NAME

INSTALL — Installation procedure for NetBSD/sparc.

CONTENTS

DESCRIPTION

About this Document

This document describes the installation procedure for NetBSD 10.2 on the *sparc* 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 10.2 on a machine of the sparc architecture.

- Fetch files needed to install NetBSD. Fetch a CD image, `NetBSD-10.2-sparc.iso`. You need the CD to boot your system. Alternatively, you may netboot the installation kernel. This process is covered below, in detail.
- The actual binary distribution is in the `sparc/binary/sets/` directory. When you boot the install 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.
- Make sure your sparc's CD-ROM drive is bootable. Burn the CD.

  The media you just prepared will be used to boot the installation kernel, which contains all the tools required to install NetBSD.
- You will need to get to the OpenBoot PROM "**`ok`**" prompt. After your system first powers on and displays some initial information, press the `STOP-A` keys, or send a BREAK if you're on a serial console. At the "**`ok`**" prompt, type the command to boot your system into NetBSD. The command to boot from CD is one of the following commands (depending on your model): "**`b sd(,30,)`**", "**`boot sd(,30,)`**", or "**`boot cdrom`**".
- For third-party programs which are not part of the base NetBSD distribution, you will want to explore the **`pkgsrc`** package management system, which contains thousands of third-party software applications.

What is NetBSD?

The NetBSD Operating System is a fully functional open-source 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 many different different system architectures (ports) across a variety of distinct CPU families, and is being ported to more. The NetBSD 10.2 release contains complete binary releases for most of these system architectures, with preliminary support for the others included in source form. For more information please visit **https://www.NetBSD.org/**.

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, NetBSD would not exist.

Changes Between The NetBSD 9 and 10 Releases

The NetBSD 10.2 release provides many significant changes, including support for many new devices, hundreds of bug fixes, new and updated kernel subsystems, and numerous userland enhancements. The result of these improvements is a stable operating system fit for production use that rivals most commercially available systems.

One important new feature in this release is the support for extended attributes and access control lists on FFS file systems.

For new installations the installer will by default disable these features, so the file system is compatible with older NetBSD releases (before 10), and allow other operating systems to mount this file systems at least in read-only mode.

If you want a new installed file system to support extended attributes, change the file system type from "FFSv2" to "FFSv2ea" in the partitioning menu. You can also convert file systems later, using the `fsck_ffs`(8) utility. More details are available in *this guide:*

https://wiki.netbsd.org/tutorials/acls_and_extended_attributes_on_ffs.

If you are upgrading from a version of NetBSD -current please also check the **Compatibility Issues With Previous NetBSD Releases**. It is impossible to completely summarize the massive development that went into the NetBSD 10.2 release. The complete list of changes can be found in

CHANGES:
https://cdn.NetBSD.org/pub/NetBSD/NetBSD-10.2/CHANGES
CHANGES-10.1:
https://cdn.NetBSD.org/pub/NetBSD/NetBSD-10.2/CHANGES-10.1
CHANGES-10.2:
https://cdn.NetBSD.org/pub/NetBSD/NetBSD-10.2/CHANGES-10.2

which are also present in the top level directory of the NetBSD 10.2 release tree.

Features to be removed in a later release

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

- `groff`(1). Man pages are now handled with `mandoc`(1), and `groff`(1) can still be found in pkgsrc as `textproc/groff`.
- `pf`(4). This packet filter is obsolete and unmaintained in NetBSD. It will be eventually removed due to possible long-standing security issues and lack of multiprocessor support. New installations should use `npf`(7).

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: **https://www.NetBSD.org/foundation/**

Sources of NetBSD

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

NetBSD 10.2 Release Contents

The root directory of the NetBSD 10.2 release is organized as follows:

`.../NetBSD-10.2/`

`CHANGES`	Changes between the 9.0 and 10.0 releases.
`CHANGES-10.0`	Changes between the initial 10.0 branch and final release of 10.0.
`CHANGES-10.1`	Changes between the 10.0 and the 10.1 release.
`CHANGES-10.2`	Changes between the 10.1 and the 10.2 release.
`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.

`images/` Images (ISO 9660 or USB) for installing NetBSD. Depending on your system, these may be bootable.

`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 10.2 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.

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.

src This set contains all of the base NetBSD 10.2 sources which are not in **gnusrc**, **sharesrc**, or **syssrc**.

syssrc This set contains the sources to the NetBSD 10.2 kernel for all architectures as well as the `config`(1) utility.

xsrc This set contains the sources to the X Window System.

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:

`MD5` MD5 digests in the format produced by the command:
cksum -a MD5 *file*.

`SHA512` SHA512 digests in the format produced by the command:
cksum -a SHA512 *file*.

The SHA512 digest is safer, but MD5 checksums are provided so that a wider range of operating systems can check the integrity of the release files.

NetBSD/sparc subdirectory structure

The sparc-specific portion of the NetBSD 10.2 release is found in the `sparc` subdirectory of the distribution: `.../NetBSD-10.2/sparc/`. 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_SCSI3.gz`
: A kernel with SCSI ID 3 explicitly mapped to `sd0`.

`netbsd-KRUPS.gz`
: A kernel for JavaStation-NC (Krups) machines.

`netbsd-MRCOFFEE.gz`
: A kernel for JavaStation-1 (Mr. Coffee) machines using Open Boot Prom firmware.

`netbsd-MRCOFFEE_OFW.gz`
: A kernel for JavaStation-1 (Mr. Coffee) machines using OpenFirmware.

`netbsd-GENERIC_SUN4U.gz`
: A kernel which supports sparc64 hardware in 32-bit mode only. See the sparc64 INSTALL notes for help installing this kernel on your sparc64 hardware.

`sets/` sparc binary distribution sets; see below.

`installation/`

`bootfs/` Netbootable RAM disk kernel and the file system in the RAM disk; see below.

`miniroot/` sparc miniroot file system image; see below.

`netboot/` Netboot bootloader and root filesystem with install tools; see below.

`tape/` sparc boot and installation tapes; see below.

Binary distribution sets

The NetBSD sparc binary distribution sets contain the binaries which comprise the NetBSD 10.2 release for sparc. The binary distribution sets can be found in the `sparc/binary/sets` subdirectory of the NetBSD 10.2 distribution tree, and are as follows:

base The NetBSD 10.2 sparc **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.

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.

debug This distribution set contains debug information for all base system utilities. It is useful when reporting issues with binaries or during development. This set is huge, if the target disk is small, do not install it.

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.

games This set includes the games and their manual pages.

kern-GENERIC
This set contains a NetBSD/sparc 10.2 `GENERIC` kernel named `/netbsd`. You *must* install this distribution set.

kern-GENERIC.MP
This set contains a NetBSD/sparc 10.2 `GENERIC.MP` kernel, which will use multiple processors (if present), named `/netbsd`.

kern-GENERIC_SCSI3
This set contains a NetBSD/sparc 10.2 `GENERIC_SCSI3` kernel, which will set SCSI ID 3 to `sd0`, named `/netbsd`.

kern-GENERIC_SUN4U
This set contains a NetBSD/sparc 10.2 `GENERIC_SUN4U` kernel, which supports only UltraSPARC systems in 32 bit mode, named `/netbsd`.

kern-KRUPS
This set contains a NetBSD/sparc 10.2 `KRUPS` kernel, which supports the JavaStation-NC, named `/netbsd`.

kern-MRCOFFEE
This set contains a NetBSD/sparc 10.2 `MRCOFFE` kernel, which which supports the OpenBOOT PROM 2 JavaStation-1, named `/netbsd`.

kern-MRCOFFEE_OFW
This set contains a NetBSD/sparc 10.2 `MRCOFFE_OFW` kernel, which which supports the OpenFirmware JavaStation-1, named `/netbsd`.

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.

misc This set includes the system dictionaries, the typesettable document set, and other files from `/usr/share`.

modules This set includes kernel modules to add functionality to a running system.

rescue This set includes the statically linked emergency recovery binaries installed in `/rescue`.

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

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.

xcomp The extra libraries and include files needed to compile X source code.

xdebug This distribution set contains debug information for all X11 binaries. It is useful when reporting issues with these binaries or during development. This set is huge, if the target disk is small, do not install it.

xfont Fonts needed by the X server and by X clients.

xetc Configuration files for X which could be locally modified.

xserver The X server. This includes Xsun, Xsun24, XsunMono and Xvfb servers with man pages.

The sparc 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: The distribution cannot be extracted using the SunOS **tar** program

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

NetBSD/sparc System Requirements and Supported Devices

The minimum configuration requires 4 MB of RAM and 140 MB of disk space. To install the entire system requires considerably more space, and more RAM to achieve reasonable performance. NetBSD with 4 MB of RAM feels like Solaris with 4 MB of RAM — slow. Note that until you have around 16 MB of RAM, getting more RAM is more important than getting a faster CPU.

Supported machines

- sun4
  - 4/100
  - 4/200
  - 4/300
  - 4/400
- sun4c
  - SPARCstation/server 1
  - SPARCstation/server 1+
  - SPARCstation/server 2
  - IPC
  - ELC
  - IPX
  - SLC
- sun4c clones
  - Aries Parrot II
  - Opus/5000
  - Opus/5250
  - FORCE Sparc CPU1e/4
  - FORCE Sparc CPU2e/4

  (and possibly more)

  Note: NetBSD doesn't support the VME interface on the FORCE models
- sun4m
  - 4/600
  - SPARC Classic
  - SPARC LX
  - SPARCstation 4
  - SPARCstation/server 5
  - SPARCstation/server 10
  - SPARCstation/server 20
  - SPARCstation Voyager

- JavaStation-1 (Mr. Coffee)
- JavaStation-NC (Krups)

Note: All known SuperSPARC, hyperSPARC, microSPARC, and TurboSPARC CPUs are supported

- sun4m clones
  - Axil 245
  - Axil 311
  - Axil 320
  - Cycle C C Cycle 5-IP
  - DTKstation/Classic+
  - RDI PowerLite-85
  - RDI PowerLite-110
  - RDI BriteLite
  - Ross sparc 20
  - Transtec SS5/170
  - Tatung microCOMPstation 5
  - Twinhead TWINstation 5G
  - TWS SuperCOMPstation-20S
  - Tadpole SPARCbook 3 series

  (and possibly more)

Machines not supported by NetBSD/sparc

- sun4d
  - SPARCcenter 1000
  - SPARCcenter 2000
- sun4u and other UltraSPARC based machines in 64-bit mode. They are supported in 32-bit mode with the NetBSD/sparc `GENERIC_SUN4U` kernel.

  UltraSPARC based machines in 32-bit or 64-bit mode are supported by the NetBSD/sparc64 port. Refer to

  http://www.NetBSD.org/ports/sparc64/index.html
- Solbourne machines

Supported devices

- sun4m mbus CPU modules; Sun 4/600, SPARCstation/server 10, SPARCstation/server 20, and clones
  - *cpu0*; All single-CPU systems are supported
  - *cpuN*; All hardware compatible dual-, tri- and quad-CPU SuperSPARC-I, SuperSPARC-II, and hyperSPARC systems are supported with the SMP kernel (`GENERIC.MP`)

    http://www.NetBSD.org/ports/sparc/faq.html#smp-cpus
- sun4c and sun4m video; Sbus card, on-board, or optional
  - *bwtwo*; 1 bpp framebuffer.
    On-board and Sbus card (MG1 and MG2)
  - *cgthree*; 8 bpp framebuffer.
    On-board and Sbus card

  - *cgsix*; 8 bpp accelerated framebuffer.
    On-board and Sbus card (GX, GX+, TGX, and TGX+)
  - *cgfourteen*; SX 24 bpp accelerated framebuffer.
    On-board SPARCstation 10SX and SPARCstation 20, requires optional 4 or 8 MB VSIMM,
    limited to *cgthree* emulation
  - *tcx*; S24 accelerated color framebuffer.
    On-board SPARCstation 4 (8 bpp), JavaStation-1 (24 bpp), and optional on SPARCstation 5 (24 bpp),
    limited to *cgthree* emulation
  - *pnozz*; Weitek p9100 24 bpp accelerated framebuffer.
    On-board Tadpole SPARCbook 3GS and 3GX,
    limited to *cgthree* emulation
  - *zx*; ZX and TZX 24 bpp accelerated framebuffer.
    Sbus card (leo)
    limited to text console (no X)
  - *igsfb*; 24 bpp IGS C1682 accelerated framebuffer.
    On-board JavaStation-NC
    limited to text console (no X)
- sun4 video; VME card, on-board, or P4 (supported only on 4/100 and 4/300 series machines)
  - *bwtwo*; 1 bpp framebuffer.
    On-board and P4
  - *cgtwo*; 8 bpp framebuffer.
    VME card
  - *cgfour*; 8 bpp framebuffer with overlay.
    P4
  - *cgsix*; LEGO 8 bpp accelerated framebuffer.
    P4
  - *cgeight*; TC 24 bpp framebuffer with overlay.
    P4 (not well tested)
- Network interfaces
  - *be*; 100 Mbps half-duplex "Big Mac" ethernet.
    Sbus card
  - *en*; Efficient Networks, Inc. ATM interface.
    Sbus card
  - *hme*; 100 Mbps full-duplex "Happy Meal" ethernet.
    On-board (JavaStation-NC) and Sbus card
    This hardware is not supported on the sun4c platform
  - *ie*; 10 Mbps Intel 82586 ethernet.
    On-board (4/100 and 4/200, may not be working properly on 4/400) and VME card
  - *le*; 10 Mbps AMD 9790 Lance ethernet.
    On-board (4/300, sun4c, and sun4m) and Sbus card

- *qec*; One 100 Mbps be interface and four 10 Mbps Mace ethernet interfaces.
  Sbus card
- *qfe*; four 100 Mbps full-duplex "Happy Meal" ethernet interfaces.
  Sbus card
- misc: various ethernet interfaces are found on multi-function Sbus cards (e.g. along with SCSI), the ethernet interfaces work even if the other functions do not

• SCSI interfaces

- *esp*; 5 MB/s and 10 MB/s (Fast) NCR 53C9x SCSI controller, and 20 MB/s (Fast+Wide) FAS336 controller (known as 'fas' on Solaris).
  On-board (sun4c, sun4m, 4/300, Ultra 1e, and Ultra 2) and Sbus cards
- *isp*; 10 MB/s (Fast) and 20 MB/s (Fast+Wide) Qlogic ISP 10x0 family.
  Sbus card
- *si*; 5 MB/s NCR 5380 SCSI controller.
  VME card (Interrupt driven and polled DMA)
- *sw*; 5 MB/s NCR 5380 "SCSI Weird" controller.
  Onboard 4/110 (polled DMA only)
- misc: SCSI functionality is sometimes found on multi-function Sbus cards (e.g. along with ethernet), the SCSI *esp* and *isp* portions of those cards should work, even if the other functions do not.

  Note: Non-Sun SCSI CD-ROMs may have trouble booting any OS, NetBSD or otherwise. Your CD-ROM must support 512 byte blocks for the OpenBoot PROM to load a file. Most newer CD-ROM drives support this automatically, and many older drives allow you to set it using a jumper. See
  http://www.ebsinc.com/solaris/suncdrom.html
  for information on bootable CD-ROMs

• VME disk interfaces
- *xd*; Xylogics 7053 VME/SMD disk controller
- *xy*; Xylogics 450/451 VME disk controller

  Note: VME/IPI disks are not supported

• Floppy drives

- *fd*; floppy disk drive.
  On-board sun4c and sun4m
- *sd*; SCSI floppy drive, acts as a SCSI drive to kernel.
  found with some SPARCbook 3 machines

• PCMCIA/Cardbus devices
- *nell*; PCMCIA interface
  On-board (SPARCstation Voyager) and Sbus card.
  All devices listed in the Machine-independent PCMCIA drivers page should work, but very few have been tested with NetBSD/sparc.
  http://www.NetBSD.org/support/hardware/pcmcia.html

  Note: See the FAQ for more information.
  http://www.NetBSD.org/ports/sparc/faq.html#nell

- Serial interfaces
  - *zs*; Zilog 8530 serial ports, which can be used as the system console
    On-board sun4, sun4c, and sun4m
  - *com*; Zilog 85C30 serial ports (ns16550a)
    On-board Tadpole SPARCbook 3GS and 3GX (internal 14.4 kbps modem, use `/dev/ttyC0`) JavaStation-1, and JavaStation-NC
- Input devices
  - Sun keyboard and mouse
  - SPARCbook 3 keyboard and joystick
  - Third-party keyboards, mice, and trackballs
  - PS/2 keyboards and mice on JavaStation models
- Audio interfaces
  - *audioamd*; AMD 7930-based 8000 Hz, 8 bit, mono, mu-law.
    On-board sun4c and SPARC Classic
  - *audiocs*; CS4231-based 44 KHz, 16 bit, stereo, PCM.
    On-board SPARCStation 5, JavaStation-1, JavaStation-NC, and optional on SPARCStation 4.
- Misc cards
  - *mtty / mbpp*; Magma Serial/Parallel ports.
    Sbus card
  - *xbox*;
    Sun Sbus expander device
- Parallel ports
  - *bpp*; Bi-directional parallel port.
    On-board and Sbus card

Unsupported devices

- DBRIe-based audio and ISDN.
  On-board some sun4m machines (such as SPARCstation 10, SPARCstation 20, SPARC LX, SPARCstation Voyager, and Tadpole SPARCbook 3)
- *sc*; Sun SCSI.
  VME card
- *cgtwelve*; I.e. GS, 24 bpp framebuffer.
  Sbus card
- PCMCIA interface
  On-board Tadpole SPARCbook 3GS and 3GX

Getting the NetBSD System on to Useful Media

Installation is supported from several media types, including:

- CD-ROM / DVD / USB stick
- FTP
- Remote NFS partition

- Tape
- Existing NetBSD 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 / USB stick
Find out where the distribution set files are on the CD-ROM, DVD or USB stick. Likely locations are `binary/sets` and `sparc/binary/sets`. (You only need to know this if you are mixing installer and installation media from different versions - the installer will know the proper default location for the sets it comes with).

Proceed to the instructions on installation.

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.

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_sets
```

where *tape_device* is the name of the tape device that represents the tape drive you're using. This might be /dev/rst0, or something similar, but it will vary from system to system. In the above example, *dist_sets* is a list of filenames corresponding to the distribution sets that you wish to place on the tape. For instance, to put the **kern-GENERIC, base,** and **etc** distributions on tape (the absolute minimum required for installation), you would do the following:

```
# cd .../NetBSD-10.2
# cd sparc/binary
# tar -cf tape_device kern-GENERIC.tgz
base.tgz etc.tgz
```

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

Configuring your PROM

Before you start, you should configure your PROM. There are three categories of PROM: "sunmon" (sometimes called Restricted Prompt, sun monitor, or old command mode), OpenBoot PROM 1, and OpenBoot PROM 2. The sun4 machines only have sunmon. Some of the early sun4c models default to sunmon but have OpenBoot PROM 1. The later sun4c models and all sun4m models have OpenBoot PROM 2.

First, you need to stop your system from automatically booting when powered on. Pressing the STOP key (sometimes called the L1 key, found on the left side of your keyboard) and the a key will halt your system and give you a PROM prompt. If you are using a Tadpole SPARCbook, you press the Pause and a keys. If you are using a serial console, send a "BREAK" signal from your terminal (the method of sending "BREAK" varies from terminal to terminal).

If the ethernet address of your sparc system is ff:ff:ff:ff:ff:ff, then your NVRAM battery is dead and you will have trouble using ethernet (among other problems). Read the *Sun NVRAM/Hostid FAQ:*
https://web.archive.org/web/20150611091616/https://www.squirrel.com/squirrel/sun-nvram-hostid.faq.html.

If you have a valid ethernet address and you plan to netboot, write down your system's ethernet address.

Next, you should set your system to always use the OpenBoot PROM (sometimes called "new command mode") if it defaults to sunmon. The NetBSD kernel relies on some of the functionality provided by the OpenBoot PROM. If your machine gives you a '>' prompt instead of an 'ok' prompt, type:

```
> n
ok setenv sunmon-compat? false
ok
```

Next, if you are using any security features of OpenBoot PROM, you should turn them off — NetBSD can't deal well with this.

```
ok setenv security-mode none
```

If you are using a serial console, the NetBSD/sparc installer defaults to using 9600 bps, 8N1 settings. You may want to configure your system and serial terminal like this prior to booting the installer. Additionally, a new installation of NetBSD/sparc will default to these settings as well.

The OpenBoot PROM 1 machines (SPARCstation/server 1, SPARCstation/server 1+, IPC, and SLC) have an odd SCSI quirk you should be aware of. There are three SCSI addressing schemes used by your system: SCSI target ID (set by physical jumpers on the device), PROM 'unit' number (set by OpenBoot PROM 1, based on its SCSI target ID), and the name you reference within an operating system (set by the kernel, based on the PROM 'unit' number).

Sun shipped these systems with the internal drives set to SCSI target IDs 3 and 1. The default value of the OpenBoot PROM variable '`sd-targets`' is '`31204567.`' This variable maps how the OpenBoot PROM 1 assigns 'unit' numbers based on the SCSI target ID. Thus the device at SCSI target ID 3 is considered 'unit' 0, and the SCSI device at target ID 0 is 'unit' 3. When you type "**`boot scsi(0,0,0)`**", the OpenBoot PROM will boot from 'unit' 0 (which is SCSI target ID 3, the internal hard drive). The SunOS kernel is hard-wired to map `sd0` to SCSI target 3, and `sd3` to SCSI target 0.

PROM Unit#	*SCSI Target*	*SunOS name*
0	3	sd0
1	1	sd1
2	2	sd2
3	0	sd3
4	4	st0
5	5	st1
6	6	cdrom

The NetBSD/sparc `GENERIC` kernel does not wire things down as does SunOS. It names the disks in the order that the SCSI targets are probed (01234567). If you only have one disk, it is always `sd0` regardless of its SCSI target ID or its PROM 'unit' number, and there are no problems. If you have two disks, one at SCSI ID 2 ('unit' 2) and SCSI ID 3 ('unit' 0), then they are recognized as `sd0` and `sd1` respectively. This can be a problem if you are not aware of it, particularly when creating an fstab.

There are two approaches to fixing this problem: changing the mapping that OpenBoot PROM 1 does, and changing the NetBSD kernel configuration. To get OpenBoot PROM 1 to number the SCSI 'unit' numbers the same as the SCSI target IDs, you need to run this command:

ok **`setenv sd-targets 01234567`**

This may, however, cause problems if you were to later attempt to use SunOS on this machine or if you reset the OpenBoot PROM variables.

The other approach is to use a NetBSD kernel that matches the PROM's odd target mapping by treating 'unit' 0 (i.e. SCSI target ID 3) as `sd0` and 'unit' 3 (i.e. SCSI target ID 0) as `sd3`. The `GENERIC_SCSI3` kernel performs this target mapping, but the `GENERIC` and `INSTALL` kernels do not.

Note: This is also a concern when you start building your own customised kernels.

The machines with OpenBoot PROM 2 (SPARCstation/server 2, ELC, IPX, and all sun4m models) have a similar SCSI target mapping in the form of a **`devalias`** entry. That is, the device alias **`disk`** is shorthand for the disk at SCSI ID 3 on the internal SCSI controller. Normally, the **`disk`** device alias is what the PROM uses as the default boot device, i.e. in the absence of a *`device`* argument to the **`boot`** command. Note that there are also pre-configured device alias entries for **`disk0, disk1, disk2`** and **`disk3`**, which are in fact a one-to-one mapping to the SCSI targets **`0`** to **`3`** (all on the internal SCSI controller).

Again, it may be advantageous to use a fixed "SCSI target" to "NetBSD disk unit" mapping in your kernel configuration file (such as is done in the `GENERIC_SCSI3` kernel) to ensure that your disks remain showing up at the same NetBSD device unit numbers even if you add disks to your system at a later time.

Determining how to access your SCSI disk from the PROM

sunmon and OpenBoot PROM 1 use an archaic `sd(c,u,p)` syntax to address SCSI devices. OpenBoot PROM 2 uses a more intuitive syntax using device aliases.

To calculate the parameters for sunmon and OpenBoot PROM 1:

c specifies the SCSI controller number (first is 0, second is 1, ...)

u the hexadecimal number obtained from evaluating the expression *(8 ∗ TARGET) + LUN*

p the partition number from which to boot ('0' = 'a', '1' = 'b', etc.)

Therefore, to boot from the swap partition on the internal hard drive (first SCSI bus, target 0, lun 0, partition 1), one would use:

ok **boot sd(0,0,1)**

To boot from a CD-ROM (first SCSI bus, target 6, lun 0, partition dynamically determined), one would use:

ok **boot sd(0,30,)**

And, to boot from a kernel named `netbsd-GENERIC` on the fourth partition ('d', often the `/usr` partition) on an external hard drive (first SCSI bus, target 2, lun 0, partition 3), one would use:

ok **boot sd(0,10,3)netbsd-GENERIC**

Now, for OpenBoot PROM 2, SCSI devices are specified by an OpenBOOT **devalias** which provides simple mnemonics for the full path to the device. Type **devalias** in OpenBoot PROM 2 to get a list of all of the available aliases. Just the alias and partition are necessary when booting.

Therefore, to boot from the swap partition on the internal hard drive (OpenBoot PROM 2 assumes the internal hard drive is at target 3), one would use:

ok **boot disk:b**

To boot from a CD-ROM (OpenBoot PROM 2 assumes the CD-ROM is at target 6), one would use:

ok **boot cdrom**

And, to boot from a kernel named `netbsd-GENERIC` on the fourth partition ('d', often the `/usr` partition) on an external hard drive (target 2, partition 3), one would use:

ok **boot disk2:d netbsd-GENERIC**

The full device path specifier for OpenBoot PROM 2 depends on how OpenBoot PROM 2 recognizes your SCSI controller. Typically, one would use something like: `/sbus/esp/sd@t,p` where t is the SCSI target and p is the partition number.

Determining how to boot from an SBUS card

Some SBUS cards have firmware that lets you use them as a boot device. These cards do not automatically create a **devalias** entry, so you must traverse the device tree to figure out what the OpenBoot PROM calls your card. You will be using OpenBoot PROM commands at the 'ok' prompt. First 'cd' to the top of the device tree and list the nodes there. The following is the procedure to boot from an HME card in a SPARC Classic.

```
ok cd /
ok ls
ffd3b790 TI,TMS390S10@0,f8fffffc
ffd2d254 virtual-memory@0,0
ffd2d198 memory@0,0
ffd2b65c obio
```

```
ffd2b310 iommu@0,10000000
ffd2b2a0 openprom
ffd24af0 aliases
ffd24abc options
ffd24a88 packages
```

Usually, you can simply type in the name before the at (@) sign and the OpenBoot PROM will fill in the rest.

```
ok cd iommu
ok ls
ffd2b454 sbus@0,10001000
ok cd sbus
ok ls
ffd467e8 cgthree@3,0
ffd42a1c SUNW,hme@1,8c00000
ffd4297c le@0,c00000
ffd40d28 ledma@4,8400010
ffd40c9c SUNW,bpp@4,c800000
ffd40c00 audio@4,1300000
ffd3dc68 espdma@4,8400000
ok cd SUNW,hme
ok ls
ok pwd
/iommu@0,10000000/sbus@0,10001000/SUNW,hme@1,8c00000
```

OK, now we know the path to the HME device in this example. Now, we need to determine if it's capable of booting. If it is, it will have the **word** 'open'.

```
ok words
reset          seek           load           open           close
watch-net      selftest       obp-selftest   write          read
preamble-32    enable-link-pulse             disable-link-pulse
force-speeds   reset-transceiver             use-bit-bang-mode
use-frame-mode                dump-phys      transfer-speed=10
transfer-speed=100            mii-write      mii-read
create-rev-id
```

Great! Also, in case you're interested in further details about your hardware, you can use the '.attributes' command.

```
ok .attributes
hm-rev                   00000022
version                  1.18
model                    SUNW,501-2919
device_type              network
intr                     00000037  00000000
interrupts               00000004
address-bits             00000030
max-frame-size           00004000
reg                      00000001  08c00000  00000108
                         00000001  08c02000  00002000
                         00000001  08c04000  00002000
                         00000001  08c06000  00002000
                         00000001  08c07000  00000020
name                     SUNW,hme
ok
```

So, when it's time to type in a boot command, use the shortened version of the **pwd** command. In this example, you'd type:

ok **boot /iommu/sbus/SUNW,hme**

And when the kernel is done booting, it may not automatically use your card as the root device -- you may need to type in the NetBSD/sparc name for that device:

```
root on sd0a dumps on sd0b
no file system for sd0 (dev 0x700)
cannot mount root, error = 79
root device (default sd0a): ?
use one of: fd0[a-h] le0 le1 hme0 sd0[a-h] halt
root device (default sd0a): hme0
dump device: hme0
file system (default generic): nfs
root on hme0
```

Deciding on partition sizes

If you're installing NetBSD/sparc for the first time it's a good idea to look at the partition sizes of disk you intend installing NetBSD/sparc on.

A minimal installation of NetBSD requires about 140 MB. A full installation requires considerably more. A good initial size for the swap partition is twice the amount of physical memory in your machine (unlike SunOS 4.x, there are no restrictions on the size of the swap partition that would render part of your memory unusable).

Note that there are limitations on the size of the root partition for various models.

On sun4 machines, the NetBSD sparc boot loader can only boot from RAID partitions that start at the beginning of the disk.

On sun4 and early PROM version sun4c machines, the PROM can only boot from the first 1Gb of the disk.

On later PROM version sun4c and early PROM version sun4m machines, the PROM can only boot from the first 2Gb of the disk.

On later PROM version sun4m machines, the PROM can only boot from the first 4Gb of the disk.

Configuration of network interfaces

Some network devices (i.e. the built-in *le* interface on sun4m machines) allow a choice between operating on a UTP or a AUI port once the NetBSD kernel is running. The *le* driver supports automatic detection of the port which is actually connected to the wire. Additionally, some of the Fast ethernet devices (such as *be*, *hme*, *qec*, and *qfe*) support selection of various speeds and options. The default is to attempt to automatically detect the speed.

If automatic detection is not available or not working properly in your environment, you may have to specify the type connection using the **media** parameter of `ifconfig`(8). During installation, you'll get the opportunity to specify the appropriate medium. Use `10base5` or `AUI` to select the AUI connector, or `10baseT` or `UTP` to select the UTP connector. Fast ethernet interfaces default to `auto`, which usually does not detect properly and runs at '10BaseT' speed. The options are `10baseT`, `10baseTX`, and `auto`. The *hme* and *qfe* interfaces also allow `10baseT-FDX` and `100baseT-FDX`.

Installing the NetBSD System

Installing NetBSD is a relatively complex process, but if you have this document in hand it shouldn't be too much trouble.

There are several ways to install NetBSD onto your system. The easiest way is to use the `miniroot.fs` image copied to your local disk's swap partition or a scratch drive. If your Sparc is hooked up in a network, you may configure another UNIX-like machine as a netboot server for your Sparc.

For more information on the commands and variables available in the OpenBoot PROM (present in all sun4c and sun4m machines), go to **http://docs.sun.com** and search for "openboot reference" (without quotes).

Installing NetBSD by placing a bootable file system on a partitioned hard drive

`installation/miniroot/miniroot.fs.gz` is a compressed, self-contained NetBSD file system holding all utilities necessary to install NetBSD onto a disk attached to your system. It is distributed as a compressed plain file you will transfer to a raw disk partition. You will then boot this installer using the appropriate PROM command. The simplest case is where you place the `miniroot.fs` file system on the swap partition of your disk. Alternatively, you can place it on any other unused partition on any disk (such as a Zip disk). Be forewarned that you will not be able to install NetBSD onto the partition that contains the `miniroot.fs` as this partition is needed during the install process.

Loading the file system onto a raw partition is straightforward. First, download `miniroot.fs` to your system. Next, reboot in single-user mode (i.e. **boot -s**) to ensure that your system will not be swapping. Finally, place the `miniroot.fs` on your partition of choice. On NetBSD or SunOS the commands are:

```
# gunzip miniroot.fs.gz
# dd if=miniroot.fs of=/dev/rsd0b bs=4k
```

Here, `/dev/rsd0b` is assumed to be your swap partition. If you decide to place `miniroot.fs` on a non-swap partition, it will overwrite all of the contents of that partition, but you will not need to reboot into single-user mode to write it.

After transferring the file system to disk, bring the system down by:

```
# halt
```

Then boot the installer by typing the appropriate command at the PROM prompt (this example is for the swap partition):

sunmon

> > **b sd(,,1)netbsd -s**

OpenBoot PROM 1

> ok **boot sd(,,1)netbsd -s**

OpenBoot PROM 2

> ok **boot disk:b netbsd -s**

The monitor boot command will load the NetBSD kernel contained in the file system image. After the initial probe messages you'll be asked to start the install or upgrade procedure. Proceed to the section **Running the installation scripts.** below.

Installing NetBSD by using a bootable CD-ROM

If you wish to burn a CD-R containing a bootable NetBSD installation, then you can either burn the prepared disk image in `images/NetBSD-10.2-sparc.iso` or create your own. If you wish to create your own, you should follow the directions on the NetBSD Bootable CD-ROM HOW-TO at:

http://www.NetBSD.org/docs/bootcd.html#sparcimage

Once you have downloaded `NetBSD-10.2-sparc.iso` or created your own disk image, then you need to burn it to a CD-R.

The CD-ROM attached to your Sparc must support 512 byte sectors to be bootable. All Sun brand CD-ROMs will work, as well as many other manufacturers. See the Sun CD-ROM FAQ: **http://saturn.tlug.org/suncdfaq/**

Sun sets their CD-ROM drives to SCSI ID 6. We recommend you do the same.

Boot the installer by typing the appropriate command at the PROM prompt:

sunmon

> **`b sd(,30,)`**

OpenBoot PROM 1

ok **`boot sd(,30,)`**

OpenBoot PROM 2

ok **`boot cdrom`**

After the initial probe messages you'll be asked to start the install or upgrade procedure. Proceed to the section **Running the installation scripts.** below.

Installing NetBSD by using a netboot setup

1. Introduction

   To netboot a sparc, you must configure one or more servers to provide information and files to your sparc (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:
   https://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 sparc systems except JavaStations with OpenFirmware PROM use a combination of RARP and BOOTP for the discovery stage. In the past, these systems used RARP and BOOTPARAMS. JavaStations booting from OpenFirmware use DHCP. TFTP is used in the bootstrap phase to download the bootstrap program, `boot.net`, which has been linked to a file name appropriate to the client's architecture and IP address as described in the TFTP section below. JavaStations with OpenFirmware need a special bootstrap program, `bootjs.net`, which is a version of boot.net in ELF format. It is not linked to the IP address. 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.

                #
                # Only needed for JavaStations with OpenFirmware!
                #
                filename                "bootjs.net";
```

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

                        #
                        # The host address of the NFS server. This is mandatory for
                        # NetBSD kernels even it's the same host as the DHCP server.
                        #
                        next-server server.test.net;
                }
        #you may paste another "host" entry here for additional
        #clients on this network
        }
```

Note that for JavaStations booting from OpenFirmware you do not need the

```
allow bootp;
```

statement, since this machines use real DHCP.

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. rarpd(8)

   Note: no **rarpd** is needed to boot OpenFirmware based JavaStations.

   Create an /etc/ethers file with the following line:

   ```
   CC:CC:CC:CC:CC:CC     client
   ```

   Add your client to the server's /etc/hosts file:

   ```
   192.168.1.10 client
   ```

   You will need to start the **rarpd**. 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/rarpd restart
   ```

4. 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 sparc machine to /tftpboot. Get boot.net from the installation/netboot directory of the distribution.

```
# cp boot.net /tftpboot
```

Now, you need to link boot.net to the filename that your sparc will look for. It will look for a filename composed of the machine's IP address (in hexadecimal) followed by the machine's architecture, separated by a period. For example, a sun4c machine which has been assigned IP address 192.168.1.10, will make a TFTP request for C0A8010A.SUN4C.

You can use bc(1) to help calculate the filename:

```
# bc
obase=16
192
C0
168
A8
1
1
10
A
quit
# cd /tftpboot
# ln -s boot.net C0A8010A.SUN4C
```

For OpenFirmware based JavaStations you use the file bootjs.net and do not create a symlink for the IP address. This machines do full DHCP autoconfiguration, so the bootstrap filename is provided by the DHCP server.

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
```

5. 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
   ```

6. NetBSD kernel and installation tools

   Now, if you place a kernel named netbsd in /export/client/root your client should boot the kernel. Use binary/kernel/netbsd-GENERIC.gz.

```
# gunzip netbsd-GENERIC.gz
# mv netbsd-GENERIC /export/client/root/netbsd
```

If you are netbooting the installer, copy the distribution files to the client's root directory and extract the tools from `installation/netboot/rootfs.tgz`.

```
# cp *.tgz /export/client/root
# cd /export/client/root
# tar -xpzf rootfs.tgz
```

7. 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 *for details:* **https://www.NetBSD.org/docs/network/netboot/nfs.html**.

   - 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' >> /etc/ex
     # /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
  ```

8. 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"
   rarpd=YES         rarpd_flags="-a"
   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.

Now, netboot your Sparc from the server by entering the appropriate **boot** command at the monitor prompt. Depending on the PROM version in your machine, this command takes one of the following forms:

sunmon

> > **b le()netbsd**

OpenBoot PROM 1

> ok **b le()netbsd**

OpenBoot PROM 2

> ok **boot net netbsd**

After the initial probe messages you'll be asked to start the install or upgrade procedure. Proceed to the section **Running the installation scripts.** below.

Installing NetBSD by using a bootable tape

Sun sets their tape drives to SCSI ID 4. We recommend you do the same.

Get the tape images (`installation/tape/tapefile1.gz` and `installation/tape/tapefile2`) and transfer them to a tape. Make sure you use the 'no rewind scsi tape' device. Run the following commands:

```
# gunzip tapefile1.gz
# mt -f /dev/nrst0 rew
# dd if=tapefile1 of=/dev/nrst0 bs=4k conv=osync
# dd if=tapefile2 of=/dev/nrst0 bs=4k conv=osync
```

Now you can transfer the NetBSD 10.2. distribution sets from (`binary/sets`) to the tape by using a series of additional

```
# dd if=<aset>.tgz of=/dev/nrst0 bs=4k conv=osync
```

commands. See also the section **Getting the NetBSD System on to Useful Media.** Be sure to mark the location of these files on the tape; you'll need them during the installation procedure.

Insert the tape into your Sparc's tapestation. Boot the installer by typing the appropriate command at the PROM prompt:

sunmon

> **b st()**

OpenBoot PROM 1

ok **boot st()**

OpenBoot PROM 2

ok **boot tape**

After the initial probe messages you'll be asked to confirm the tape device and tape file number containing the installation tools. Then, proceed to the section **Running the installation scripts.** below.

Running the installation scripts

The first thing that the installation scripts will inquire about is the type of console you're using on your machine. If you're using a Sun frame buffer display, choose the terminal type **sun** (this is presented as the default) .

Note: if your frame buffer device is a cgsix Sbus board, you should change the terminal type to **sun-cgsix** to avoid triggering a bug in the board's firmware code that can garble your screen.

If you're using a terminal device connected to a serial port, choose a terminal type appropriate for that device, e.g. **vt100** or **vt200**.

Next, you will be presented with a choice of two installation methods: a new full-screen **sysinst** program, or the traditional script-based **miniroot** program. The **sysinst** program is easier to use, but the **miniroot** program is more flexible. Both of these installation methods will follow the same set of procedures and will fully install NetBSD on your Sparc.

You will frequently be asked for confirmation before the system proceeds with each phase of the installation process.

Occasionally, you will have to provide a piece of information such as the name of the disk you want to install on, partitioning information, or IP addresses and domain names you want to assign. If your system has more than one disk, you may want to look at the output of the dmesg(8) command to see how your disks have been identified by the kernel.

Note: If you repartition a hard drive, be certain that every partition starts on a cylinder boundary for compatibility with the Sparc ROM.

The installation script goes through the following phases:
- determination of the disk on which to install NetBSD
- checking/creation of the partition information on the disk
- creating and mounting the NetBSD file systems
- setup of network configuration
- extraction of the distribution tar files
- installation of bootstrap programs

Now try a reboot. Initially we'd suggest you **boot sd()netbsd -bs**, then try multiuser after that. If you boot single-user the NetBSD incantation to make the root file system writable is

mount -u /dev/sd0a /

Your PROM might have been setup to instruct the boot program to load a file called `vmunix`. On OpenBoot PROM systems you can change it to load `netbsd` instead using the following commands:

OpenBoot PROM 1

OpenBoot PROM 2

ok **setenv boot-from sd(0,0,0)netbsd**

ok **setenv boot-file netbsd**
ok **setenv boot-device disk:a**

Congratulations, you have successfully installed NetBSD 10.2.

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. Before all else, read postinstall(8).

2. Configuring /etc/rc.conf

   If you or the installation software haven't done any configuration of /etc/rc.conf (**sysinst** normally 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 **sun** for a local console, or whatever is appropriate for your serial console (some systems display garbage with a **sun** terminal type, you may need to use **sun-ss5**) 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, along the lines of

   ```
   ifconfig_le0="inet 192.0.2.123 netmask 255.255.255.0"
   ```

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

   ```
   ifconfig_le0="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 networking, DHCP can be used by setting dhcpcd=YES in /etc/rc.conf.

3. Select the proper terminal devices

   If you are using a serial console, you will have to edit the /etc/ttys file and change sun to the appropriate terminal type, such as vt220.

4. Logging in

   After reboot, you can log in as root at the login prompt. If you didn't 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.

5. 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.

6. The X Window System

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

7. 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 the software may depend upon.

   - More information on the package system is available at
     https://www.NetBSD.org/docs/software/packages.html
   - A list of available packages suitable for browsing is at
     https://cdn.NetBSD.org/pub/pkgsrc/current/pkgsrc/README.html
   - Precompiled binaries can be found at
     https://cdn.NetBSD.org/pub/pkgsrc/packages/NetBSD/
     usually in the sparc/10.2/All subdir. If you installed pkgin(1) in the **sysinst** post-installation configuration menu, you can use it to automatically install binary packages over the network. Assuming that /usr/pkg/etc/pkgin/repositories.conf is correctly configured, you can install them with the following commands:

     ```
     # pkgin install tcsh bash perl apache xfce4 firefox
     ...
     ```

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

     If it was not automatically installed, pkgin(1) can be installed on a fresh NetBSD system with pkg_add(1):

     ```
     export PKG_PATH=https://cdn.netbsd.org/pub/pkgsrc/packages/NetBSD/$(uname -p)/$
     pkg_add pkgin
     ```

   - If you wish to use the pkgsrc(7) framework for compiling packages and did not install it from the sysinst(8) post-installation configuration menu, you can obtain it by retrieving the file
     https://cdn.NetBSD.org/pub/pkgsrc/stable/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.

8. Misc

- Edit `/etc/mail/aliases` to forward root mail to the right place. Don't forget to run `newaliases`(1) afterwards.
- 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 10.2 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.

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.

The upgrade procedure is similar to an installation, but without the hard disk partitioning.

Fetching the binary sets is done in the same manner as the installation procedure; refer to the installation part of the document for help. File systems are checked before unpacking the sets.

After a new kernel has been copied to your hard disk, your machine is a complete NetBSD 10.2 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
```

`sysinst` will attempt to merge the settings stored in your `/etc` directory with the new version of NetBSD using the `postinstall`(8) utility. However, `postinstall`(8) is only able to deal with changes that are easily automated. It is recommended that you use the `etcupdate`(8) tool to merge any remaining configuration changes.

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 10.2.

Note that **`sysinst`** will automatically invoke

```
postinstall fix
```

and thus all issues that are fixed by **`postinstall`** by default will be handled.

In NetBSD 9 and earlier, filesystems listed in `/etc/fstab` would be mounted before non-legacy **`zfs`** filesystems. Starting from NetBSD 10 this order has been reversed.

If you have ever run a version of NetBSD -current between April 18, 2020 and September 23, 2022 (the version numbers used in the affected time range are between 9.99.56 and 9.99.106) your FFS file systems might have broken extended attributes stored.

You should follow this *guide:*
https://wiki.netbsd.org/features/UFS2ea/
before booting the updated system multi-user for the first time.

Note that you do not need to do anything special if you never did run any affected kernel, especially if you have never run NetBSD -current.

The display drivers used for modern GPUs and the whole subsystem supporting it (DRM/KMS) have been updated to a newer version. Unfortunately not all issues with this have been resolved before the NetBSD 10.0 release. You can find a list of issues in the *Open issues with new DRM/KMS:*
https://wiki.netbsd.org/releng/netbsd-10/
section of the release engineering wiki page.

A number of things have been removed from the NetBSD 10.2 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`. See **https://www.NetBSD.org/mailinglists/** for details.

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
https://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 **https://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 NetBSD, 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. 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 Columbia University Computer Science Department for hosting the 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 **https://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!)

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 https://www.NetBSD.org/ for information about NetBSD.
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 Luleå.
This product includes software developed at Ludd, University of Luleå, Sweden and its contributors.
This product includes software developed at the Information Technology Division, US Naval Research Laboratory.
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.
This product includes software developed by Adam Glass and Charles M. Hannum.
This product includes software developed by Alex Zepeda.
This product includes software developed by Alex Zepeda, and Colin Wood 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 Ben Gray.
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 Möller.
This product includes software developed by Boris Popov.
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 Hannum.
This product includes software developed by Charles M. 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 Christian E. Hopps.
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 Christopher G. Demetriou.
This product includes software developed by Christopher G. Demetriou for the NetBSD Project.
This product includes software developed by Chuck Silvers.
This product includes software developed by Cisco Systems, Inc.
This product includes software developed by Colin Wood.
This product includes software developed by Colin Wood for the NetBSD Project.
This product includes software developed by Computing Services at Carnegie Mellon University (http://www.cmu.edu/computing/).
This product includes software developed by Daan Vreeken.
This product includes software developed by Daishi Kato
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 David Jones and Gordon Ross
This product includes software developed by David Miller.
This product includes software developed by Dean Huxley.
This product includes software developed by Emmanuel Dreyfus
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@cryptsoft.com)
This product includes software developed by Eric Young (eay@mincom.oz.au)
This product includes software developed by Ezra Story.
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, and Michael Teske.
This product includes software developed by Ezra Story, by Kari Mettinen, Michael Teske and by Bernd Ernesti.
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 Harvard University.
This product includes software developed by Harvard University and its contributors.
This product includes software developed by Hellmuth Michaelis and Joerg Wunsch

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 W. Dall.
This product includes software developed by Intel Corporation and its contributors.
This product includes software developed by Internet Initiative Japan Inc.
This product includes software developed by Internet Research Institute, 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 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.
This product includes software developed by Jonathan Stone and Jason R. Thorpe for the NetBSD Project.
This product includes software developed by Jonathan Stone for the NetBSD Project.
This product includes software developed by Julian Highfield.
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 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 Kyma Systems.
This product includes software developed by Leo Weppelman and Waldi Ravens.
This product includes software developed by Lloyd Parkes.
This product includes software developed by Lutz Vieweg.
This product includes software developed by Marc Horowitz.
This product includes software developed by Marcus Comstedt.
This product includes software developed by Mark Brinicombe.
This product includes software developed by Mark Brinicombe for the NetBSD Project.
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 Marshall M. Midden.
This product includes software developed by Masanobu Saitoh.
This product includes software developed by Masaru Oki.
This product includes software developed by Matt DeBergalis
This product includes software developed by Matthew Fredette.
This product includes software developed by Michael Smith.
This product includes software developed by Microsoft
This product includes software developed by Mika Kortelainen
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 MINOURA Makoto, Takuya Harakawa.
This product includes software developed by Niels Provos.
This product includes software developed by Niklas Hallqvist.
This product includes software developed by Niklas Hallqvist, Brandon Creighton and Job de Haas.
This product includes software developed by Paolo Abeni.
This product includes software developed by Paul Kranenburg.
This product includes software developed by Paul Mackerras.
This product includes software developed by Paul Mackerras <paulus@samba.org>.
This product includes software developed by Pedro Roque Marques <pedro_m@yahoo.com>
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 QUALCOMM Incorporated.

This product includes software developed by Ravikanth.
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 Rolf Grossmann.
This product includes software developed by Ross Harvey.
This product includes software developed by Ross Harvey for the NetBSD Project.
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 Stephan Thesing.
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 Tommi Komulainen <Tommi.Komulainen@iki.fi>.
This product includes software developed by TooLs GmbH.
This product includes software developed by Trimble Navigation, Ltd.
This product includes software developed by Waldi Ravens.
This product includes software developed by WIDE Project and its contributors.
This product includes software developed by Winning Strategies, Inc.
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 Computer Systems Engineering Group at Lawrence Berkeley Laboratory.
This product includes software developed by the Computer Systems Laboratory at the University of Utah.
This product includes software developed by the Harvard University and its contributors.
This product includes software developed by the Kungliga Tekniska Högskolan 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, Lawrence Berkeley Laboratories.
This product includes software developed by the University of California, Lawrence Berkeley Laboratory.
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 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 by the Urbana-Champaign Independent Media Center.
This product includes software developed for the FreeBSD project
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 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 Eiji Kawauchi.
This product includes software developed for the NetBSD Project by Frank van der Linden
This product includes software developed for the NetBSD Project by Genetec Corporation.
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 for the NetBSD Project by Jonathan Stone.
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 Perry E. Metzger.
This product includes software developed for the NetBSD Project by Piermont Information Systems Inc.
This product includes software developed for the NetBSD Project by Shigeyuki Fukushima.
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 Wasabi Systems, Inc.
This product includes software developed or owned by Caldera International, Inc.
This product includes software developed under OpenBSD by Per Fogelstrom.
This product includes software developed under OpenBSD by Per Fogelstrom Opsycon AB for RTMX Inc, North Carolina, USA.
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).

The End