp://ftp.NetBSD.org/pub/NetBSD/arch/macppc/macos-utils/pdisk.sea.hqx

There is built-in help describing how it works. When it asks you to enter the "`Type of partition`", use `Apple_UNIX_SVR2` for NetBSD partitions, `Apple_HFS` for HFS and HFS+ partitions, and `Apple_UFS` for UFS partitions.

After you've written the partition map with **pdisk**, you will need to create the file systems. Use newfs(8) and mount(8) for NetBSD file systems, and **hfstools** to create and mount HFS file systems.
ftp://ftp.NetBSD.org/pub/NetBSD/packages/pkgsrc/sysutils/hfsutils/README.html

Make the following partitions:

A/UX Swap	Any size. The recommendation is your RAM size, although this is not strictly necessary for machines with a lot of RAM. NetBSD interprets an *A/UX Swap* partition as the second partition (*b*) on the disk. This partition is not readable from MacOS.
A/UX User and *A/UX Free1*	Use these for any additional partitions you may want to use under NetBSD, such as /usr (at least 200 MB), /home, /usr/local, or /usr/pkg. NetBSD interprets these partitions as normal NetBSD-style partitions. These partitions are not readable from MacOS.
HFS	Any size. You may want to leave an additional partition available to transfer files between MacOS and NetBSD. If would like to create such a partition, then see the Partitioning HOW-TO. **http://www.NetBSD.org/ports/macppc/partitioning.html#msdos**
UFS	Any size. *UFS* partitions are not readable from MacOS versions prior to X 10.0.0. If you use an *UFS* partition as your root, then it may not be recognized by the NetBSD kernel as the first partition (*a*) on the disk. You will need to compile a new kernel with the root partition explicitly defined to be the *UFS* partition.

Now would be a good time to use **pdisk** to determine the partition numbers for your bootloader and kernel.

Preparing the Open Firmware 1 or 2 Bootable Media

The purpose of this section is to prepare the media from which your system will boot the installer. We'll describe how to put the files in the right places on your disk(s) or netboot server and prepare it for use on your system.

If you will be running your system diskless (i.e. entirely over NFS, not using any local hard drives), then you do not need to run the installer, you only need to extract the distribution sets on the diskless server.

To get the distribution sets onto appropriate media, see the above section entitled **Getting the NetBSD System on to Useful Media**. You may want to get the distribution sets when you create the bootable media.

Note: Some MacOS **ftp** clients default to downloading files in 'ASCII' mode. This will render the NetBSD files useless. Make sure to set your ftp program to download in 'binary' mode.

What follows are the steps to create different types of bootable media for the NetBSD install kernel. You should only need to create one of these to get your system to boot the installer

- *Creating the NetBSD/macppc CD-R*

  Go to one of the NetBSD mirror sites and download the CD-R image macppccd-5.0.1.iso from the pub/NetBSD/iso directory. **http://www.NetBSD.org/mirrors/#iso**

  – *From an UNIX-like machine*

  Get and install cdrecord. NetBSD users should install it from the package collection. Other UNIX-like systems should get it from the official website:
  ftp://ftp.NetBSD.org/pub/NetBSD/packages/pkgsrc/sysutils/cdrtools/README.html
  http://www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/cdrecord.html

```
# cdrecord -v speed=4 dev=/dev/cd1c macppccd-5.0.1.iso
```

You will need to substitute the correct name of the disk image file, speed for your CD writer, and the correct device for your system (for i386 it would be `/dev/cd1d`).

– *From a MacOS machine (using Toast)*

1. Click the 'Other' button in the main window.
2. Open the contextual menu on the 'Other' button and select 'Disk Image'
3. Click the 'Select' button and select the disk image you downloaded.
4. Click the 'Record' button

Skip forward to **Installing the NetBSD System**

- *Creating the NetBSD/macppc install floppies*

1. First, make sure you have done a low-level format on the disks. A bad floppy can produce Open Firmware errors similar to other problems.
2. If the boot floppy images are compressed, then you will need to uncompress them first.
3. Use a tool to write the boot floppy images `installation/floppy/boot1.fs` and `installation/floppy/boot2.fs` to two floppy disks.

– *From an UNIX-like machine*

```
# dd if=boot1.fs of=/dev/rfd0a bs=36k
# dd if=boot2.fs of=/dev/rfd0a bs=36k
```

– *From a Windows machine*

Get either the **rawrite.exe** program from the NetBSD/i386 distribution or **rawrite32.exe** from **http://www.duskware.com/rawrite32/**

– *From a MacOS 9 (or earlier) machine*

Get **suntar** from
http://hyperarchive.lcs.mit.edu/HyperArchive/Archive/cmp/suntar-223.hqx

1. Open the 'Preferences' menu and pick 'Expert Mode'
2. Open the 'Special' menu and pick 'Overwrite Sectors ...'
3. Insert the first floppy disk
4. Hit return (i.e. start at sector number zero)
5. Select `boot1.fs` and click "Open"
6. Open the 'File' menu and pick 'Eject'
7. Repeat the process for the second floppy

Skip forward to **Installing the NetBSD System**

- *Creating a custom ISO9660 CD-R*

This section describes how to create your own bootable NetBSD/macppc CD-R. We recommend that you use the official NetBSD/macppc–5.0.1 CD-R image, as described in the section above.

Place `ofwboot.xcf` and the installation kernel `netbsd-GENERIC_MD.gz` at the top level of the CD. Due to restrictions in the way that Open Firmware deals with ISO filenames, you may wish to name your kernel `netbsd.gz`. You may also place the NetBSD 5.0.1 distribution sets on the disk.

– *From an UNIX-like machine*

Get and install mkisofs. This is now part of the cdrecord package. NetBSD users should install it from the package collection. Other UNIX-like systems should get it from the official website:
ftp://ftp.NetBSD.org/pub/NetBSD/packages/pkgsrc/sysutils/cdrtools/README.html
http://www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/cdrecord.html

```
# mkisofs -o output.iso -l -J -r -L -N /cdsources
# cdrecord -v speed=4 dev=/dev/cd1c output.iso
```

You will need to substitute the correct speed for your CD writer, and the correct device for your system (for i386 it would be `/dev/cd1d`).

See the NetBSD Bootable CD-ROM HOW-TO for more detail:
http://www.NetBSD.org/docs/bootcd.html

– *From a MacOS machine (using Toast)*

1. Click the 'Other' button in the main window
2. Open the contextual menu on the 'Other' button and select 'ISO 9660'
3. Click the 'Select' button.
4. Click the 'Settings' tab, open the 'Naming' popup menu, and pick 'Allow Macintosh Names'
5. Click on the 'Files' tab
6. Drag `installation/ofwboot.xcf` and `binary/kernel/netbsd-GENERIC_MD.gz` into the window. You may also want to drag the NetBSD 5.0.1 distribution sets to this window as well.
7. Click the 'Done' button
8. Click the 'Record' button

Skip forward to **Installing the NetBSD System**

• *Creating an MS-DOS disk*

Unfortunately, the installation kernel is now too large to fit onto a floppy disk, but you can still create a bootable Zip disk with this method.

1. First, make sure you have done a low-level format on the disk. A bad disk can produce Open Firmware errors similar to other problems.
2. Mount the disk on your computer.
   – Insert it into a Windows or DOS machine.
   – Use 'File Exchange or 'PC Exchange'' with MacOS versions prior to X 10.0.0
   – Use `mount_msdos`(8) on any UNIX-like machine.
3. Copy `netbsd-INSTALL.gz` and `ofwboot.xcf` to the disk.
4. Rename `netbsd-INSTALL.gz` to `netbsd.gz`, since OpenFirmware can't deal with MS-DOS filenames longer than eight characters.

Skip forward to **Installing the NetBSD System**

• *Preparing the netboot server*

1. Introduction

To netboot a macppc, you must configure one or more servers to provide information and files to your macppc (the 'client'). If you are using NetBSD (any architecture) on your netboot server(s), the information provided here should be sufficient to configure everything. Additionally, you may wish to look at the diskless(8) manual page and the manual pages for each daemon you'll be configuring. If the server(s) are another operating system, you should consult the NetBSD Diskless HOWTO, which will walk you through the steps necessary to configure the netboot services on a variety of platforms.

http://www.NetBSD.org/docs/network/netboot/

You may either netboot the installer so you can install onto a locally attached disk, or you may run your system entirely over the network.

Briefly, the netboot process involves discovery, bootstrap, kernel and file system stages. In the first stage, the client discovers information about where to find the bootstrap program. Next, it downloads and executes the bootstrap program. The bootstrap program goes through another discovery phase to determine where the kernel is located. The bootstrap program tries to mount the NFS share containing the kernel. Once the kernel is loaded, it starts executing. For RAM disk kernels, it mounts the RAM disk file system and begins executing the installer from the RAM disk. For normal (non-RAM disk) kernels, the kernel tries to mount the NFS share that had the kernel and starts executing the installation tools or init(8). All macppc systems use BOOTP for the discovery stage. TFTP is used in the bootstrap phase to download the bootstrap program, ofwboot.xcf. NFS is used in both the kernel and file system stages to download the kernel, and to access files on the file server.

We will use 'CC:CC:CC:CC:CC:CC' as the MAC address (ethernet hardware address) of your netboot client machine. You should have determined this address in an earlier stage. In this example, we will use '192.168.1.10' as the IP address of your client and 'client.test.net' as its name. We will assume you're providing all of your netboot services on one machine called 'server.test.net' with the client's files exported from the directory /export/client/root. You should, of course, replace all of these with the names, addresses, and paths appropriate to your environment.

You should set up each netboot stage in order (i.e., discovery, bootstrap, kernel, and then file system) so that you can test them as you proceed.

2. dhcpd(8) in bootpd(8) compatible mode

   Put the following lines in your /etc/dhcpd.conf (see dhcpd.conf(5) and dhcp-options(5) for more information):

```
ddns-update-style none;
                # Do not use any dynamic DNS features
                #
allow bootp;    # Allow bootp requests, thus the dhcp server
                # will act as a bootp server.
                #
authoritative;  # master DHCP server for this subnet
                #
subnet 192.168.1.0 netmask 255.255.255.0 {
                # Which network interface to listen on.
                # The zeros indicate the range of addresses
                # that are allowed to connect.
}
group {
                # Set of parameters common to all clients
```

```
                # in this "group".
                #
        option broadcast-address        192.168.1.255;
        option domain-name              "test.net";
        option domain-name-servers      dns.test.net;
        option routers                  router.test.net;
        option subnet-mask              255.255.255.0;
                #
                # An individual client.
                #
        host client.test.net {
                hardware ethernet       CC:CC:CC:CC:CC:CC;
                fixed-address           192.168.1.10;
                #
                # Name of the host (if the fixed address
                # doesn't resolve to a simple name).
                #
                option host-name        "client";
                #
                # Name of the bootloader or kernel
                # to download via tftp.
#
                filename                "ofwboot.xcf";

                #
                # The path on the NFS server.
                #
                option root-path "/export/client/root";

                #
                # If your DHCP server is not your NFS server, supply the
                # address of the NFS server. Since we assume you run every
                # on one server, this is not needed.
#
                # macppc machines will look for their bootloader,
                # ofwboot.xcf, on the next-server as well.
                #
                # next-server server.test.net;
        }
 #you may paste another "host" entry here for additional
 #clients on this network
}
```

You will need to make sure that the `dhcpd.leases` file exists.

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

You will need to start the **dhcpd**. If it's already running, you will need to restart it to force it to re-read its configuration file. If the server is running NetBSD, you can achieve this with:

```
# /etc/rc.d/dhcpd restart
```

3. tftpd(8)

   The default configuration of the TFTP server is to run in a chroot(8) environment in the /tftpboot directory. Thus, the first order of business is to create this directory:

   ```
   # mkdir -p /tftpboot
   ```

   Next, edit /etc/inetd.conf and uncomment the line with the TFTP daemon:

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

   Now, restart inetd(8). If the server is running NetBSD, you can achieve this with:

   ```
   # /etc/rc.d/inetd restart
   ```

   Now, you need to copy the bootloader for your macppc machine to /tftpboot. Get ofwboot.xcf(8) from the installation directory of the distribution.

   ```
   # cp ofwboot.xcf /tftpboot
   ```

   Just to be sure, let's make everything readable.

   ```
   # chmod -R a+rX /tftpboot
   ```

   Sometimes, the arp(8) table gets messed up, and the TFTP server can't communicate with the client. In this case, it will write a log message (via syslogd(8)) to /var/log/messages saying: 'tftpd: write: Host is down'. If this is the case, you may need to force the server to map your client's ethernet address to its IP address:

   ```
   # arp -s client CC:CC:CC:CC:CC:CC
   ```

4. nfsd(8), mountd(8), and rpcbind(8)

   Now your system should be able to load the bootstrap program and start looking for the kernel. Let's set up the NFS server. Create the directory you are exporting for the netboot client:

   ```
   # mkdir -p /export/client/root
   ```

   Put the following line in /etc/exports to enable NFS sharing:

   ```
   /export/client/root -maproot=root client.test.net
   ```

   If your server is currently running an NFS server, you only need to restart mountd(8). Otherwise, you need to start rpcbind(8) and nfsd(8). If the server is running NetBSD, you can achieve this with:

   ```
   # /etc/rc.d/rpcbind start
   # /etc/rc.d/nfsd start
   # /etc/rc.d/mountd restart
   ```

5. NetBSD kernel and installation tools

   Now, if you place a kernel named netbsd in /export/client/root your client should boot the kernel. If you are netbooting the installer, use binary/kernel/netbsd-GENERIC_MD.gz (this has the installation tools in a RAM disk). Also, copy the distribution files to the client's root directory.

   Note: Some combinations of Open Firmware version and ofwboot.xcf version have trouble loading compressed kernels. If you have trouble loading a kernel, try uncompressing it with gunzip(1)

   ```
   # cp *tgz /export/client/root
   # gunzip netbsd-GENERIC_MD.gz
   # mv netbsd-GENERIC_MD /export/client/root/netbsd
   ```

If you are running your macppc diskless, simply use `binary/kernel/netbsd-GENERIC.gz`.

6. Client file system

   You can skip this step if you do not plan to run your client diskless after installation. Otherwise, you need to extract and set up the client's installation of NetBSD. The Diskless HOW-TO describes how to provide better security and save space on the NFS server over the procedure listed here. See
   http://www.NetBSD.org/docs/network/netboot/nfs.html
   for details.

   – Extracting distribution sets

   ```
   # cd /export/client/root
   # tar -xpzf /path/to/files/base.tgz
   # tar -xpzf /path/to/files/etc.tgz
   ```

   Continue with the other non-essential distribution sets if desired.

   – Set up swap

   ```
   # mkdir /export/client/root/swap
   # dd if=/dev/zero of=/export/client/swap bs=4k count=4k
   # echo '/export/client/swap -maproot=root:wheel client.test.net' | cat >>
   # /etc/rc.d/mountd restart
   ```

   This creates a 16 MB swap file and exports it to the client.

   – Create device nodes

   ```
   # cd /export/client/root/dev
   # ./MAKEDEV all
   ```

   This procedure only works on NetBSD hosts.

   – Set up the client's `fstab`(5)

   Create a file in `/export/client/root/etc/fstab` with the following lines:

   ```
   server:/export/client/swap   none  swap  sw,nfsmntpt=/swap
   server:/export/client/root   /     nfs   rw 0 0
   ```

   – Set up the client's `rc.conf`(5)

   Edit `/export/client/root/etc/rc.conf`

   ```
   rc_configured=YES
   hostname="client"
   defaultroute="192.168.1.1"
   nfs_client=YES
   auto_ifconfig=NO
   net_interfaces=""
   ```

   Make sure rc does not reconfigure the network device since it will lose its connection to the NFS server with your root file system.

   – Set up the client's `hosts`(5) file.

   Edit `/export/client/root/etc/hosts`

   ```
   ::1                     localhost
   127.0.0.1               localhost
   192.168.1.10 client.test.net client
   192.168.1.5  server.test.net server
   ```

7. Setting up the server daemons

   If you want these services to start up every time you boot your server, make sure the following lines are present in your `/etc/rc.conf`:

```
dhcpd=YES         dhcpd_flags="-q"
nfs_server=YES          # enable server daemons
mountd=YES
rpcbind=YES       rpcbind_flags="-l"   # -l logs libwrap
```

   Also, you'll need to make sure the `tftpd` line in `/etc/inetd.conf` remains uncommented.

Skip forward to **Installing the NetBSD System**

- *Preparing a SCSI or IDE drive with the CD-R image*

  If the NetBSD/macppc CD-R image is compressed, then you will need to uncompress it first.

  Find a spare bootable drive (i.e. SCSI or IDE), and use some tool to write the disk image `macppccd-5.0.1.iso` to your spare drive, and boot from that drive. For example, you could use a Zip drive, a Jaz drive, a Compact Flash drive, or even a spare hard drive. The disk image has a 'partition zero' bootloader which ultimately loads the installation kernel from the ISO file system on the disk image.

  – *From an UNIX-like machine (including MacOS X)*

    # **`dd if=macppccd-5.0.1.iso of=/dev/rsd0c`**

    where `/dev/rsd0c` is the 'whole disk' partition for the drive you will be using. Be certain you have typed this correctly, as it will erase the disk.

  – *From a Windows machine*

    Get **`rawrite32.exe`** from **http://www.duskware.com/rawrite32/**

    Be certain you have selected the correct disk, as it will erase the contents.

  – *From a MacOS 9 (or earlier) machine*

    Get **`suntar`** from
    http://hyperarchive.lcs.mit.edu/HyperArchive/Archive/cmp/suntar-223.hqx

    1. Open the 'Preferences' menu and pick 'Expert Mode'
    2. Open the 'File' menu and pick 'Open Device ...'
    3. Click on the "scan SCSI" button to get a list of which drives are attached.
    4. Pick the correct drive. Be certain you have selected the correct disk, as it will erase the contents.
    5. Open the 'Special' menu and pick 'Overwrite Sectors ...'
    6. Hit return (i.e. start at sector number zero)
    7. Select `macppccd-5.0.1.iso` and click "Open"
    8. Open the 'File' menu and pick 'Close Device'

Installing the NetBSD System

Open Firmware boot syntax

The syntax of the Open Firmware **boot** command is:

```
boot boot-device [boot-file] [-as]
```

where the `boot-device` describes where to find the bootloader, `boot-file` describes where to find the NetBSD kernel, and the options specify how you want to boot.

You use the `boot-device` to tell Open Firmware where to find `ofwboot`(8) by listing the device, the partition (if it's a disk), and the filename of the bootloader (if using `ofwboot.xcf`).

If the `boot-file` is on the same device and partition (if it's a disk) as the `boot-device` then you can just specify the kernel filename. Otherwise, you need to specify the full Open Firmware path to the kernel.

The **-a** flag will ask you for the location of the next item to load (i.e. the bootloader will ask where the kernel is (if unspecified), or the kernel will ask where the root file system is). The **-s** flag will boot into 'single-user' mode.

The exact command you will be using depends on which version of Open Firmware your machine has and which device you will be booting from. Sometimes you may have to guess as we don't know all of the combinations of models, device names, and file names. In general the format is: `device:[partition][,\filename]`. Keep in mind for the future that you may be able to have your `boot-device` and `boot-file` on entirely different devices (such as the bootloader netbooted from `enet` and the kernel loaded from a hard drive on the `ultra0` ATA/IDE bus).

We'll try to walk you through the process of figuring out what Open Firmware calls your device, partition, and file names. To start with, Open Firmware keeps a "device tree" with all of the devices it finds in your system. You can get a listing of the nodes in this device tree with the **dev** and **ls** commands. **dev** is similar to the unix **cd** command and is used to change between the nodes in the Open Firmware device tree (similar to a file system). **ls** of course is similar to the unix **ls** command and is used to list the contents of the current device node. To get a listing of all the devices available in your system, use the following commands:

```
0 > dev /
0 > ls
```

Open Firmware has device aliases which are simple names for the full hardware path to a device (similar to **alias** in `csh`(1)). You can find out what device aliases Apple created on your machine with the **devalias** command. For example, here are the devaliases on a PowerMacintosh 7300:

```
0 > devalias
vci0                  /chaos@F0000000
pci1                  /bandit@F2000000
pci2                  /bandit@F4000000
fd                    /bandit/gc/swim3
kbd                   /bandit/gc/via-cuda/adb/keyboard
ttya                  /bandit/gc/escc/ch-a
ttyb                  /bandit/gc/escc/ch-b
enet                  /bandit/gc/mace
scsi                  /bandit/gc/53c94
scsi-int              /bandit/gc/mesh
 ok
```

On most systems, you'll find the devices you're looking for. Typical aliases are:

`hd`	internal hard drive
`cd`	CD-ROM drive
`zip`	internal Zip drive
`enet`	ethernet
`fd`	floppy drive
`scsi`	SCSI bus
`scsi-int`	internal SCSI bus (on systems with multiple SCSI busses)
`ata`	ATA/IDE bus
`ideN`	ATA/IDE bus number N
`ultraN`	Ultra/66 or Ultra/100 IDE bus number N

Note that some of these items are the device itself, and some are a bus. When you only have the devalias to a bus, you need to specify which device on that bus you want to use. You can use the Open Firmware **dev** and **ls** commands. For example, here are the devices on the internal SCSI bus of a PowerMacintosh 7300:

```
0 > dev scsi-int
0 > ls
FF83C850: /sd@0,0
FF83D480: /st@0,0
 ok
```

In this case, Open Firmware seems to be saying there are two devices, both at address zero (one is a SCSI disk 'sd@0,0' and the other is a SCSI tape 'st@0,0).' Unfortunately, older systems will only list the naming convention and not the actual devices currently connected, but that's OK -- we've got more tricks up our sleeve.

If you've got ATA/IDE drives, you have all the device information you need (since Apple only ever ships drives as "master" which is typically something like `ata-disk@0`, `ATA-Disk@0`, `atapi-disk`, `or disk@0 ).`

You can find out the devices on your SCSI bus with the **show-children** command:

```
0 > dev scsi-int
0 > show-children
Target 0
  Unit 0  Disk     IBM     DCAS-32160      S65A
Target 3
  Unit 0  Removable Read Only device    SONY    CD-ROM CDU-8005 1.0j
 ok
```

Open Firmware calls SCSI IDs "`Target`". The "`Unit`" number is the Logical Unit Number (LUN). This is almost always zero. Thus, this PowerMacintosh system has an IBM hard drive (DCAS-32160) at SCSI ID 0, and a Sony CD-ROM drive (CDU-8005) at SCSI ID 3.

Now, we've got enough information to construct the device name for Open Firmware. Just stick everything together to describe to Open Firmware what you want. For example, Open Firmware calls the CD-ROM drive in this system `scsi-int/sd@3`.

To determine if a device is bootable, type:

```
0 > dev scsi-int/sd@3
0 > words
load          write        read          seek        close     open
write-blocks  read-blocks  max-transfer  block-size  dma-sync  dma-map-out
dma-map-in    dma-free     dma-alloc
 ok
```

If the **word** "open" is present in the list, then the device is almost certainly bootable.

Next, you need to figure out what partition Open Firmware thinks your bootloader is located on if you're going to boot from a disk. If you're using a "partition zero" bootloader, the answer is obvious: 0. Thus, your boot-device for the NetBSD/macppc CD-R image on an Open Firmware 1.0.5 system would be scsi-int/sd@3:0 since the image has a 'partition zero' bootloader.

Other situations get a little trickier, as we know of no way to get a partition map from within Open Firmware, and it uses a different numbering scheme than either NetBSD or MacOS 9 (or earlier). You can use **pdisk** to get a listing of the partitions on a disk. See the Partitioning HOW-TO for help:

http://www.NetBSD.org/ports/macppc/partitioning.html#pdisk

Typically, MS-DOS and ISO9660 formatted disks have their file systems at partition 1. Typically, Drive Setup formatted disks have their file systems starting at partition number 9. Often, if you omit the partition number, Open Firmware looks in the first partition it understands holding a valid file system.

Open Firmware uses a comma (,) to separate the partition number from the filename. It uses a backslash (the \ character) to separate directories. The bootloader uses forward slashes (the / character) to separate directories when specifying the boot-file. Thus, to specify the top of the file system on a CD-ROM in the example PowerMacintosh 7300 system, you'd use: scsi-int/sd@3:,\

Now, to confirm that you and Open Firmware are looking at the same files, you can get a directory listing of the file system on your device with the **dir** command. This command is only useful with file systems that Open Firmware understands and is able to boot from. On Open Firmware 1.0.5, 1.1.22, and 2.0.x systems, you can use **dir** on ISO9660 (not hybrid) and MS-DOS file systems. On Open Firmware 2.4, you can use it on HFS, HFS+, hybrid, ISO9960, and MS-DOS file systems. On Open Firmware 3, you can use it on HFS, HFS+, hybrid (not pure ISO9660), and MS-DOS file systems. The one exception to this rule is that Open Firmware cannot list files on a disk with a 'partition zero' bootloader (including the NetBSD/macppc CD-R image and installation floppies).

```
0 > dir fd:,\
FINDER  .DAT 022 2 2B8
DESKTOP .    022 0 0
RESOURCE.FRK 012 3 0
NETBSD~1.GZ  020 5 1FDFCA
TRASH   .    010 B00 0
OFWBOOT .XCF 020 A75 D8F4
 ok
```

You can see that this MS-DOS formatted disk has a bunch of stuff, as well as the two important files: NETBSD~1.GZ and OFWBOOT.XCF. Note that MacOS shortened netbsd-GENERIC_MD.gz to NETBSD~1.GZ since MS-DOS file systems can only natively hold 8 characters for the filename.

Note: Keep in mind that Open Firmware is often case-sensitive when it comes to filenames.

Note: You may need to append a ;1 to the filename when using a ISO 9660 file system. This "version number" is part of the ISO 9660 specification and will show up in the directory listing if it is present on the disk. For example:

```
0 > boot cd:,\OFWBOOT.XCF;1 NETBSD.MACPPC;1
```

If the **dir** command showed you the files you're looking for, then you've figure out how to tell Open Firmware to look for your bootloader! In this case, your 'boot-device' is 'fd:,\OFWBOOT.XCF' and your 'boot-file' is 'fd:,/NETBSD~1.GZ'.

For additional help, see ofwboot(8) and the FAQ on topics like how to use the Open Firmware command environment and how to boot from a device attached to a PCI card which has Open Firmware support:

http://www.NetBSD.org/ports/macppc/faq.html#ofw-use

http://www.NetBSD.org/ports/macppc/faq.html#boot-pci

Examples of Open Firmware boot commands

Here are some examples of the commands you might use to boot your system:

- *Booting the NetBSD/macppc install CD-R*
  (Open Firmware 1.0.5, Open Firmware 1.1.22, Open Firmware 2.0.x, Open Firmware 2.4)

  Here are some examples of what you might use to boot from CD-ROM (Apple usually sets their CD-ROM drives to SCSI ID 3):

  ```
  0 > boot cd:0 NETBSD.MACPPC
  0 > boot scsi-int/sd@3:0 NETBSD.MACPPC
  0 > boot scsi/sd@3:0 NETBSD.MACPPC
  0 > boot ata/atapi-disk:0 NETBSD.MACPPC
  0 > boot ide1/disk@0:0 NETBSD.MACPPC
  ```

- *Booting the NetBSD/macppc install CD-R*
  (Open Firmware 3)

  ```
  0 > boot cd:,\ofwboot.xcf netbsd.macppc
  ```

- *Booting the NetBSD install floppies*
  (Open Firmware 1.0.5, Open Firmware 1.1.22, Open Firmware 2.0.x, Open Firmware 2.4)

  The first floppy disk has a 'partition zero' bootloader. When it has loaded the kernel, it will ask you to insert the second floppy disk.

  Note: This is not a normal MacOS boot floppy -- you must enter Open Firmware and type a boot command.

  All you need to do is:

  ```
  0 > boot fd:0
  ```

  It is common to see "READ TIMEOUT@" from the floppy. This means that Open Firmware is having difficulty reading the media. See the section below on **Common Problems and Error Messages** for more information about this and what you can do to get a successful boot.

  You can eject a floppy by typing:

  ```
  0 > eject fd
  ```

- *Booting an IDE or SCSI drive with an HFS or HFS+ partition*
  (Open Firmware 2.4, Open Firmware 3)

  MacOS drives have several system-level partitions reserved for MacOS drivers. You may find that your first HFS or HFS+ partition might be as high as partition 9. You may need to keep trying higher partition numbers until you find the one that has your bootloader. You can use **pdisk** to print out your partition map:

  http://www.NetBSD.org/ports/macppc/partitioning.html#pdisk

  You should use the Open Firmware **dir** command to get a directory listing of the files on your hard drive.

  If bootloader is on a different partition from the NetBSD kernel, you will need to specify where to find the kernel.

  Remember, that SCSI Zip disks are usually ID 5 or 6. Internal hard drives are usually SCSI ID 0.

Here are some examples of what you might use to boot the installer kernel located on an HFS or HFS+ partition:

```
0 > boot hd:9,\ofwboot.xcf netbsd-GENERIC_MD.gz
0 > boot scsi/sd@0:9,\ofwboot.xcf netbsd-GENERIC_MD.gz
0 > boot ide0/disk@0:10,\ofwboot.xcf netbsd-GENERIC_MD.gz
0 > boot ultra1:9,ofwboot.xcf netbsd-GENERIC_MD.gz
```

- *Booting a custom CD-ROM*
  (All Open Firmware versions)

  Open Firmware doesn't understand long filenames (created with the RockRidge and Joliet extensions), so you may need to figure out what your file is called. You should use the Open Firmware **dir** command to get a listing of the files on your CD.

  For instance, **Toast** for MacOS creates CDs with long filenames, but uses MS-DOS style short names as well. Instead of referencing `netbsd-GENERIC` you would need to open `NETBSD-G.ENE` The **mkisofs** program has a similar problem, `netbsd-GENERIC_MD.gz` becomes `NETBSD-GENERIC_MD.GZ`.

  Another thing to note is that you *must* use the same case when specifying the filename to load that Open Firmware uses. Also, keep in mind what format your CD-R has to be for your version of Open Firmware (pure ISO versus hybrid ISO/HFS).

  We'll assume that you've wisely renamed your kernel to `netbsd.gz` to avoid many of these name issues. Here are some examples of what you might use to boot from CD-ROM (Apple usually sets their CD-ROM drives to SCSI ID 3):

  ```
  0 > boot cd:,\ofwboot.xcf netbsd.gz
  0 > boot scsi-int/sd@3:,\OFWBOOT.XCF NETBSD.GZ
  0 > boot scsi/sd@3:,\OFWBOOT.XCF NETBSD.GZ
  0 > boot ata/atapi-disk:,\OFWBOOT.XCF NETBSD.GZ
  0 > boot ide1/disk@0:,\OFWBOOT.XCF NETBSD.GZ
  ```

- *Booting an MS-DOS floppy*
  (Open Firmware 1.0.5, Open Firmware 1.1.22, Open Firmware 2.0.x, Open Firmware 2.4)

  Note: This is not a normal MacOS boot floppy -- you must enter Open Firmware and type a boot command.

  Use the Open Firmware **dir** command to get a listing of the files on the floppy. Typically you'll find filenames like `OFWBOOT.XCF` and `NETBSD~1.GZ`.

  ```
  0 > boot fd:,\OFWBOOT.XCF NETBSD~1.GZ
  ```

  You can eject a floppy by typing:

  ```
  0 > eject fd
  ```

- *Booting over the ethernet*
  (Open Firmware 1.0.5, Open Firmware 1.1.22, Open Firmware 2.0.x, Open Firmware 2.4)

  You can try the simple form (i.e. that you are booting from ethernet):

  ```
  0 > boot enet:,ofwboot.xcf
  ```

  Or you may be more specific, specifying the bootloader filename and the kernel name:

  ```
  0 > boot enet:,ofwboot.xcf enet:,/netbsd-GENERIC_MD.gz
  ```

Note: Some Open Firmware 1.0.5 machines have their MAC address stored incorrectly. Make sure that your netboot server is using the same MAC address that your macppc client is using. See the section on **Setting up Open Firmware 1 and 2 to boot NetBSD** to figure out your MAC address.

Note: Some machines cannot load compressed kernels over ethernet. Uncompress them first.

- *Booting over the ethernet*
  (Open Firmware 3)

  You can try the simple form (i.e. that you are booting from ethernet):

  0 > **boot enet:0**

  Or you may be more specific, specifying the bootloader filename and the kernel name:

  0 > **boot enet:0,ofwboot.xcf enet:0,/netbsd-GENERIC_MD.gz**

- *Booting an IDE or SCSI drive with a 'partition zero' bootloader*
  (Open Firmware 1.0.5, Open Firmware 1.1.22, Open Firmware 2.0.x, Open Firmware 2.4)

  You do not specify a file to load, since the 'partition zero' booloader knows what to do. You would boot such a system if you have dumped the CD-R image to your drive, or if you have an already-installed NetBSD/macppc system

  Remember, that SCSI Zip disks are usually ID 5 or 6. Internal hard drives are usually SCSI ID 0.

  Here are some examples of what you might use to boot from such a drive:

  0 > **boot zip:0**
  0 > **boot scsi-int/sd@0:0**
  0 > **boot scsi/sd@0:0**
  0 > **boot ata/ata-disk@0:0**
  0 > **boot ata/ATA-Disk@0:0**
  0 > **boot ide0/disk@0:0**

Example of a normal boot

Of course, a lot of the information in this example depends on your model and what your boot method is, but we'll include this anyways just so you get an idea of what to expect (user-typed commands are in **bold**).

```
Apple PowerBook3,1 2.1f1 BootROM built on 01/29/00 at 22:38:07
Copyright 1994-2000 Apple Computer, Inc.
All Rights Reserved

Welcome to Open Firmware.
To continue booting, type "mac-boot" and press return.
To shut down, type "shut-down" and press return.
 ok
0 > boot enet:,ofwboot.xcf netbsd-GENERIC_MD.gz
loading XCOFF
tsize=C280 dsize=14AC bsize=2620 entry=600000
SECTIONS:
.text    00600000 00600000 0000C280 000000E0
.data    0060D000 0060D000 000014AC 0000C360
.bss     0060E4B0 0060E4B0 00002620 00000000
loading .text, done..
loading .data, done..
clearing .bss, done..
```

```
>> NetBSD/macppc OpenFirmware Boot, Revision 1.3
>> (tsubai@mint.iri.co.jp, Sun Nov 26 01:41:27 JST 2000)
1701508+177748 [100+68176+55886]=0x1e9468

 start=0x100000
Copyright (c) 1996, 1997, 1998, 1999, 2000, 2001
    The NetBSD Foundation, Inc.  All rights reserved.
Copyright (c) 1982, 1986, 1989, 1991, 1993
    The Regents of the University of California.  All rights reserved.

NetBSD 1.5.1 (INSTALL) #0: Thu Mar 15 00:52:56 PST 2001
    mw@al:/usr/src/sys/arch/macppc/compile/INSTALL
total memory = 192 MB
avail memory = 172 MB
using 2483 buffers containing 9932 KB of memory
[...]
erase ^H, werase ^W, kill ^U, intr ^C, status ^T
Terminal type? [vt100]
Erase is backspace.
(I)nstall, (S)hell or (H)alt ?
```

Common Problems and Error Messages

This is a brief list of some of the Open Firmware problems you may run into. See the NetBSD/macppc FAQ for a thorough list.

http://www.NetBSD.org/ports/macppc/faq.html#boot-trouble

Note: You may find it necessary to remove all non-Apple devices from your machine. Some users have found this necessary.

- Black screen

  If your system is a PowerBook, NetBSD may have turned down the brightness of the backlight. Use the buttons to turn your brightness back up.

  Otherwise, you need a serial console. See the section entitled **Open Firmware 1 and 2 System Preparation**

- Grey screen with flashing question mark

  This means that your system is trying to boot MacOS. You might get to this state if you didn't properly enter Open Firmware, or if your Open Firmware boot command has bad syntax.

  If you typed a bad boot command, Open Firmware tries to boot from the value stored in the `boot-device` variable, whose default value is the MacOS ROM.

  The grey screen with the icon is generated by the MacOS ROM. A flashing question-mark or broken folder means that your machine is looking for a bootable MacOS file system, but can't find one. A globe icon means that your machine is looking for a netboot server. A happy face or spinning disc means it's starting to boot MacOS.

  The boot floppy does not have a bootable MacOS file system, it has a bootable NetBSD file system.

  Reboot, re-enter Open Firmware and check your syntax carefully.

Note: Don't forget to check your Open Firmware environment variables, as they may have been changed by your brief excursion into MacOS.

- Information on your screen seems garbled or out of sync

  If you have a PowerMacintosh 7300 through 8600, then you need to read the section on System Disk and the patches it applies. See the section entitled **Open Firmware 1 and 2 System Preparation**

  If you have a Twentieth Anniversary Macintosh, Open Firmware does not work with the internal display, you will need to set up a serial console.

- "`DEFAULT CATCH!`"

  This is a general message from Open Firmware to the effect that it had trouble loading a file.

  If your machine is Open Firmware version 1.0.5, 2.0.x, or 2.4, this error does sometimes appear randomly. You might try the boot command a second time (this is known to work on some models). It's also an indication that either your floppy disk is bad, or the floppy drive is bad. Try doing a low-level format on the floppy, re-copy the files, and try again.

  There are several models that cannot be booted while using the on-board video and keyboard. Try using a serial console.

- "`CLAIM failed`"

  This is a general message from Open Firmware to the effect that it failed to allocate some memory or memory is messed up.

  First, make sure you have the Open Firmware variable `load-base` set correctly. If your system is Open Firmware version 1.0.5, 1.1.22, 2.0.x, or 2.4, see the section above on **Setting up Open Firmware 1 and 2 to boot NetBSD**

  Do not change `load-base` on an Open Firmware 3 system.

  Reset your system and try using a different boot command (this is the only way some people have gotten their Open Firmware 2.4 systems to work):

  0 > **`0 bootr boot-device boot-file`**

  If you tried one boot command and it failed, then you tried a second and got the `CLAIM failed` message, then this is an indication that you should reboot between attempts. Use the Open Firmware **`reset-all`** command.

  Contradictorily, if your machine is Open Firmware version 1.0.5, 2.0.x or 2.4, this error does sometimes appear randomly. You might try the boot command a second time (this is known to work on some models).

  There are several models that cannot be booted while using the on-board video and keyboard. Try using a serial console.

- "`can't OPEN`"

  Open Firmware either can't open the device you specified (because it is not present or the device path is mistyped) or the file you specified. Check your typing and check to make sure that the media has the files you think it has.

  Use the NetBSD **`pdisk`** command to list the partition map and the Open Firmware **`dir`** command to verify that the file(s) you tried accessing are really there.

- "`unrecognized Client Program formatstate not valid`"

This is a general Open Firmware error message indicating that the filename you tried to open either doesn't exist or is in the wrong format. For Open Firmware 1 and 2 machines, it must be an XCOFF file (such as `ofwboot.xcf`) and for Open Firmware 3 machines, it must be either XCOFF or ELF (such as a kernel). Make sure that you have use binary mode to FTP the files, and that they are properly uncompressed.

- "`bad partition number, using 0no bootable HFS partition`"

  If you're trying to boot an Open Firmware 1.0.5, 1.1.22, or 2.0.x system, this probably means that your media (i.e. hard drive or CD-ROM) has an HFS file system on it (such as a hybrid CD-R or a hard drive with MacOS partitions).

- "`READ TIMEOUT@`"

  Open Firmware is having trouble reading your floppy disk. Things to try:
  - Try booting from the floppy again
  - Try a different floppy disk
  - Try writing the floppy on the same machine you're trying to boot
  - Clean the floppy drive
  - Use another boot method

  As disk drives get older, especially with portables, they can get a little bit out of alignment so that they do not consistently read disks written on other floppy drives. Strongly consider writing the floppies on the same machine that will be reading them.

- "`TFTP timeout`"

  Either the server's TFTP server isn't running, or you're using a model with Open Firmware 1.0.5 and not specifying the location of the bootloader. Unfortunately, on the early models, `ofwboot.xcf` gets confused and doesn't work right unless you load it explicitly from the `boot-device`, even if the bootp or DHCP server provides the correct information. You need to boot with a command like:

  0 > **`boot enet:,ofwboot.xcf`**

- "`enet:,/netbsd.ram.gz: Inappropriate file type or format`"

  Some systems booting over ethernet can't load compressed kernels. Use `gunzip`(1) to uncompress the kernel before attempting to netboot.

- Bootloader hangs before the copyright notice and the kernel configuration

  You forgot to set `real-base` in Open Firmware or it got erased by your booting into MacOS.

  Make sure you have the Open Firmware variable `real-base` set correctly. If your system is Open Firmware version 1.0.5, 1.1.22, 2.0.x, or 2.4, see the section above on **Setting up Open Firmware 1 and 2 to boot NetBSD**

  Do not set `real-base` on an Open Firmware 3 system.

- Hang after configuring devices, but before doing anything else

  Actually, this can have many causes. The most likely is a keyboard problem. First, try plugging the USB keyboard directly into the computer (i.e. not through a hub) and unplugging the mouse.

  If you're trying a model that's not on the supported list (such as a new laptop model), they keyboard may not be supported yet.

Milestone

If you've reached this point, then you must've gotten the NetBSD installer to boot. Congratulations! That was the hard part. From now through the rest of this document, there should be no more Open Firmware specific problems, so read everything because it applies to all models.

Running the sysinst installation program

1. *Introduction*

   Using **sysinst**, installing NetBSD is a relatively easy process. Still, you should read this document and have it in hand when doing the installation process. This document tries to be a good guide to the installation, and as such, covers many details for the sake of completeness. Do not let this discourage you; the install program is not hard to use.

2. *General*

   The following is a walk-through of the steps you will take while getting NetBSD installed on your hard disk. **sysinst** is a menu driven installation system that allows for some freedom in doing the installation. Sometimes, questions will be asked and in many cases the default answer will be displayed in brackets ("[ ]") after the question. If you wish to stop the installation, you may press CONTROL-C at any time, but if you do, you'll have to begin the installation process again from scratch by running the /sysinst program from the command prompt. It is not necessary to reboot.

3. *Quick install*

   First, let's describe a quick install. The other sections of this document go into the installation procedure in more detail, but you may find that you do not need this. If you want detailed instructions, skip to the next section. This section describes a basic installation, using a CD-ROM install as an example.

   - What you need.
     - The distribution sets (in this example, they are on CD).
     - Some form of bootable media, described above.
     - A CD-ROM drive (SCSI or ATAPI), a hard disk and a minimum of 16 MB of memory installed.
     - The hard disk should have at least 200 + *n* megabytes of space free, where *n* is the number of megabytes of main memory in your system. If you wish to install the X Window System as well, you will need at least 215 MB more.
   - The Quick Installation
     - Boot the system as described above. You should be at the **sysinst** main menu.

```
.*******************************************.
* NetBSD-5.0.1 Install System                *
*                                            *
*>a: Install NetBSD to hard disk             *
* b: Upgrade NetBSD on a hard disk           *
* c: Re-install sets or install additional sets *
* d: Reboot the computer                     *
* e: Utility menu                            *
* x: Exit Install System                     *
.*******************************************.
```

     - If you wish, you can configure some network settings immediately by choosing the **Utility menu** and then **Configure network**. It isn't actually required at this point, but it may be more convenient. Go back to the main menu.

- Choose **Utility menu** then **Run /bin/sh** for Open Firmware 3 systems. You'll need to create the NetBSD filesystem(s) and an `fstab`. Now exit the shell, return to the main menu, and choose **Re-install**.

  Choose **install** for Open Firmware 1.0.5, 1.1.22, 2.0.x, and 2.4 systems.
- You will be guided through some steps regarding the setup of your disk, and the selection of distributed components to install. When in doubt, refer to the rest of this document for details.
- After your disk has been prepared, choose **CD-ROM** as the medium. The default values for the path and device should be ok.
- After all the files have been unpacked, go back to the main menu and select **reboot.**
- Once the system reaches the Open Firmware prompt, you will need to type the correct command to boot from your hard drive. NetBSD will now boot. If you haven't already done so in **`sysinst`**, you should log in as `root` and set a password for that account. You are also advised to edit `/etc/rc.conf` to match your needs.
- Your installation is now complete.

4. *Booting NetBSD*

   You may want to read the boot messages, to notice your disk's name and capacity. Its name will be something like `sd0` or `wd0` and the geometry will be printed on a line that begins with its name. As mentioned above, you may need your disk's geometry when creating NetBSD's partitions. You will also need to know the name, to tell **`sysinst`** which disk to use. The most important thing to know is that `wd0` is NetBSD's name for your first IDE disk, `wd1` the second, etc. `sd0` is your first SCSI disk, `sd1` the second, etc.

   Once NetBSD has booted and printed all the boot messages, you will be presented with a welcome message and a main menu. It will also include instructions for using the menus.

5. *Network configuration*

   If you do not intend to use networking during the installation, but you do want your machine to be configured for networking once it is installed, you should first go to the **Utility menu** and select the **Configure network** option. If you only want to temporarily use networking during the installation, you can specify these parameters later. If you are not using the Domain Name System (DNS), you can give an empty response when asked to provide a server.

6. *Preparing a disk which will be used for Open Firmware 3 systems*

   Skip this step if you are installing NetBSD on an Open Firmware 1 or 2 system.

   Go to the **Utility Menu**, and select the **Run /bin/sh** option which will give you a shell prompt. From this shell prompt, you will do some of the steps that the normal install procedure runs automatically. Unfortunately, at the moment, our install tools aren't smart enough to deal with drives with Apple Partition Maps and will overwrite important information describing your partitions.

   You may need to type one of the following commands to get your delete key to work properly, depending on your keyboard:

   ```
   # stty erase '^h'
   # stty erase '^?'
   ```

   Type the following command (replacing `wd0` with the name of your destination hard drive):

   ```
   # disklabel wd0
   ```

   This will print out the partition info that was generated by **`pdisk`**, Drive Setup, or Disk Utility. Note that, as discussed above in the **Partitioning your hard drive for NetBSD** section, your *A/UX Root* typi-

cally is the first partition (*a*) and your *A/UX Swap* typically is the second partition (*b*). You may also find that your *A/UX User* partition is the seventh partition (*g*). For example:

```
     # disklabel wd0
[...]
#        size   offset     fstype   [fsize bsize cpg/sgs]
  a:   426613   837432     4.2BSD        0     0     0   # (Cyl. 1622*- 2449*
  b:   204800   632632       swap                        # (Cyl. 1226*- 1622*
  c:  2134305        0     unused        0     0         # (Cyl.    0 - 4136*
  d:   426616     1216        HFS                        # (Cyl.    2*- 829*)
  e:   204800   427832        HFS                        # (Cyl.  829*- 1226*
  f:       21  2134284    unknown                        # (Cyl. 4136*- 4136*
  g:   870239  1264045     4.2BSD        0     0     0   # (Cyl. 2449*- 4136*
disklabel: boot block size 0
disklabel: super block size 0
```

Now, you need to create file systems on the partitions that NetBSD will be using.

Do not modify any partitions labeled `HFS`, `UFS`, *or* `unknown`. The partitions you will be using have their `fstype` listed as `4.2BSD`.

Run the **newfs** command on the `4.2BSD` partitions:

```
     # newfs /dev/wd0a
newfs: /dev/wd0a: not a character-special device
Warning: 120 sector(s) in last cylinder unallocated
/dev/wd0a:      426612 sectors in 827 cylinders of 4 tracks, 129 sectors
        208.3MB in 52 cyl groups (16 c/g, 4.03MB/g, 1024 i/g)
super-block backups (for fsck -b #) at:
     32,   8432,  16832,  25232,  33056,  41456,  49856,  58256,  66080,
  74480,  82880,  91280,  99104, 107504, 115904, 124304, 132128, 140528,
 148928, 157328, 165152, 173552, 181952, 190352, 198176, 206576, 214976,
 223376, 231200, 239600, 248000, 256400, 264224, 272624, 281024, 289424,
 297248, 305648, 314048, 322448, 330272, 338672, 347072, 355472, 363296,
 371696, 380096, 388496, 396320, 404720, 413120, 421520,
newfs: ioctl (WDINFO): Invalid argument
newfs: /dev/wd0a: can't rewrite disk label
```

You can ignore the ~ 'not a character-special device', ~ 'sector(s) in last cylinder unallocated', ~ 'ioctl (WDINFO): Invalid argument', and ~ 'can't rewrite disk label' warnings.

Now you need to mount your destination root partition:

```
# mount /dev/wd0a /mnt
```

Make an `fstab` file for your new system (right now, you only really need to include `/`, `/usr`, and `swap`), for example:

```
# mkdir /mnt/etc
# cat > /mnt/etc/fstab
/dev/wd0a / ffs rw 1 1
/dev/wd0b none swap sw 0 0
/dev/wd0g /usr ffs rw 1 2
```

If you mess up while typing, you can press `CONTROL-U` to erase everything on the current line, or `CONTROL-C` to cancel the file creation, so you can start over. `CONTROL-D` finishes and writes the file to disk.

Great, now create the mountpoints for the file systems you listed in the `fstab`:

```
# mkdir /mnt/usr
```

Clean up and return to **sysinst**:

```
# cd /
# umount /mnt
# exit
```

7. *Installation drive selection and parameters*

   To start the installation onto a dedicated NetBSD drive (Open Firmware 1 or 2), select **Install NetBSD to hard disk** from the main menu. To start the installation onto a drive with an Apple Partition Map (Open Firmware 3), select **Re-install sets or install additional sets** from the main menu.

   The first thing is to identify the disk on which you want to install NetBSD. **sysinst** will report a list of disks it finds and ask you for your selection. You should see disk names like `wd0`, `wd1`, `sd0` or `sd1`.

   If **sysinst** reports

   ```
   I can not find any hard disk for use by NetBSD
   ```

   or the drive you wish to install onto is missing, then you should look at the FAQ entry

   http://www.NetBSD.org/ports/macppc/faq.html#nodisk

8. *Selecting which sets to install*

   The next step is to choose which distribution sets you wish to install. Options are provided for full, minimal, and custom installations. If you choose sets on your own, **base**, **etc**, and a kernel must be selected.

9. *Partitioning the disk*

   You can skip a few steps, down to '*Getting the distribution sets*', if you are installing onto a drive with an Apple Partition Map (Open Firmware 3), i.e. you selected **Re-install sets or install additional sets** from the main menu.

10. *Editing the NetBSD disklabel*

    The partition table of the NetBSD part of a disk is called a *disklabel*. In actuality, NetBSD/macppc uses an Apple Partition Map. The installer creates something like a real Apple Partition Map, but it is not compatible with Mac OS or Open Firmware, which is one of the reasons why you cannot use this installer to partition a disk that can be used with Mac OS or Open Firmware 3 systems.

    If your disk already has a disklabel written to it, you can choose **Use existing partition sizes**. Otherwise, select **Set sizes of NetBSD partitions**.

    After you have chosen your partitions and their sizes (or if you opted to use the existing partitions), you will be presented with the layout of the NetBSD disklabel and given one more chance to change it. For each partition, you can set the type, offset and size, block and fragment size, and the mount point. The type that NetBSD uses for normal file storage is called **4.2BSD**. A swap partition has a special type called **swap**. Some partitions in the disklabel have a fixed purpose.

    `a` Root partition (`/`)

    `b` Swap partition.

    `c` The NetBSD portion of the disk.

    `d`–`p` Available for other use. Traditionally, `g` is the partition mounted on `/usr`, but this is historical practice and not a fixed value.

You will then be asked to name your disk's disklabel. The default response will be ok for most purposes. If you choose to name it something different, make sure the name is a single word and contains no special characters. You don't need to remember this name.

11. *Preparing your hard disk*

    You are now at the point of no return. Nothing has been written to your disk yet, but if you confirm that you want to install NetBSD, your hard drive will be modified. If you are sure you want to proceed, select **yes**.

    The install program will now label your disk and make the file systems you specified. The file systems will be initialized to contain NetBSD bootstrapping binaries and configuration files. You will see messages on your screen from the various NetBSD disk preparation tools that are running. There should be no errors in this section of the installation. If there are, restart from the beginning of the installation process. Otherwise, you can continue the installation program after pressing the return key.

    Note: The bootstrapping code installed in this step will *not* boot a machine with Open Firmware 3. You will still need to have `ofwboot.xcf` on an HFS or HFS+ partition.

12. *Getting the distribution sets*

    The NetBSD distribution consists of a number of *sets* that come in the form of gzipped tarfiles. At this point, you will be presented with a menu which enables you to choose from one of the following methods of installing the sets. Some of these methods will first load the sets on your hard disk, others will extract the sets directly.

    For all these methods, the first step is making the sets available for extraction, and then do the actual installation. The sets can be made available in a few different ways. The following sections describe each of those methods. After reading the one about the method you will be using, you can continue to the section labeled 'Extracting the distribution sets'.

13. *Installation from CD-ROM*

    When installing from a CD-ROM, you will be asked to specify the device name for your CD-ROM drive (usually `cd0`), and the directory name on the CD-ROM where the distribution files are.

    sysinst will then check if the files are indeed available in the specified location, and proceed to the actual extraction of the sets.

14. *Installation using ftp*

    To be able to install using ftp, you first need to configure your network setup if you haven't already done so. **sysinst** will do this for you, asking you if you want to use DHCP. If you do not use DHCP, you can enter network configuration details yourself. If you do not have DNS set up for the machine that you are installing on, you can just press `RETURN` in answer to this question, and DNS will not be used.

    You will also be asked to specify the host that you want to transfer the sets from, the directory on that host, the account name and password used to log into that host using ftp, and optionally a proxy server to use. If you did not set up DNS, you will need to specify an IP address instead of a hostname for the ftp server.

    sysinst will proceed to transfer all the default set files from the remote site to your hard disk.

15. *Installation using NFS*

    To be able to install using NFS, you first need to configure your network setup if you haven't already done so. **sysinst** will do this for you, asking you if you want to use DHCP. If you do not use DHCP, you can enter network configuration details yourself. If you do not have DNS set up for the machine that you are installing on, you can just press `RETURN` in answer to this question, and DNS will not be used.

You will also be asked to specify the host that you want to transfer the sets from and the directory on that host that the files are in. This directory should be mountable by the machine you are installing on, i.e., correctly exported to your machine.

If you did not set up DNS, you will need to specify an IP address instead of a hostname for the NFS server.

16. *Installation from an unmounted file system*

    In order to install from a local file system, you will need to specify the device that the file system resides on (for example `sd1e`) the type of the file system, and the directory on the specified file system where the sets are located. **sysinst** will then check if it can indeed access the sets at that location. Remember, NetBSD/macppc doesn't grok HFS or HFS+ partitions

17. *Installation from a local directory*

    This option assumes that you have already done some preparation yourself. The sets should be located in a directory on a file system that is already accessible. **sysinst** will ask you for the name of this directory.

18. *Extracting the distribution sets*

    Before extraction begins, you can elect to watch the files being extracted; the name of each file that is extracted will be shown. This can slow down the installation process considerably on machines with slow graphics consoles or serial consoles. Alternatively, you can choose to see a progress bar. This is the preferred option as it shows progress without significantly slowing down the installation process.

    If you are installing using the **Re-install sets or install additional sets** option, then you will need to create the device nodes in `/dev`, otherwise, the device node files will be created. If you have already configured networking, you will be asked if you want to use this configuration for normal operation. If so, these values will be installed in the network configuration files. The next menu will allow you to select the time zone that you're in, to make sure your clock has the right offset from UTC. Finally you will be asked to select a password encryption algorithm and can then set a password for the "root" account, to prevent the machine from coming up without access restrictions.

19. *Making the device nodes*

    If you are installing using the **Re-install sets or install additional sets** option, then you will need to create the device nodes in `/dev` now. Otherwise, skip this step.

    Go to the main installation menu, and select **Utility menu** and then select the **Run /bin/sh** option, which will give you a shell prompt. You may need to type one of the following commands to get your delete key to work properly, depending on your keyboard:

    ```
    # stty erase '^h'
    # stty erase '^?'
    ```

    Type the following command (replacing `wd0a` with the partition name of your destination root partition): Now you need to mount your destination root partition:

    ```
    # mount /dev/wd0a /mnt
    # cd /mnt/dev
    # sh MAKEDEV all
    # cd /
    # umount /mnt
    # exit
    ```

20. *Finalizing your installation*

Congratulations, you have successfully installed NetBSD 5.0.1.

Finalizing Open Firmware settings

Now, you can reboot to get to the Open Firmware prompt. You still need to figure out how to get Open Firmware to boot the operating system(s) of your choice. First, try to get NetBSD/macppc running. Using the methods described in the section **Examples of Open Firmware boot commands** figure out the boot command for your installation of NetBSD. Try booting. Once you've got the syntax worked out, decide which operating systems you'll be using regularly.

- *Booting NetBSD exclusively*
  If you'll only be using NetBSD on your macppc system, then simply set the Open Firmware '`boot-device`' and '`boot-file`' variables to the values you just determined. Also, you might want to enable the system to always boot NetBSD when powered on or reset. Once you set up auto-booting you can get to the Open Firmware prompt again by using the `shutdown`(8) command to halt the system.

  If you are not using a 'partition zero' style boot scheme (e.g. Open Firmware 3 models), then you would type something like the following:

  ```
  0 > setenv auto-boot? true
  0 > setenv boot-device ide0/disk@0:8,\ofwboot.xcf
  0 > setenv boot-file ide0/disk@0:13,/netbsd
  0 > reset-all
  ```

  The last command resets the system so that these settings are stored.

  If you are using a 'partition zero' style boot scheme, you would type something like the following:

  ```
  0 > setenv auto-boot? true
  0 > setenv boot-device scsi/sd@0:0
  0 > setenv boot-file netbsd
  0 > reset-all
  ```

  Replace **scsi/sd@0:0** with the actual device you will be booting from.

- *Additional Open Firmware tips*
  If you find that your system tries booting before your hard drive has spun up, try one of the two following '`boot-command`' settings:

  ```
  0 > setenv boot-command catch 5000 ms boot
  0 > setenv boot-command begin ['] boot catch 1000 ms cr again
  ```

  Also, you may be able to pause a system at the Open Firmware prompt if you have '`auto-boot?`' set to '`true`' by holding down any key while the system is resetting. Set the following '`boot-command`' (this may not work on system with Open Firmware 1.0.5) (this method should interrupt booting, even when holding down any key while using a serial console):

  ```
  0 > setenv boot-command key? invert if boot then
  ```

- *Booting NetBSD and MacOS X or Darwin*
  Alas, this takes a little more work. Usually, when you select a system to boot in the "Startup Disk" panel of the "System Preferences" application, it stores the Open Firmware path to that device in the '`boot-device`' variable. So, instead of writing the NetBSD device path to Open Firmware, you'll store the paths to your operating systems in NVRAM. Open Firmware cannot deal with nested **devalias** entries. You must, therefore, enter the *entire* path to your device. See the NetBSD/macppc **nvedit** HOW-TO for more help:
  http://www.NetBSD.org/ports/macppc/nvedit.html

```
0 > printenv boot-device
boot-device /pci@f2000000/mac-io@17/ata-4@1f000/@0:10,\\:tbxi
 ok
0 > nvalias osx /pci@f2000000/mac-io@17/ata-4@1f000/@0:10,\\:tbxi
0 > nvalias bsd /pci@f2000000/mac-io@17/ata-4@1f000/@0:9,ofwboot.xcf
0 > nvstore
0 > setenv use-nvramrc? true
0 > reset-all
```

Now, when the system is reset, it will stop at the Open Firmware prompt and you can type one of the following to boot an operating system:

```
0 > boot osx
0 > boot bsd
```

- *Booting NetBSD and MacOS 9 or earlier*
  For Open Firmware 3 systems, the procedure is identical to the section above on **Booting NetBSD and MacOS X or Darwin**

  For older systems, you're in for more hassle. Booting an Open Firmware 1.0.5, 2.0.x, or 2.4 system into MacOS 9 or earlier will erase some or all of your Open Firmware settings. Try it and see which Open Firmware variables survive.

  If only '`real-base`' is lost, you can compile a kernel that does not require changing the '`real-base`'. Just build a kernel that is less than 4 MB uncompressed. This is easy if you remove all of the USB devices from the config file. Once you've got a smaller kernel, just follow the procedure in the section above on **Booting NetBSD and MacOS X or Darwin**

  If everything is lost when you boot into MacOS 9 or earlier, you will need to make a custom BootVars configuration that you run before trying to boot NetBSD/macppc since System Disk does not preserve enough information when it saves a configuration.

  1. If your system supports System Disk, run it and click "Save" to install the NVRAMRC patches.
  2. Run BootVars, click the "All Variables" button.
  3. Fill in '`real-base`', '`boot-device`', '`boot-file`', '`input-device`', and '`output-device`'.
  4. From the "File" menu, pick "Save". When you run this file, it will load BootVars with all your settings. To boot NetBSD click the "Write&reboot" button.
  5. Now your system will always boot NetBSD. To boot MacOS, use the `shutdown`(8) command to halt the system at the Open Firmware prompt and use the Open Firmware "`bye`" or "`mac-boot`" command to boot MacOS.

- *Other boot techniques*
  See the FAQ for some vague information on how to use the Linux **`yaboot`** system to boot NetBSD.
  http://www.NetBSD.org/ports/macppc/faq.html#yaboot

Post installation steps

Once you've got the operating system running, there are a few things you need to do in order to bring the system into a properly configured state. The most important steps are described below.

1. Configuring `/etc/rc.conf`

   If you or the installation software haven't done any configuration of `/etc/rc.conf` (**`sysinst`** usually will), the system will drop you into single user mode on first reboot with the message

```
/etc/rc.conf is not configured. Multiuser boot aborted.
```

and with the root file system (/) mounted read-only. When the system asks you to choose a shell, simply press RETURN to get to a /bin/sh prompt. If you are asked for a terminal type, respond with **vt100** (or whatever is appropriate for your terminal type) and press RETURN. You may need to type one of the following commands to get your delete key to work properly, depending on your keyboard:

```
# stty erase '^h'
# stty erase '^?'
```

At this point, you need to configure at least one file in the /etc directory. You will need to mount your root file system read/write with:

```
# /sbin/mount -u -w /
```

Change to the /etc directory and take a look at the /etc/rc.conf file. Modify it to your tastes, making sure that you set rc_configured=YES so that your changes will be enabled and a multi-user boot can proceed. Default values for the various programs can be found in /etc/defaults/rc.conf, where some in-line documentation may be found. More complete documentation can be found in rc.conf(5).

When you have finished editing /etc/rc.conf, type **exit** at the prompt to leave the single-user shell and continue with the multi-user boot.

Other values that may need to be set in /etc/rc.conf for a networked environment are *hostname* and possibly *defaultroute*. You may also need to add an *ifconfig_int* for your ⟨int⟩ network interface, where your on-board interfaces may be one of mc0, bm0, tlp0 or gem0. For example:

```
ifconfig_mc0="inet 192.0.2.123 netmask 255.255.255.0"
```

or, if you have *myname.my.dom* in /etc/hosts:

```
ifconfig_mc0="inet myname.my.dom netmask 255.255.255.0"
```

To enable proper hostname resolution, you will also want to add an /etc/resolv.conf file or (if you are feeling a little more adventurous) run named(8). See resolv.conf(5) or named(8) for more information. Instead of manually configuring network and naming service, DHCP can be used by setting dhclient=YES in /etc/rc.conf.

Other files in /etc that may require modification or setting up include /etc/mailer.conf, /etc/nsswitch.conf, and /etc/wscons.conf.

2. Logging in

   After reboot, you can log in as root at the login prompt. Unless you've set a password in **sysinst**, there is no initial password. You should create an account for yourself (see below) and protect it and the "root" account with good passwords. By default, root login from the network is disabled (even via ssh(1)). One way to become root over the network is to log in as a different user that belongs to group "wheel" (see group(5)) and use su(1) to become root.

3. Adding accounts

   Use the useradd(8) command to add accounts to your system. Do not edit /etc/passwd directly! See vipw(8) and pwd_mkdb(8) if you want to edit the password database.

4. The X Window System

   If you installed the X Window System, you may want to read the chapter about X in the NetBSD Guide:
   http://netbsd.org/docs/guide/en/chap-x.html

5. Installing third party packages

If you wish to install any of the software freely available for UNIX-like systems you are strongly advised to first check the NetBSD package system, pkgsrc. pkgsrc automatically handles any changes necessary to make the software run on NetBSD. This includes the retrieval and installation of any other packages on which the software may depend.

- More information on the package system is available at
  http://www.NetBSD.org/docs/software/packages.html
- A list of available packages suitable for browsing is at
  ftp://ftp.NetBSD.org/pub/NetBSD/packages/pkgsrc/README.html
- Precompiled binaries can be found at
  ftp://ftp.NetBSD.org/pub/pkgsrc/packages/NetBSD/
  usually in the `macppc/5.0.1/All` subdir. You can install them with the following commands under `sh`(1):

```
# PKG_PATH=ftp://ftp.NetBSD.org/pub/pkgsrc/packages/NetBSD/macppc/5.0.1/All
# export PKG_PATH
# pkg_add -v tcsh
# pkg_add -v bash
# pkg_add -v perl
# pkg_add -v apache
# pkg_add -v kde
# pkg_add -v firefox
...
```

  If you are using `csh`(1) then replace the first two lines with the following:

```
# setenv PKG_PATH ftp://ftp.NetBSD.org/pub/pkgsrc/packages/NetBSD/macppc/5.0.1/
```

  Note: Some mirror sites don't mirror the `/pub/pkgsrc` directory. If you would like to use such mirrors, you could also try the `/pub/NetBSD/packages/current-packages/NetBSD/macppc/5.0.1/All` directory, which may have the same contents.

  The above commands will install the Tenex-csh and Bourne Again shells, the Perl programming language, Apache web server, KDE desktop environment and the Firefox web browser as well as all the packages they depend on.

  Note: In some cases the `pkg_add`(1) command will complain about a version mismatch of packages with a message like the following:

```
Warning: package 'foo' was built for a different version
of the OS:
NetBSD/i386 M.N (pkg) vs. NetBSD/i386 5.0.1 (this host),
```

  This warning is harmless if the formal major release numbers are the same between the pkg and your host. Please refer to the NetBSD release glossary and graphs at
  http://www.NetBSD.org/releases/release-map.html
  for more information about NetBSD's release numbering scheme.
- The framework for compiling packages can be obtained by retrieving the file
  ftp://ftp.NetBSD.org/pub/NetBSD/packages/pkgsrc.tar.gz
  It is typically extracted into `/usr/pkgsrc` (though other locations work fine) with the commands:

```
# cd /usr
# tar -zxpf pkgsrc.tar.gz
```

After extracting, see the `doc/pkgsrc.txt` file in the extraction directory (e.g., `/usr/pkgsrc/doc/pkgsrc.txt`) for more information.

6. Misc

   - Edit `/etc/mail/aliases` to forward root mail to the right place. Don't forget to run `newaliases`(1) afterwards.
   - The `/etc/postfix/main.cf` file will almost definitely need to be adjusted. If you prefer a different MTA, then install it using pkgsrc or by hand and adjust `/etc/mailer.conf`.
   - Edit `/etc/rc.local` to run any local daemons you use.
   - Many of the `/etc` files are documented in section 5 of the manual; so just invoking

     ```
     # man 5 filename
     ```

     is likely to give you more information on these files.

Upgrading a previously-installed NetBSD System

The easiest way to upgrade to NetBSD 5.0.1 is with binaries, and that is the method documented here.

To do the upgrade, you must boot the install kernel using one of the methods described above. You must also have at least the **base** and **kern** binary distribution sets available. Finally, you must have sufficient disk space available to install the new binaries. Since files already installed on the system are overwritten in place, you only need additional free space for files which weren't previously installed or to account for growth of the sets between releases. If you have a few megabytes free on each of your root (`/`) and `/usr` partitions, you should have enough space.

Since upgrading involves replacing the kernel, boot blocks, and most of the system binaries, it has the potential to cause data loss. You are strongly advised to *back up* any important data on the NetBSD partition or on another operating system's partition on your disk before beginning the upgrade process. Since installation of the bootloader will prevent Mac OS from using the disk and will prevent Open Firmware 3 machines from booting, there is an option to bypass this step. Make sure you know whether or not to install the bootloader.

The upgrade procedure is similar to an installation, but without the hard disk partitioning. **sysinst** will attempt to merge the settings stored in your `/etc` directory with the new version of NetBSD. Also, file systems are checked before unpacking the sets. Fetching the binary sets is done in the same manner as the installation procedure; refer to the installation part of the document for help.

After a new kernel has been copied to your hard disk, your machine is a complete NetBSD 5.0.1 system. However, that doesn't mean that you're finished with the upgrade process. You will probably want to update the set of device nodes you have in `/dev`. If you've changed the contents of `/dev` by hand, you will need to be careful about this, but if not, you can just cd into `/dev`, and run the command:

```
# sh MAKEDEV all
```

Finally, you will want to delete old binaries that were part of the version of NetBSD that you upgraded from and have since been removed from the NetBSD distribution.

Compatibility Issues With Previous NetBSD Releases

Users upgrading from previous versions of NetBSD may wish to bear the following problems and compatibility issues in mind when upgrading to NetBSD 5.0.1.

pkg_install now depends on the pkgdb cache for automatic conflict detection. It is recommended to rebuild the cache with

```
# pkg_admin rebuild
```

`audit-packages.conf`(5) has been superseded by `pkg_install.conf`(5). The default configuration is the same.

Support for `pkg_view`(1) has been retired.

The functionality of `audit-packages`(1) and `download-vulnerability-list`(1) has moved into `pkg_admin`(1). However, wrapper scripts that handle the common use cases are provided.

The pthread libraries from previous versions of NetBSD require that the `sysctl`(3) node `kern.no_sa_support` be set to `0`. This affects the following environments:

- Running a 5.0 kernel with an older userland.
- Running an older userland inside a chroot'ed environment on a 5.0 system.
- Running older statically linked pthread applications.

The 5.0 kernel defaults to `0` for `kern.no_sa_support`, which covers the first case. However, please note that a full installation of 5.0 (either from scratch or through an upgrade) will set `kern.no_sa_support` to 1 during the boot process. This means that for the last two cases, you will have to manually set `kern.no_sa_support` to `0`, using either the `sysctl`(8) command or through `sysctl.conf`(5).

Note that **`sysinst`** will automatically invoke

```
postinstall fix
```

and thus all issues that are fixed by **`postinstall`** by default (see below) will be handled.

Issues affecting an upgrade from NetBSD 3.x releases

See the section below on upgrading from NetBSD 4.x as well.

The following issues can generally be resolved by running *postinstall* with the **etc** set:

```
postinstall -s /path/to/etc.tgz check
postinstall -s /path/to/etc.tgz fix
```

Issues fixed by *postinstall*:

- Various files in `/etc` need upgrading. These include:
  - `/etc/defaults/*`
  - `/etc/mtree/*`
  - `/etc/daily`
  - `/etc/weekly`
  - `/etc/monthly`
  - `/etc/security`
  - `/etc/rc.subr`
  - `/etc/rc`
  - `/etc/rc.shutdown`
  - `/etc/rc.d/*`
  - `/etc/envsys.conf`

The following issues need to be resolved manually:

- The users '_proxy', '_rwhod', and '_sdpd' and the groups '_proxy', '_rwhod' and '_sdpd' need to be created and the user 'uucp' needs to be updated.
- A number of things were removed in the NetBSD 4.0 release, including: the evbsh5 port, the Fortran 77 compiler (g77), NETCCITT, NETNS, Sendmail, Sushi, UUCP, and Vinum. If you were using any of these, please see the "Components removed from NetBSD" at
  http://www.NetBSD.org/releases/formal-4/NetBSD-4.0.html#removals

- The replacement of Sendmail by Postfix can be handled automatically by *postinstall* but it is not done by default. If you want to transition to Postfix, the command

  ```
  postinstall -s /path/to/etc.tgz fix mailerconf
  ```

  will update your `/etc/mailer.conf` file to use Postfix as the MTA. When using **sysinst** to upgrade the system, it will ask if you want this to be done.

  Note that if you have a customized Sendmail setup, you need to set up Postfix in an equivalent way; there is no tool for automatic conversion of Sendmail configuration to a Postfix one.

  Postfix will be started automatically when the system boots. You may see messages like "$sendmail is not set properly" at boot. You can suppress them by removing `/etc/rc.d/sendmail` and `/etc/rc.d/smmsp`. Those files and other parts of sendmail configuration like files under `/usr/share/sendmail` are not removed by default while upgrading for those who want to continue using sendmail from outside the base system. If you want to delete them, *postinstall* can be used:

  ```
  postinstall -s /path/to/etc.tgz fix sendmail
  ```

Issues affecting an upgrade from NetBSD 4.x releases

The following issues can generally be resolved by running *postinstall* with the **etc** set:

```
postinstall -s /path/to/etc.tgz check
postinstall -s /path/to/etc.tgz fix
```

Issues fixed by *postinstall*:

- Various files in `/etc` need upgrading. These include:
  - `/etc/defaults/*`
  - `/etc/mtree/*`
  - `/etc/daily`
  - `/etc/weekly`
  - `/etc/monthly`
  - `/etc/security`
  - `/etc/rc.subr`
  - `/etc/rc`
  - `/etc/rc.shutdown`
  - `/etc/rc.d/*`
  - `/etc/envsys.conf`

The following issues need to be resolved manually:

- The users '_httpd' and '_timedc' and the groups '_httpd' and '_timedc' need to be created.
- Unprivileged use of the `mount`(8) command now requires the *`nosuid`* and *`nodev`* options to be explicitly specified. Previously, these options were automatically enforced even if they were not explicitly specified.
- A number of things have been removed from the NetBSD 5.0.1 release. See the "Components removed from NetBSD" section near the beginning of this document for a list.

Using online NetBSD documentation

Documentation is available if you installed the manual distribution set. Traditionally, the "man pages" (documentation) are denoted by '`name(section)`'. Some examples of this are

- intro(1),
- man(1),
- apropos(1),
- passwd(1), and
- passwd(5).

The section numbers group the topics into several categories, but three are of primary interest: user commands are in section 1, file formats are in section 5, and administrative information is in section 8.

The *man* command is used to view the documentation on a topic, and is started by entering **man** [*section*] *topic*. The brackets [] around the section should not be entered, but rather indicate that the section is optional. If you don't ask for a particular section, the topic with the lowest numbered section name will be displayed. For instance, after logging in, enter

```
# man passwd
```

to read the documentation for passwd(1). To view the documentation for passwd(5), enter

```
# man 5 passwd
```

instead.

If you are unsure of what man page you are looking for, enter

```
# apropos subject-word
```

where *subject-word* is your topic of interest; a list of possibly related man pages will be displayed.

Administrivia

If you've got something to say, do so! We'd like your input. There are various mailing lists available via the mailing list server at majordomo@NetBSD.org. To get help on using the mailing list server, send mail to that address with an empty body, and it will reply with instructions. See **http://www.NetBSD.org/mailinglists/** for a web interface.

There are various mailing lists set up to deal with comments and questions about this release. Please send comments to: netbsd-comments@NetBSD.org.

To report bugs, use the send-pr(1) command shipped with NetBSD, and fill in as much information about the problem as you can. Good bug reports include lots of details.

Bugs also can be submitted and queried with the web interface at
http://www.NetBSD.org/support/send-pr.html

There are also port-specific mailing lists, to discuss aspects of each port of NetBSD. Use majordomo to find their addresses, or visit **http://www.NetBSD.org/mailinglists/**

If you're interested in doing a serious amount of work on a specific port, you probably should contact the 'owner' of that port (listed below).

If you'd like to help with this effort, and have an idea as to how you could be useful, send us mail or subscribe to: netbsd-users@NetBSD.org.

As a favor, please avoid mailing huge documents or files to these mailing lists. Instead, put the material you would have sent up for FTP or WWW somewhere, then mail the appropriate list about it, or, if you'd rather not do that, mail the list saying you'll send the data to those who want it.

Thanks go to

- The former members of UCB's Computer Systems Research Group, including (but not limited to):

  Keith Bostic
  Ralph Campbell
  Mike Karels
  Marshall Kirk McKusick

  for their work on BSD systems, support, and encouragement.
- The Internet Systems Consortium, Inc. for hosting the NetBSD FTP, CVS, AnonCVS, mail, mail archive, GNATS, SUP, Rsync and WWW servers.
- The Internet Research Institute in Japan for hosting the server which runs the CVSweb interface to the NetBSD source tree.
- The Luleå Academic Computer Society for providing the backup services server.
- The Columbia University Computer Science Department for hosting the NYC build cluster.
- The Western Washington University Computer Science Department for running the WWU build cluster.
- The many organizations that provide NetBSD mirror sites.
- Without CVS, this project would be impossible to manage, so our hats go off to Brian Berliner, Jeff Polk, and the various other people who've had a hand in making CVS a useful tool.
- We list the individuals and organizations that have made donations or loans of hardware and/or money, to support NetBSD development, and deserve credit for it at **http://www.NetBSD.org/donations/** (If you're not on that list and should be, tell us! We probably were not able to get in touch with you, to verify that you wanted to be listed.)
- Finally, we thank all of the people who've put sweat and tears into developing NetBSD since its inception in January, 1993. (Obviously, there are a lot more people who deserve thanks here. If you're one of them, and would like to be mentioned, tell us!)

We are...

(in alphabetical order)

The NetBSD core group:

Alistair Crooks	agc@NetBSD.org
Quentin Garnier	cube@NetBSD.org
Matt Thomas	matt@NetBSD.org
YAMAMOTO Takashi	yamt@NetBSD.org
Christos Zoulas	christos@NetBSD.org

The portmasters (and their ports):

Erik Berls	cyber@NetBSD.org	**cobalt**
Manuel Bouyer	bouyer@NetBSD.org	**xen**
Simon Burge	simonb@NetBSD.org	**evbmips**
Simon Burge	simonb@NetBSD.org	**pmax**
Simon Burge	simonb@NetBSD.org	**sbmips**
Julian Coleman	jdc@NetBSD.org	**atari**
Marcus Comstedt	marcus@NetBSD.org	**dreamcast**
Andrew Doran	ad@NetBSD.org	**amd64**
Andrew Doran	ad@NetBSD.org	**i386**
Matthias Drochner	drochner@NetBSD.org	**cesfic**
Gavan Fantom	gavan@NetBSD.org	**iyonix**
Jaime A Fournier	ober@NetBSD.org	**zaurus**
Matt Fredette	fredette@NetBSD.org	**sun2**
Ichiro FUKUHARA	ichiro@NetBSD.org	**hpcarm**
Chris Gilbert	chris@NetBSD.org	**cats**
Ben Harris	bjh21@NetBSD.org	**acorn26**
Ross Harvey	ross@NetBSD.org	**alpha**
Nick Hudson	skrll@NetBSD.org	**hp700**
Martin Husemann	martin@NetBSD.org	**sparc64**
IWAMOTO Toshihiro	toshii@NetBSD.org	**hpcarm**
Darrin Jewell	dbj@NetBSD.org	**next68k**
Søren Jørvang	soren@NetBSD.org	**sgimips**
Wayne Knowles	wdk@NetBSD.org	**mipsco**
Takayoshi Kochi	kochi@NetBSD.org	**ia64**
Paul Kranenburg	pk@NetBSD.org	**sparc**
Michael Lorenz	macallan@NetBSD.org	**macppc**
Anders Magnusson	ragge@NetBSD.org	**vax**
Cherry G. Mathew	cherry@NetBSD.org	**ia64**
NISHIMURA Takeshi	nsmrtks@NetBSD.org	**x68k**
Tohru Nishimura	nisimura@NetBSD.org	**luna68k**
Tohru Nishimura	nisimura@NetBSD.org	**sandpoint**
Andrey Petrov	petrov@NetBSD.org	**sparc64**
Scott Reynolds	scottr@NetBSD.org	**mac68k**
Tim Rightnour	garbled@NetBSD.org	**ofppc**
Tim Rightnour	garbled@NetBSD.org	**prep**
Tim Rightnour	garbled@NetBSD.org	**rs6000**
Noriyuki Soda	soda@NetBSD.org	**arc**
Ignatios Souvatzis	is@NetBSD.org	**amiga**
Jonathan Stone	jonathan@NetBSD.org	**pmax**
Shin Takemura	takemura@NetBSD.org	**hpcmips**
Matt Thomas	matt@NetBSD.org	**alpha**
Matt Thomas	matt@NetBSD.org	**netwinder**
Jason Thorpe	thorpej@NetBSD.org	**algor**
Jason Thorpe	thorpej@NetBSD.org	**evbarm**
Jason Thorpe	thorpej@NetBSD.org	**shark**
Izumi Tsutsui	tsutsui@NetBSD.org	**ews4800mips**
Izumi Tsutsui	tsutsui@NetBSD.org	**hp300**
Izumi Tsutsui	tsutsui@NetBSD.org	**news68k**

Valeriy E. Ushakov	uwe@NetBSD.org	**landisk**
Nathan Williams	nathanw@NetBSD.org	**sun3**
Steve Woodford	scw@NetBSD.org	**evbppc**
Steve Woodford	scw@NetBSD.org	**mvme68k**
Steve Woodford	scw@NetBSD.org	**mvmeppc**
Reinoud Zandijk	reinoud@NetBSD.org	**acorn32**

The NetBSD 5.0.1 Release Engineering team:

Manuel Bouyer	bouyer@NetBSD.org
David Brownlee	abs@NetBSD.org
James Chacon	jmc@NetBSD.org
Julian Coleman	jdc@NetBSD.org
Håvard Eidnes	he@NetBSD.org
Liam J. Foy	liamjfoy@NetBSD.org
John Heasley	heas@NetBSD.org
Geert Hendrickx	ghen@NetBSD.org
Soren Jacobsen	snj@NetBSD.org
Phil Nelson	phil@NetBSD.org
Jeff Rizzo	riz@NetBSD.org

NetBSD Developers:

Nathan Ahlstrom	nra@NetBSD.org
Steve Allen	wormey@NetBSD.org
Jukka Andberg	jandberg@NetBSD.org
Julian Assange	proff@NetBSD.org
Lennart Augustsson	augustss@NetBSD.org
Christoph Badura	bad@NetBSD.org
Bang Jun-Young	junyoung@NetBSD.org
Dieter Baron	dillo@NetBSD.org
Robert V. Baron	rvb@NetBSD.org
Alan Barrett	apb@NetBSD.org
Grant Beattie	grant@NetBSD.org
Jason Beegan	jtb@NetBSD.org
Erik Berls	cyber@NetBSD.org
Hiroyuki Bessho	bsh@NetBSD.org
John Birrell	jb@NetBSD.org
Mason Loring Bliss	mason@NetBSD.org
Charles Blundell	cb@NetBSD.org
Rafal Boni	rafal@NetBSD.org
Stephen Borrill	sborrill@NetBSD.org
Sean Boudreau	seanb@NetBSD.org
Manuel Bouyer	bouyer@NetBSD.org
John Brezak	brezak@NetBSD.org
Allen Briggs	briggs@NetBSD.org
Mark Brinicombe	mark@NetBSD.org
Aaron Brown	abrown@NetBSD.org
Andrew Brown	atatat@NetBSD.org

David Brownlee	abs@NetBSD.org
Frederick Bruckman	fredb@NetBSD.org
Jon Buller	jonb@NetBSD.org
Simon Burge	simonb@NetBSD.org
Robert Byrnes	byrnes@NetBSD.org
Pavel Cahyna	pavel@NetBSD.org
D'Arcy J.M. Cain	darcy@NetBSD.org
Daniel Carosone	dan@NetBSD.org
Dave Carrel	carrel@NetBSD.org
James Chacon	jmc@NetBSD.org
Mihai Chelaru	kefren@NetBSD.org
Bill Coldwell	billc@NetBSD.org
Julian Coleman	jdc@NetBSD.org
Ben Collver	ben@NetBSD.org
Marcus Comstedt	marcus@NetBSD.org
Jeremy Cooper	jeremy@NetBSD.org
Chuck Cranor	chuck@NetBSD.org
Alistair Crooks	agc@NetBSD.org
Aidan Cully	aidan@NetBSD.org
Garrett D'Amore	gdamore@NetBSD.org
Johan Danielsson	joda@NetBSD.org
John Darrow	jdarrow@NetBSD.org
Jed Davis	jld@NetBSD.org
Matt DeBergalis	deberg@NetBSD.org
Arnaud Degroote	degroote@NetBSD.org
Rob Deker	deker@NetBSD.org
Chris G. Demetriou	cgd@NetBSD.org
Tracy Di Marco White	gendalia@NetBSD.org
Jaromír Dolecek	jdolecek@NetBSD.org
Andy Doran	ad@NetBSD.org
Roland Dowdeswell	elric@NetBSD.org
Emmanuel Dreyfus	manu@NetBSD.org
Matthias Drochner	drochner@NetBSD.org
Jun Ebihara	jun@NetBSD.org
Håvard Eidnes	he@NetBSD.org
Jaime A Fournier	ober@NetBSD.org
Stoned Elipot	seb@NetBSD.org
Michael van Elst	mlelstv@NetBSD.org
Enami Tsugutomo	enami@NetBSD.org
Bernd Ernesti	veego@NetBSD.org
Erik Fair	fair@NetBSD.org
Gavan Fantom	gavan@NetBSD.org
Hauke Fath	hauke@NetBSD.org
Hubert Feyrer	hubertf@NetBSD.org
Jason R. Fink	jrf@NetBSD.org
Matt J. Fleming	mjf@NetBSD.org
Marty Fouts	marty@NetBSD.org
Liam J. Foy	liamjfoy@NetBSD.org
Matt Fredette	fredette@NetBSD.org

Thorsten Frueauf	frueauf@NetBSD.org
Castor Fu	castor@NetBSD.org
Ichiro Fukuhara	ichiro@NetBSD.org
Quentin Garnier	cube@NetBSD.org
Thomas Gerner	thomas@NetBSD.org
Simon J. Gerraty	sjg@NetBSD.org
Justin Gibbs	gibbs@NetBSD.org
Chris Gilbert	chris@NetBSD.org
Eric Gillespie	epg@NetBSD.org
Brian Ginsbach	ginsbach@NetBSD.org
Paul Goyette	pgoyette@NetBSD.org
Michael Graff	explorer@NetBSD.org
Brian C. Grayson	bgrayson@NetBSD.org
Matthew Green	mrg@NetBSD.org
Andreas Gustafsson	gson@NetBSD.org
Ulrich Habel	rhaen@NetBSD.org
Jun-ichiro itojun Hagino	itojun@NetBSD.org
HAMAJIMA Katsuomi	hamajima@NetBSD.org
Adam Hamsik	haad@NetBSD.org
Juergen Hannken-Illjes	hannken@NetBSD.org
Charles M. Hannum	mycroft@NetBSD.org
Ben Harris	bjh21@NetBSD.org
Ross Harvey	ross@NetBSD.org
Eric Haszlakiewicz	erh@NetBSD.org
John Hawkinson	jhawk@NetBSD.org
HAYAKAWA Koichi	haya@NetBSD.org
John Heasley	heas@NetBSD.org
Geert Hendrickx	ghen@NetBSD.org
René Hexel	rh@NetBSD.org
Iain Hibbert	plunky@NetBSD.org
Kouichirou Hiratsuka	hira@NetBSD.org
Michael L. Hitch	mhitch@NetBSD.org
Ádám Hóka	ahoka@NetBSD.org
Jachym Holecek	freza@NetBSD.org
David A. Holland	dholland@NetBSD.org
Christian E. Hopps	chopps@NetBSD.org
Ken Hornstein	kenh@NetBSD.org
Marc Horowitz	marc@NetBSD.org
Eduardo Horvath	eeh@NetBSD.org
Nick Hudson	skrll@NetBSD.org
Shell Hung	shell@NetBSD.org
Martin Husemann	martin@NetBSD.org
Dean Huxley	dean@NetBSD.org
Love Hörnquist Åstrand	lha@NetBSD.org
Roland Illig	rillig@NetBSD.org
Bernardo Innocenti	bernie@NetBSD.org
Tetsuya Isaki	isaki@NetBSD.org
ITOH Yasufumi	itohy@NetBSD.org
IWAMOTO Toshihiro	toshii@NetBSD.org

Matthew Jacob	mjacob@NetBSD.org
Soren Jacobsen	snj@NetBSD.org
Lonhyn T. Jasinskyj	lonhyn@NetBSD.org
Darrin Jewell	dbj@NetBSD.org
Nicolas Joly	njoly@NetBSD.org
Chris Jones	cjones@NetBSD.org
Søren Jørvang	soren@NetBSD.org
Takahiro Kambe	taca@NetBSD.org
Masanori Kanaoka	kanaoka@NetBSD.org
Antti Kantee	pooka@NetBSD.org
Frank Kardel	kardel@NetBSD.org
Mattias Karlsson	keihan@NetBSD.org
KAWAMOTO Yosihisa	kawamoto@NetBSD.org
Mario Kemper	magick@NetBSD.org
Min Sik Kim	minskim@NetBSD.org
Thomas Klausner	wiz@NetBSD.org
Klaus Klein	kleink@NetBSD.org
John Klos	jklos@NetBSD.org
Wayne Knowles	wdk@NetBSD.org
Takayoshi Kochi	kochi@NetBSD.org
John Kohl	jtk@NetBSD.org
Daniel de Kok	daniel@NetBSD.org
Jonathan A. Kollasch	jakllsch@NetBSD.org
Paul Kranenburg	pk@NetBSD.org
Lubomir Kundrak	lkundrak@NetBSD.org
Jochen Kunz	jkunz@NetBSD.org
Martti Kuparinen	martti@NetBSD.org
Kentaro A. Kurahone	kurahone@NetBSD.org
Arnaud Lacombe	alc@NetBSD.org
Kevin Lahey	kml@NetBSD.org
David Laight	dsl@NetBSD.org
Johnny C. Lam	jlam@NetBSD.org
Martin J. Laubach	mjl@NetBSD.org
Greg Lehey	grog@NetBSD.org
Ted Lemon	mellon@NetBSD.org
Christian Limpach	cl@NetBSD.org
Frank van der Linden	fvdl@NetBSD.org
Joel Lindholm	joel@NetBSD.org
Tonnerre Lombard	tonnerre@NetBSD.org
Mike Long	mikel@NetBSD.org
Michael Lorenz	macallan@NetBSD.org
Warner Losh	imp@NetBSD.org
Tomasz Luchowski	zuntum@NetBSD.org
Federico Lupi	federico@NetBSD.org
Brett Lymn	blymn@NetBSD.org
Paul Mackerras	paulus@NetBSD.org
MAEKAWA Masahide	gehenna@NetBSD.org
Anders Magnusson	ragge@NetBSD.org
Cherry G. Mathew	cherry@NetBSD.org

David Maxwell	david@NetBSD.org
Gregory McGarry	gmcgarry@NetBSD.org
Dan McMahill	dmcmahill@NetBSD.org
Jared D. McNeill	jmcneill@NetBSD.org
Neil J. McRae	neil@NetBSD.org
Julio M. Merino Vidal	jmmv@NetBSD.org
Perry Metzger	perry@NetBSD.org
Luke Mewburn	lukem@NetBSD.org
Jean-Yves Migeon	jym@NetBSD.org
Brook Milligan	brook@NetBSD.org
Minoura Makoto	minoura@NetBSD.org
Simas Mockevicius	symka@NetBSD.org
der Mouse	mouse@NetBSD.org
Joseph Myers	jsm@NetBSD.org
Ken Nakata	kenn@NetBSD.org
Takeshi Nakayama	nakayama@NetBSD.org
Phil Nelson	phil@NetBSD.org
John Nemeth	jnemeth@NetBSD.org
Bob Nestor	rnestor@NetBSD.org
NISHIMURA Takeshi	nsmrtks@NetBSD.org
Tohru Nishimura	nisimura@NetBSD.org
NONAKA Kimihiro	nonaka@NetBSD.org
Takehiko NOZAKI	tnozaki@NetBSD.org
Tobias Nygren	tnn@NetBSD.org
OBATA Akio	obache@NetBSD.org
Jesse Off	joff@NetBSD.org
Tatoku Ogaito	tacha@NetBSD.org
OKANO Takayoshi	kano@NetBSD.org
Masaru Oki	oki@NetBSD.org
Atsushi Onoe	onoe@NetBSD.org
Greg Oster	oster@NetBSD.org
Rui Paulo	rpaulo@NetBSD.org
Jonathan Perkin	sketch@NetBSD.org
Andrey Petrov	petrov@NetBSD.org
Herb Peyerl	hpeyerl@NetBSD.org
Matthias Pfaller	matthias@NetBSD.org
Chris Pinnock	cjep@NetBSD.org
Adrian Portelli	adrianp@NetBSD.org
Peter Postma	peter@NetBSD.org
Dante Profeta	dante@NetBSD.org
Chris Provenzano	proven@NetBSD.org
Niels Provos	provos@NetBSD.org
Mindaugas Rasiukevicius	rmind@NetBSD.org
Michael Rauch	mrauch@NetBSD.org
Marc Recht	recht@NetBSD.org
Darren Reed	darrenr@NetBSD.org
Jeremy C. Reed	reed@NetBSD.org
Antoine Reilles	tonio@NetBSD.org
Tyler R. Retzlaff	rtr@NetBSD.org

Scott Reynolds	scottr@NetBSD.org
Michael Richardson	mcr@NetBSD.org
Tim Rightnour	garbled@NetBSD.org
Alan Ritter	rittera@NetBSD.org
Jeff Rizzo	riz@NetBSD.org
Hans Rosenfeld	hans@NetBSD.org
Gordon Ross	gwr@NetBSD.org
Steve Rumble	rumble@NetBSD.org
Ilpo Ruotsalainen	lonewolf@NetBSD.org
Heiko W. Rupp	hwr@NetBSD.org
Blair J. Sadewitz	bjs@NetBSD.org
David Sainty	dsainty@NetBSD.org
SAITOH Masanobu	msaitoh@NetBSD.org
Kazuki Sakamoto	sakamoto@NetBSD.org
Curt Sampson	cjs@NetBSD.org
Wilfredo Sanchez	wsanchez@NetBSD.org
Ty Sarna	tsarna@NetBSD.org
SATO Kazumi	sato@NetBSD.org
Jan Schaumann	jschauma@NetBSD.org
Matthias Scheler	tron@NetBSD.org
Silke Scheler	silke@NetBSD.org
Karl Schilke (rAT)	rat@NetBSD.org
Amitai Schlair	schmonz@NetBSD.org
Konrad Schroder	perseant@NetBSD.org
Georg Schwarz	schwarz@NetBSD.org
Lubomir Sedlacik	salo@NetBSD.org
Christopher SEKIYA	sekiya@NetBSD.org
Reed Shadgett	dent@NetBSD.org
John Shannon	shannonjr@NetBSD.org
Tim Shepard	shep@NetBSD.org
Takeshi Shibagaki	shiba@NetBSD.org
Naoto Shimazaki	igy@NetBSD.org
Takao Shinohara	shin@NetBSD.org
Takuya SHIOZAKI	tshiozak@NetBSD.org
Daniel Sieger	dsieger@NetBSD.org
Chuck Silvers	chs@NetBSD.org
Thor Lancelot Simon	tls@NetBSD.org
Jeff Smith	jeffs@NetBSD.org
Noriyuki Soda	soda@NetBSD.org
Wolfgang Solfrank	ws@NetBSD.org
SOMEYA Yoshihiko	someya@NetBSD.org
Bill Sommerfeld	sommerfeld@NetBSD.org
Jörg Sonnenberger	joerg@NetBSD.org
Ignatios Souvatzis	is@NetBSD.org
T K Spindler	dogcow@NetBSD.org
Bill Squier	groo@NetBSD.org
Jonathan Stone	jonathan@NetBSD.org
Bill Studenmund	wrstuden@NetBSD.org
Kevin Sullivan	sullivan@NetBSD.org

SUNAGAWA Keiki	kei@NetBSD.org
Kimmo Suominen	kim@NetBSD.org
Robert Swindells	rjs@NetBSD.org
Shin Takemura	takemura@NetBSD.org
TAMURA Kent	kent@NetBSD.org
Shin'ichiro TAYA	taya@NetBSD.org
Ian Lance Taylor	ian@NetBSD.org
Matt Thomas	matt@NetBSD.org
Jason Thorpe	thorpej@NetBSD.org
Christoph Toshok	toshok@NetBSD.org
Greg Troxel	gdt@NetBSD.org
Tsubai Masanari	tsubai@NetBSD.org
Izumi Tsutsui	tsutsui@NetBSD.org
UCHIYAMA Yasushi	uch@NetBSD.org
Masao Uebayashi	uebayasi@NetBSD.org
Shuichiro URATA	ur@NetBSD.org
Valeriy E. Ushakov	uwe@NetBSD.org
Todd Vierling	tv@NetBSD.org
Aymeric Vincent	aymeric@NetBSD.org
Paul Vixie	vixie@NetBSD.org
Mike M. Volokhov	mishka@NetBSD.org
Krister Walfridsson	kristerw@NetBSD.org
Lex Wennmacher	wennmach@NetBSD.org
Leo Weppelman	leo@NetBSD.org
Assar Westerlund	assar@NetBSD.org
Todd Whitesel	toddpw@NetBSD.org
Frank Wille	phx@NetBSD.org
Nathan Williams	nathanw@NetBSD.org
Rob Windsor	windsor@NetBSD.org
Dan Winship	danw@NetBSD.org
Jim Wise	jwise@NetBSD.org
Michael Wolfson	mbw@NetBSD.org
Colin Wood	ender@NetBSD.org
Steve Woodford	scw@NetBSD.org
YAMAMOTO Takashi	yamt@NetBSD.org
Yuji Yamano	yyamano@NetBSD.org
David Young	dyoung@NetBSD.org
Reinoud Zandijk	reinoud@NetBSD.org
S.P.Zeidler	spz@NetBSD.org
Maria Zevenhoven	maria7@NetBSD.org
Christos Zoulas	christos@NetBSD.org

Other contributors:

Dave Burgess	burgess@cynjut.infonet.net
Brian R. Gaeke	brg@dgate.org
Brad Grantham	grantham@tenon.com
Lawrence Kesteloot	kesteloo@cs.unc.edu
Waldi Ravens	waldi@moacs.indiv.nl.net

Legal Mumbo-Jumbo

All product names mentioned herein are trademarks or registered trademarks of their respective owners.

The following notices are required to satisfy the license terms of the software that we have mentioned in this document:

NetBSD is a registered trademark of The NetBSD Foundation, Inc.
This product includes software developed by the University of California, Berkeley and its contributors.
This product includes software developed by the NetBSD Foundation.
This product includes software developed by The NetBSD Foundation, Inc. and its contributors.
This product includes software developed for the NetBSD Project. See http://www.netbsd.org/ for information about NetBSD.
This product contains software developed by Ignatios Souvatzis for the NetBSD project.
This product contains software written by Ignatios Souvatzis and Michael L. Hitch for the NetBSD project.
This product includes software developed by Intel Corporation and its contributors.
This product contains software written by Michael L. Hitch for the NetBSD project.
This product includes software developed by Microsoft.
This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)
This product includes cryptographic software written by Eric Young (eay@mincom.oz.au)
This product includes software designed by William Allen Simpson.
This product includes software developed at Ludd, University of Lulea, Sweden and its contributors.
This product includes software developed at Ludd, University of Lulea.
This product includes software developed at the Information Technology Division, US Naval Research Laboratory.
This product includes software developed by Berkeley Software Design, Inc.
This product includes software developed by David Jones and Gordon Ross
This product includes software developed by Gordon W. Ross and Leo Weppelman.
This product includes software developed by Hellmuth Michaelis and Joerg Wunsch
This product includes software developed by Internet Research Institute, Inc.
This product includes software developed by Leo Weppelman and Waldi Ravens.
This product includes software developed by Mika Kortelainen
This product includes software developed by Aaron Brown and Harvard University.
This product includes software developed by Adam Ciarcinski for the NetBSD project.
This product includes software developed by Adam Glass and Charles M. Hannum.
This product includes software developed by Adam Glass.
This product includes software developed by Adrian Portelli for the NetBSD project.
This product includes software developed by Advanced Risc Machines Ltd.
This product includes software developed by Alex Zepeda, and Colin Wood for the NetBSD Projet.
This product includes software developed by Alex Zepeda.
This product includes software developed by Alistair G. Crooks.
This product includes software developed by Alistair G. Crooks. for the NetBSD project.
This product includes software developed by Allen Briggs.
This product includes software developed by Amancio Hasty and Roger Hardiman
This product includes software developed by Berkeley Software Design, Inc.
This product includes software developed by Berkeley Software Design, Inc.
This product includes software developed by Bill Paul.
This product includes software developed by Bodo Moeller. (If available, substitute umlauted o for oe)
This product includes software developed by Boris Popov.
This product includes software developed by Brad Pepers
This product includes software developed by Bradley A. Grantham.
This product includes software developed by Brini.
This product includes software developed by Bruce M. Simpson.
This product includes software developed by Causality Limited.
This product includes software developed by Charles D. Cranor and Seth Widoff.
This product includes software developed by Charles D. Cranor and Washington University.
This product includes software developed by Charles D. Cranor, Washington University, and the University of California, Berkeley and its contributors.

This product includes software developed by Charles D. Cranor, Washington University, the University of California, Berkeley and its contributors.
This product includes software developed by Charles D. Cranor.
This product includes software developed by Charles Hannum.
This product includes software developed by Charles M. Hannum, by the University of Vermont and State Agricultural College and Garrett A. Wollman, by William F. Jolitz, and by the University of California, Berkeley, Lawrence Berkeley Laboratory, and its contributors.
This product includes software developed by Charles M. Hannum.
This product includes software developed by Christian E. Hopps, Ezra Story, Kari Mettinen, Markus Wild, Lutz Vieweg and Michael Teske.
This product includes software developed by Christian E. Hopps.
This product includes software developed by Christian Limpach.
This product includes software developed by Christopher G. Demetriou for the NetBSD Project.
This product includes software developed by Christopher G. Demetriou.
This product includes software developed by Christos Zoulas.
This product includes software developed by Chuck Silvers.
This product includes software developed by Colin Wood for the NetBSD Project.
This product includes software developed by Colin Wood.
This product includes software developed by Daan Vreeken.
This product includes software developed by Daishi Kato
This product includes software developed by Dale Rahn.
This product includes software developed by Daniel Widenfalk and Michael L. Hitch.
This product includes software developed by Daniel Widenfalk for the NetBSD Project.
This product includes software developed by Darrin B. Jewell
This product includes software developed by David Miller.
This product includes software developed by Dean Huxley.
This product includes software developed by Eduardo Horvath.
This product includes software developed by Eric S. Hvozda.
This product includes software developed by Eric S. Raymond
This product includes software developed by Eric Young (eay@@mincom.oz.au)
This product includes software developed by Eric Young (eay@cryptsoft.com)
This product includes software developed by Eric Young (eay@mincom.oz.au)
This product includes software developed by K. Kobayashi.
This product includes software developed by K. Kobayashi and H. Shimokawa.
This product includes software developed by Ezra Story and by Kari Mettinen.
This product includes software developed by Ezra Story, by Kari Mettinen and by Bernd Ernesti.
This product includes software developed by Ezra Story, by Kari Mettinen, Michael Teske and by Bernd Ernesti.
This product includes software developed by Ezra Story, by Kari Mettinen, and Michael Teske.
This product includes software developed by Ezra Story.
This product includes software developed by Florian Stoehr.
This product includes software developed by Frank van der Linden for the NetBSD Project.
This product includes software developed by Gardner Buchanan.
This product includes software developed by Garrett D'Amore.
This product includes software developed by Gary Thomas.
This product includes software developed by Gordon Ross
This product includes software developed by Gordon W. Ross
This product includes software developed by HAYAKAWA Koichi.
This product includes software developed by Harvard University and its contributors.
This product includes software developed by Harvard University.
This product includes software developed by Henrik Vestergaard Draboel.
This product includes software developed by Herb Peyerl.
This product includes software developed by Hidetoshi Shimokawa.
This product includes software developed by Hubert Feyrer for the NetBSD Project.
This product includes software developed by Ian F. Darwin and others.
This product includes software developed by Ian W. Dall.
This product includes software developed by Ichiro FUKUHARA.
This product includes software developed by Ignatios Souvatzis for the NetBSD Project.
This product includes software developed by Internet Initiative Japan Inc.
This product includes software developed by James R. Maynard III.
This product includes software developed by Jared D. McNeill.
This product includes software developed by Jason L. Wright
This product includes software developed by Jason R. Thorpe for And Communications, http://www.and.com/
This product includes software developed by Jeremy C. Reed for the NetBSD project.
This product includes software developed by Joachim Koenig-Baltes.
This product includes software developed by Jochen Pohl for The NetBSD Project.
This product includes software developed by Joerg Wunsch

This product includes software developed by John Birrell.
This product includes software developed by John P. Wittkoski.
This product includes software developed by John Polstra.
This product includes software developed by Jonathan R. Stone for the NetBSD Project.
This product includes software developed by Jonathan Stone and Jason R. Thorpe for the NetBSD Project.
This product includes software developed by Jonathan Stone.
This product includes software developed by Jukka Marin.
This product includes software developed by Julian Highfield.
This product includes software developed by Kazuhisa Shimizu.
This product includes software developed by Kazuki Sakamoto.
This product includes software developed by Kenneth Stailey.
This product includes software developed by Kiyoshi Ikehara.
This product includes software developed by Klaus Burkert,by Bernd Ernesti, by Michael van Elst, and by the University of California, Berkeley and its contributors.
This product includes software developed by LAN Media Corporation and its contributors.
This product includes software developed by Leo Weppelman for the NetBSD Project.
This product includes software developed by Leo Weppelman.
This product includes software developed by Lloyd Parkes.
This product includes software developed by Lutz Vieweg.
This product includes software developed by MINOURA Makoto, Takuya Harakawa.
This product includes software developed by Manuel Bouyer.
This product includes software developed by Marc Horowitz.
This product includes software developed by Marcus Comstedt.
This product includes software developed by Mark Brinicombe for the NetBSD project.
This product includes software developed by Mark Brinicombe.
This product includes software developed by Mark Tinguely and Jim Lowe
This product includes software developed by Markus Wild.
This product includes software developed by Masanobu Saitoh.
This product includes software developed by Masaru Oki.
This product includes software developed by Mats O Jansson and Charles D. Cranor.
This product includes software developed by Mats O Jansson.
This product includes software developed by Matt DeBergalis
This product includes software developed by Matthew Fredette.
This product includes software developed by Matthias Pfaller.
This product includes software developed by Michael Graff for the NetBSD Project.
This product includes software developed by Michael Graff.
This product includes software developed by Michael L. Hitch.
This product includes software developed by Michael Shalayeff.
This product includes software developed by Michael Smith.
This product includes software developed by Mike Pritchard.
This product includes software developed by Mike Pritchard and contributors.
This product includes software developed by Minoura Makoto.
This product includes software developed by Niels Provos.
This product includes software developed by Niklas Hallqvist, Brandon Creighton and Job de Haas.
This product includes software developed by Niklas Hallqvist.
This product includes software developed by Onno van der Linden.
This product includes software developed by Paul Kranenburg.
This product includes software developed by Paul Mackerras.
This product includes software developed by Per Fogelstrom
This product includes software developed by Peter Galbavy.
This product includes software developed by Phase One, Inc.
This product includes software developed by Philip A. Nelson.
This product includes software developed by RiscBSD.
This product includes software developed by Roar Thronæs.
This product includes software developed by Rodney W. Grimes.
This product includes software developed by Roger Hardiman
This product includes software developed by Roland C. Dowdeswell.
This product includes software developed by Rolf Grossmann.
This product includes software developed by Ross Harvey for the NetBSD Project.
This product includes software developed by Ross Harvey.
This product includes software developed by Scott Bartram.
This product includes software developed by Scott Stevens.
This product includes software developed by Shingo WATANABE.
This product includes software developed by Softweyr LLC, the University of California, Berkeley, and its contributors.
This product includes software developed by Soren S. Jorvang.
This product includes software developed by Stephan Thesing.

This product includes software developed by Steve Woodford.
This product includes software developed by Steven M. Bellovin.
This product includes software developed by Takashi Hamada.
This product includes software developed by Takumi Nakamura.
This product includes software developed by Tatoku Ogaito for the NetBSD Project.
This product includes software developed by Terrence R. Lambert.
This product includes software developed by Tetsuya Isaki.
This product includes software developed by Thomas Gerner
This product includes software developed by Tobias Weingartner.
This product includes software developed by Todd C. Miller.
This product includes software developed by Tohru Nishimura and Reinoud Zandijk for the NetBSD Project.
This product includes software developed by Tohru Nishimura for the NetBSD Project.
This product includes software developed by Tohru Nishimura. for the NetBSD Project.
This product includes software developed by TooLs GmbH.
This product includes software developed by Trimble Navigation, Ltd.
This product includes software developed by WIDE Project and its contributors.
This product includes software developed by Waldi Ravens.
This product includes software developed by Wasabi Systems for Zembu Labs, Inc. http://www.zembu.com/
This product includes software developed by Winning Strategies, Inc.
This product includes software developed by Wolfgang Solfrank.
This product includes software developed by Yasushi Yamasaki.
This product includes software developed by Yen Yen Lim and North Dakota State University.
This product includes software developed by Zembu Labs, Inc.
This product includes software developed by the Alice Group.
This product includes software developed by the Center for Software Science at the University of Utah.
This product includes software developed by the Charles D. Cranor, Washington University, University of California, Berkeley and its contributors.
This product includes software developed by the Computer Systems Engineering Group at Lawrence Berkeley Laboratory.
This product includes software developed by the David Muir Sharnoff.
This product includes software developed by the Harvard University and its contributors.
This product includes software developed by the Network Research Group at Lawrence Berkeley Laboratory.
This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)
This product includes software developed by the PocketBSD project and its contributors.
This product includes software developed by the RiscBSD kernel team
This product includes software developed by the RiscBSD team.
This product includes software developed by the SMCC Technology Development Group at Sun Microsystems, Inc.
This product includes software developed by the University of California, Berkeley and its contributors, as well as the Trustees of Columbia University.
This product includes software developed by the University of California, Lawrence Berkeley Laboratory and its contributors.
This product includes software developed by the University of California, Lawrence Berkeley Laboratory.
This product includes software developed by the University of Illinois at Urbana and their contributors.
This product includes software developed by the Urbana-Champaign Independent Media Center.
This product includes software developed by the University of Vermont and State Agricultural College and Garrett A. Wollman.
This product includes software developed by the University of Vermont and State Agricultural College and Garrett A. Wollman, by William F. Jolitz, and by the University of California, Berkeley, Lawrence Berkeley Laboratory, and its contributors.
This product includes software developed for the FreeBSD project
This product includes software developed for the NetBSD Project by Bernd Ernesti.
This product includes software developed for the NetBSD Project by Christopher G. Demetriou.
This product includes software developed for the NetBSD Project by Christos Zoulas
This product includes software developed for the NetBSD Project by Emmanuel Dreyfus.
This product includes software developed for the NetBSD Project by Frank van der Linden
This product includes software developed for the NetBSD Project by Ignatios Souvatzis.
This product includes software developed for the NetBSD Project by Jason R. Thorpe.
This product includes software developed for the NetBSD Project by John M. Vinopal.
This product includes software developed by Kyma Systems.
This product includes software developed for the NetBSD Project by Kyma Systems LLC.
This product includes software developed for the NetBSD Project by Matthias Drochner.
This product includes software developed for the NetBSD Project by Michael L. Hitch.
This product includes software developed for the NetBSD Project by Perry E. Metzger.
This product includes software developed for the NetBSD Project by Scott Bartram and Frank van der Linden
This product includes software developed for the NetBSD Project by Allegro Networks, Inc., and Wasabi Systems, Inc.
This product includes software developed for the NetBSD Project by Genetec Corporation.
This product includes software developed for the NetBSD Project by Jonathan Stone.
This product includes software developed for the NetBSD Project by Piermont Information Systems Inc.
This product includes software developed for the NetBSD Project by SUNET, Swedish University Computer Network.
This product includes software developed for the NetBSD Project by Shigeyuki Fukushima.

This product includes software developed for the NetBSD Project by Wasabi Systems, Inc.
This product includes software developed under OpenBSD by Per Fogelstrom Opsycon AB for RTMX Inc, North Carolina, USA.
This product includes software developed under OpenBSD by Per Fogelstrom.
This software is a component of "386BSD" developed by William F. Jolitz, TeleMuse.
This software was developed by Holger Veit and Brian Moore for use with "386BSD" and similar operating systems. "Similar operating systems" includes mainly non-profit oriented systems for research and education, including but not restricted to "NetBSD", "FreeBSD", "Mach" (by CMU).
This software includes software developed by the Computer Systems Laboratory at the University of Utah.
This product includes software developed by Computing Services at Carnegie Mellon University (http://www.cmu.edu/computing/).
This product includes software developed by Marshall M. Midden.
This product includes software developed or owned by Caldera International, Inc.
The Institute of Electrical and Electronics Engineers and The Open Group, have given us permission to reprint portions of their documentation.

In the following statement, the phrase ‘‘this text’’ refers to portions of the system documentation.

Portions of this text are reprinted and reproduced in electronic form in NetBSD, from IEEE Std 1003.1, 2004 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2004 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between these versions and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document.

The original Standard can be obtained online at http://www.opengroup.org/unix/online.html.

This notice shall appear on any product containing this material

This product includes software developed by Allen K. Briggs.
This product includes software developed by Apple Computer, Inc.
This product includes software developed by Bill Studenmund.
This product includes software developed by Bradley A. Grantham.
This product includes software developed by Carnegie-Mellon University.
This product includes software developed by Chris P. Cputo.
This product includes software developed by Colin Wood.
This product includes software developed by David Huang.
This product includes software developed by Internet Research Institute, Inc.
This product includes software developed by John P. Wittkoski.
This product includes software developed by Open Software Foundation, Inc.
This product includes software developed by Per Fogelstrom, Opsycon AB and RTMX Inc, USA.
This product includes software developed by Takashi Hamada.
This product includes software developed by Tsubai Masanari.
This product includes software developed by University of Utah.
This product includes software developed by Wolfgang Solfrank.

The End