BERKELEY SOFTWARE DESIGN, INC.

Internet Super Server (BSD/OS 4.0) Administrative Notes July, 1998

Copyright © 1992, 1993, 1994, 1995, 1996, 1997, 1998, Berkeley Software Design, Inc.
Copyright © 1998 Berkeley Software Design, Inc. All rights reserved. BSDI acknowledges all trademarks appearing in this document.

This is Version 4.0 of the administrative notes.

This product includes software developed by the University of California, Berkeley, Lawrence Berkeley Laboratory, and their contributors. Other credits are listed inside.

Table of Contents

Introduction

Welcome to Release 4.0 of BSD/OS! We hope it meets your requirements.

Problem Resolution

BSD/OS includes 60 days of upgrades and full telephone support from the day you receive the initial shipment. BSDI-direct customers in the USA: if you have any problems, please have your customer ID in hand and call 800 487 2738 (or +1 719 536 9346) to speak with a BSDI customer service agent or send email to support@BSDI.COM. Other customers: please contact your vendor. The BSDI service desk is available five days a week, from 9 am to 5 pm United States Mountain Time.

Further support after 60 days is available from BSDI. Customers can purchase additional support in the form of:

- Upgrades only
- Upgrades + PaperMail/email/fax support
- Upgrades + PaperMail/email/fax and phone support
- By-the-minute telephone support

BSDI does its best to respond to customer problems quickly. Any user encountering a problem with BSD/OS is welcome to send an email problem report to problems@BSDI.COM (the `sendbug` program can help you with this). BSDI will attempt to fix such problems and incorporate the solutions into future releases.

See the section on reporting problems for details of the problem resolution process.

Mailing List and Support

Here are the email addresses and USA telephone numbers for problem reporting and support calls (see also the chapter entitled *Problem Reporting Procedures*).

EMAIL:

support@BSDI.COM (for fast response for BSDI supported customers)
problems@BSDI.COM (for reporting any problems that you might encounter)
info@BSDI.COM (for asking general questions or for pricing information)

FAX: +1 719 598 4238

Phone support: 800 487 2738; +1 719 536 9346

Orders: +1 719 593 2082

PaperMail: BSDI Bugs
Suite 110
5575 Tech Center Drive
Colorado Springs, CO 80919

Address Updates

To report changes in your paper mail or email address, please call the BSDI service desk at the numbers above or send email to info@BSDI.COM. BSDI uses the email address on record for each customer to send notices of upgrades and important fixes, so it's important to keep us up to date!

Patches

BSDI provides a patch server in order to make important fixes and enhancements available immediately. See the chapter entitled *Problem Reporting Procedures* for additional information.

Documentation Conventions

Examples of computer interaction use fixed-width type (`like this`) for computer output and bold italic fixed-width type for input that you, the user, type (***`like this`***). Italic fixed-width type (*`like this`*) indicates input that you type but that requires some special substitution (e.g., *`username`* requires you to change the *`username`* to the correct user's name).

Command sequences and computer output appear as left justified, fixed width text. A command line with options in square brackets like this:

```
  # ls [-l]
```

means that the specifications in square brackets (e.g., `-l` in our example) are optional.

Special keys on the keyboard, such as the Enter key, are referred to like this: <Enter>.

General Precautions

Please read the directions before installing the software! Skipping any step might cause the installation to fail. If you call for help, we will always go back to the first step and discuss all the directions.

While the installation process makes every effort to avoid losing or damaging any of your data, the ***Express Install procedure erases the primary disks***.

Please backup all of your data now! While the installation process makes every effort to avoid losing or damaging any of your data, it's too valuable for you to take any risk! See the section **Backing Up Your System** in the release notes for the version of BSD/OS that you have installed on your system.

New Features of BSD/OS 4.0

Hardware: Multiprocessor Support

BSD/OS 4.0 is BSDI's first release to include support for multiprocessors. The system supports many Intel-standard MP systems. As this is the initial release, there are several caveats. First, in the current version, only one CPU can run system calls at a time. This means that workloads with substantial user-load processing in more than one process can benefit from additional CPUs. But, workloads with a large percentage of system (kernel) time may not benefit at all. Second, hardware vendors seem to have taken many liberties with the Intel MP spec. Getting some of these systems running has taken some manual tweaking. Although this facility is relatively new and testing on many new MP systems has required such tweaks, once running, most systems have been quite stable and reliable. This feature was updated several times during the BSD/OS 4.0 beta test, and will continue to change over the following months. For up-to-date information on known working hardware configurations, as well as the latest configuration software please see the BSDI web page:

```
http://www.bsdi.com/products/internet/mpmb
```

Network

- **NFS: Network Lock Manager, Network Status Manager**
  This feature consists of daemons to support the standard NFS lock and status protocols. This means that BSD/OS will be able to do file locking between it and other systems (e.g., Solaris) when the filesystems on which they reside are mounted via NFS. This feature includes both client and server components.
- **IP security option (IPSEC), ISAKMP key-management server**
  Support authentication of IP packets. For domestic sites, it also supports the encryption of IP packets.
- **Virtual Private Network**
  BSD/OS 4.0 supports the ability to set up a Virtual Private Network (VPN). In BSD/OS, a VPN is the ability to set up an authenticated and/or encrypted tunnel between separate networks.
- **IPv6 (next generation IP)**
  Version 6 of the IP protocol. The main feature of IPv6 is 128-bit addresses, instead of the 32-bit IPv4 addresses.
- **IP: Packet Filter**
  Flexible extensible IP Packet Filtering. The system supports multiple simultaneous filtering mechanisms with filtering based on the Berkeley Packet Filter (BPF) machine code. Two filtering points allow the modification of incoming and outgoing packets. Filtering of local packets (host machine is the source or destination) as well as forwarded packets is supported. Filters may report packets to the user level for logging or further processing (packets that are very difficult to process may be filtered at user level at the cost of some performance).

- **IP: hashed address lookup for faster virtual hosting**
  The recognition of local IP addresses is now done with a hash table instead of with a linear search. This will improve the performance of systems with large numbers of virtual addresses.
- **IP: statistics per virtual host**
  Packet and byte counts are now recorded for each IP address configured on the system. This allows for the monitoring of usage by virtual and real addresses. The `netstat -is` and `systat -netstat` output has been updated to report these statistics.
- **Frame Relay**
  Support for the Frame Relay protocol over synchronous serial connections. Both DTE and DCE modes are supported, though DCE mode **packet switching** is not supported.
- **IPX/IP gateway (with BSDI Internet Client)**
  This feature provides two components. First, a proxy daemon that is installed on the BSD/OS machine. Secondly, a winsock.dll that is installed on IPX based PC's (Windows 3.1 or Windows 95/98/NT). These provide full TCP/IP services for the Windows PC through the proxy server. Since there is only one actual IP address being used (the IP address of the BSD/OS proxy server), security is greatly enhanced and there is no need to individually administer IP addresses for each workstation. In addition, a single dynamic IP dialup account from an ISP may be used to provide internet access to an entire group of workstations. The facility also provides auditing and filtering features to allow more efficient use of an internet connection.
- **NetCon/IPX: File Server (NetWare 3.12 File/Print compatible)**
  The core portion of the BSDI IPX services allows any BSD/OS system to become either a Novell file and print server, or a Novell client for file services. A five user license key can be obtained from BSDI's web site. See the chapter on *Configuring IPX/SPX Services* for more detailed information and information on how to get a license key for more than five users. A typical use for this would be allowing the sharing of file and print resources between Unix and Novell systems. A single BSD/OS machine could serve files to Unix machines with NFS and Novell clients via IPX. Any printer defined in the OS system's printcap appears as a printer to Novell clients.

  It should be noted that with IPX services installed and running on a BSD/OS machine, it appears to be a Novell 3.12 server to Novell systems (clients or servers).
- **POSIX 1003.12 socket interface to networking, RFC 2133 support.**
  The socket networking interface has been modified to align with draft 6.5 of the upcoming POSIX 1003.12 standard, plus some changes from draft 6.6 as well as RFC 2133. New interfaces such as getaddrinfo() have been added. This allows applications written to the POSIX interface to be supported, as well as the thousands of programs using the historical BSD socket interface.

File System Support

- **Fast-Filesystem: Trickle sync**
  The release has replaced the historic 'update' process with a new 'trickle sync' facility. This approach evens out the load on the underlying I/O system and avoids writing short-lived files.
- **Fast-Filesystem: Soft-dependency updates**
  The release contains a new experimental facility called soft updates that allows the fast filesystem to eliminate most synchronous writes without jeopardizing the integrity of the filesystem data.

Hardware: Drivers

- **SCSI wide enabled for most SCSI host adapters and Ultra SCSI, various SCSI improvements.**
  Both `aic` and `ncr` now support Ultra. With non-wide drives this allows transfer rates of 20 MB/s, with wide drives this allows transfer rates of 40 MB/s.

  It is now possible to designate a target as being a RAID device, and send more operations in parallel. This can significantly improve performance on RAID devices.

  It is now possible to enable/disable sync, disconnect, Ultra, wide, and tags on a per target basis. This allows the system to take advantage of these features on other targets when a single defective target exists.
- **Blue Heat**
  This release now has support for Blue Heat multiport PCI cards.
- **Multispeed COM**
  A standard serial port cannot go higher than 115200 baud. This new feature provides generic support for serial boards that give N times clock speeds (e.g., 12x or 16x).
- **Compaq NCR SCSI**
  We now include the Compaq SCSI Manager driver (compaqscsi) to support variants of the NCR SCSI part used on Compaq systems that are not supported by the BSDI ncr driver.
- **Compaq SMART-2 RAID**
  It is now possible to install and boot BSD/OS on logical disks exported by the Compaq SMART-2 RAID controller. This allows users to take advantage of the performance and reliability features of the Compaq SMART-2 RAID controller. Compaq-provided, stand-alone utilities are still used to configure the controller.
- **AMD PCnet-PCI Ethernet**
  This is a new driver in 4.0, for 10Mb/s ethernet boards using the AMD PCnet-PCI chipset. Some NE5500 clones use this chipset. The driver also supports 100Mb/s boards. The chips supported are the Am79C971 (PCnet Fast) and Am79C972 (PCNet Fast+) as well as the older 10Mb/s chip (79c970).
- **PCI bha SCSI controllers**

These controllers are now found on the PCI bus (no need to enable the "ISA compatibility" port). Also, more than 2 bha controllers are now supported by the generic kernel.

- **Plug and Play for Ethernet, Modem and Sound Cards**
  We support PnP for the following:
  - ○ 3c509
  - ○ 3c508
  - ○ Soundblaster 16, AWE-64
  - ○ IDE controller cards (wdc)
  - ○ Modem and other com port cards

  There are more devices showing up which only support PnP. This is an initial cut at supporting the most needed PnP devices.

Other Kernel Changes

- **ELF execution support**
  The system now executes ELF programs in addition to a.out and COFF programs. Almost all programs on the system are now ELF programs, and ELF is the default format for newly compiled programs.
- **Reduced interrupt overhead**
  The kernel now waits until a hardware interrupt occurs before physically masking the interrupt (which is a slow operation).
- **IP checksum speed-ups**
  Many IPv4 checksums are now computed in-line with machine specific code.
- **Bootable CD-ROMs**
  It is now possible to build CD-ROMs that are directly bootable on many CD-ROM drives. The BSD/OS 4.0 Install CD is now bootable. If the BIOS supports bootable CDROM's the installation may be run entirely from CDROM.
- **Kdebug and KTR**
  A console-based kernel debugger is now available, as well as a very-low-overhead, kernel-event-tracing facility.

Libraries

- **ELF dynamic (and static) libraries**
  BSD/OS 4.0 provides dynamically linked versions of all of our standard libraries. Dynamically linked ELF shared libraries and shared objects are a widely used industry standard, and they are much more flexible and maintainable than the statically linked shared libraries.
- **Assembly coded math library routines**
  On systems with hardware floating point, programs can use a version of the standard math library (`-lm`) that implements many core functions using hardware transcendental instructions.
- **Enhancements to the BSD Authentication.**
  BSD Authentication has been enhanced. Password expiration times have

been moved to the individual login scripts. Account expiration, no-logins check and the home directory check have been moved to a common routine used to approve a user after authenticating them. Virtually all mechanisms of becoming a user (login, ftpd, ppp, etc) now utilize the approval routines.

Programs

- **X Server upgrade**
  We've upgraded our X11 distribution to X11R6.3. The major new functionality in X11R6.3 is support for World Wide Web integration, protection of data from "untrusted" client connections, a bandwidth- and latency-optimized protocol for using X across the Internet, a print protocol following the Xlib API, and support for vertical text writing and user-defined characters in the Xlib implementation.

  During the installation, you may now choose which X server you prefer from a list of available servers. BSD/OS 4.0 provides a commercial X server from Metro Link, Inc., as well as XFree86. If upgrading from an earlier release that was licensed for the Xi Graphics server, you may elect to keep that server. The Metro Link server and the XFree86 server use new graphical installation utilities.
- **Enhanced GCC support**
  GCC 2 is the standard compiler. We now distribute GAS 2.8.1 as the standard assembler; it has much broader opcode support and fewer bugs.
- **Netscape Navigator and Composer 4.0**
  BSD/OS 4.0 comes with a newer version of Netscape's Navigator browser, and Netscape's Composer editor for HTML files.
- **New sendmail**
  We now include sendmail 8.9.0.
- **New ftpd**
  We ship a completely different ftpd, which we think is more secure, better integrated and more functional than the old wu-ftpd. The wu-ftpd sources are still provided with contributed code.
- **Scsicmd: user-configurable command descriptions**
  The `scsicmd` program now uses a human-readable, modifiable table of command descriptions rather than a compiled-in list of command data.
- **MaxIM additions/improvements**
  MaxIM offers simplified configuration of virtual hosts for mail and web servers. A Japanese version of MaxIM is now also available.
- **Many upgrades to contributed software**
  We provide updated versions of many programs that we have integrated into BSD/OS from customers and developers.

Bug Fixes

- **Contributions from staff, customers and other developers**
  We have integrated many bug fixes from staff, customers and other developers. LOTS of bugs have been retired.

Supported Hardware and Hardware Configuration

This list is current at the time of this printing. For the most current list of supported hardware, please visit BSDI's World Wide Web site (http://www.bsdi.com). The supported hardware and hardware configuration page is found at:

```
http://www.bsdi.com/products/internet/hardware
```

This chapter includes lists of hardware supported by BSD/OS 4.0, as well as hardware for which unsupported drivers are available. Listed hardware is supported unless noted. Please take care to read notes regarding a particular piece of hardware. The hardware is listed by type. Some notes are found at the end of the chapter.

Most entries are preceded by the name of the driver used for that hardware. The drivers have manual pages in section 4 of the manual; use the command

```
# man 4 driver
```

to get more information about a specific driver.

If you need support for hardware that BSDI does not yet support, please let us know so that we can add it to the list of needed hardware, as well as judge what support is most useful to our user community!

Device Recommendations

- For various reasons, using some of the supported hardware cannot be recommended for use in production systems. Generally, due to hardware limitations, these devices do not offer adequate performance and/or reliability for busy servers. These devices are noted as *"Performance is inadequate for use on server machines"* in the sections that follow.
- Some hardware support is contributed to us by individuals or hardware vendors. While BSDI does not fully support this software/hardware, we will not ask for it to be removed before we can consider problems that might occur. If you have problems with these devices, please contact BSDI support. If we are unable to help, we will either forward the information or refer you to the appropriate party. These devices are noted as *"Contributed driver, limited support"* in the sections that follow.
- As a convenience to our customers, BSDI includes some hardware support for which BSDI offers no support at all. While this code is not known to be broken, **users are not encouraged to use these devices**. The drivers might well be broken and cause your system to break in all the worst possible ways (e.g., lose your entire set of files). Forewarned is forearmed. These devices are noted as *"Supplied, but unsupported driver"* in the sections that follow.
- Some devices are compatible (to some degree) with another fully supported device. They have been reported to work by a customer, but BSDI has not

tested them. These devices are noted as *"Compatible with another supported device, but untested"* in the sections that follow.

- Some hardware drivers are of an experimental nature and the support is limited and only available with a source code license. These devices are noted as *"Experimental driver"* in the sections that follow.

Autoconfiguration of Devices

BSD/OS includes an "autoconfiguration" facility that probes your system to determine which peripherals are present each time your system boots. It can determine many device parameters automatically, but other parameters (listed below section) must be set up as expected by the operating system. Booting from the floppy uses a GENERIC kernel that has a large selection of devices – but not all of them. The default kernel installed on the root filesystem is also a GENERIC kernel.

Device Notation

This chapter contains lists and tables of devices that are supported by BSD/OS 4.0. The hardware is divided into categories of devices. Where possible, either a list *or* a table is provided for a set of devices. Tables are used to provide configuration information and often provide the manufacturer's models that are supported. Sometimes the table will become unmanageable with all the device information. In these cases a list is provided which will provide model information, and the accompanying table will have the configuration information, without all of the model information. If only a list is provided for a set of devices, then there is no configuration information for the devices (likely because there is no way to configure the devices, or all configurations are automatically discovered).

In both the tables and lists, device names that appear in **bold** letters (e.g. **abc**) are devices which are found in the GENERIC kernel. If they do not appear in bold letters (e.g. abc), they are not found in the GENERIC kernel.

The device names presented are the names of the device drivers in the kernel, either found in the GENERIC kernel, or configured into the kernel after installation.

Adding devices not found in the GENERIC kernel

Devices not found in the GENERIC kernel may be configured only by building a new, custom kernel. The GENERIC kernel configuration file includes examples for configuring these files but each of those examples has a # at the beginning of the line, indicating that the line is a comment that should be ignored by the `config` program. Some devices may also be used at other addresses with localized kernel configurations; see section 4 of the online manual pages. See the chapter "Rebuilding the Kernel" for complete information on making and installing a custom kernel.

Within this chapter you will find tables to aid with the hardware configuration. For items with IRQ, DRQ, Port and IO mem requirements, the requirements will be enumerated. For these characteristics, an asterisk means the item is probed to find the value to which the card is configured (thus, all valid hardware configurations will be determined automatically).

Note that it is possible to override some of these parameters at boot time (see the `boot -dev` command in the *Debugging Hardware Setup and Booting* section).

If your configuration doesn't conform to these specifications, make it match or override these parameters when booting.

It is extremely important that you avoid conflicts in the port addresses, IRQ, and "iomem" addresses between and among the peripherals. Newer systems with Plug-n-Play and PCI also require you to ensure the interrupts are properly configured.

Conflicts are one of the biggest sources of reported service problems. Please pay extra attention to this part of the configuration. Check any notes regarding your hardware for any configuration requirements.

PCI and EISA devices

These devices do not require a base address to be configured, however interrupts must be set up for PCI devices using the system's BIOS setup menu (many newer BIOS's automatically assign interrupts with no user intervention). PCI devices should use IRQ values that are not used by any ISA or EISA device (where configurable); if the BIOS ends up assigning an IRQ that is used by another ISA peripheral you might have to reconfigure the ISA peripheral for a different IRQ. In most cases PCI peripherals may share an IRQ, however if the kernel reports an interrupt conflict during boot, the driver in question is not able to share the IRQ. As a general rule it is better to assign a separate IRQ to each PCI device (or class of devices, such as disk vs. network) for performance reasons.

CPU

Intel 80386, 80486, Pentium®, Pentium Pro®, Pentium® II, and compatibles such as AMD-K5™ & AMD-K6™ and Cyrix® 5x86 & 6x86.

BSD/OS does support processors with MMX capabilties, although the kernel and BSD/OS supplied utilities do not take advantage of the MMX instructions. User-supplied applications may make use of the MMX instructions.

Multiprocessing support requires Pentium family processors with integral APIC. APIC stands for "Advanced Programmable Interupt Controller" - the programming of the APIC determines how interrupts are routed to CPUs in our symmetric I/O environment when running multiple CPU's. For Dual Processor Pentium (and Pentium MMX) systems all processors must be at the same stepping level. It is suggested that same stepping CPU's be used in Pentium Pro, Pentium II, and Pentium Xeon systems, however this is not a hard requirement. A diagnostic program (`/usr/sbin/cpu check`) is included that can be run from uniprocessor mode to determine if a CPU is capable of multiprocessor operation. A list of known compatible motherboards and chipsets can be found on the BSDI World Wide Web site:

```
http://www.bsdi.com/products/internet/mpmb
```

FPU

npx: 387 or equivalent (486DX, Pentium, Pentium II, Pentium Pro, etc. have on-chip FPU) Automatic software emulation when FPU unavailable.

The BSD/OS kernel supports the floating point coprocessor or automatically emulates the hardware if none exists.

RAM

4MB and up; we have tested up to 2GB, but there is no known reason the system will not run with more than that. A minimum of 16MB is recommended to run X11, and 6MB is the minimum required to run the install procedure.

Some older (Pentium and Pentium Pro) motherboards claiming support for large amounts of memory do not work reliably with more than 128MB installed. This problem has been especially acute when using 64MB SIMMS. BSDI strongly recommends the use of ECC memory on large memory configurations, if the chipset does not support ECC, then parity is still much better than no parity.

Display

vga: Mono, CGA, EGA, VGA, SVGA, XGA, SXGA (VGA or better required for X11 and GUI administration)

Keyboard

pccons: Standard PC/AT and PS/2 keyboard and compatibles

Floppy

The 2.88 MB floppies are supported at 1.44 MB density.

Dev	Name/Details	Port	IRQ	DRQ
fd	Floppy Disk 3.5″ Dual-sided (720K, 1.44Mb), 5.25″ Dual-sided (360K, 1.2Mb)			
fdc	FD Controller standard PC/AT	0x3f0	*	2

Parallel

Dev	Name	Port	IRQ
lp	Standard PC/AT Parallel Port	0x378,0x3bc	7

Serial

Serial devices with FIFO support (16550) are strongly recommended. The *PnP* serial ports (including modems) must be a standard 16550 compatible PnP device.

Dev	Name	Port	IRQ
com	COM port - first	0x3f8	*
com	COM port - second	0x2f8	*
com	PnP serial cards and modems		

Multiport Serial Cards

com:	Generic "COM" multiport
com:	MU-440
com:	USENET II
com:	AST AST-4
com:	Boca 4-port async
com:	MOXA C128+, C102P, C102H, C102HI, C102HIS, C104A, C104P, C104H, C104S, C104HS, C168P, C168H, C168HS
aimc:	Chase Research IOPRO, Iolite
bheat:	Connect Tech Blue Heat
cy:	Cyclades Cyclom-Y
digi:	Digiboard PC/4E, PC/8e, PC/8i (w/64k), PC/16e, PC/16em
digi:	Digiboard PC/XEm (to 64 ports), PC/Xr
eqnx:	Equinox SST
ms:	Maxpeed SS-4/2, SS-4PLUS, SS-8/2, SS-8PLUS, SS-16/2
rc:	SDL
rp:	Comtrol RocketPort, Rocketport RA
si:	Specialix SLXOS 8-32 (PLUS and HOST)
stl:	Stallion EasyIO, Stallion
stli:	Stallion EasyConnection 8/64, ONboard, Brumby

Multiport Serial Cards				
Dev	Manufacturer	Port	IRQ	iomem
com	Generic multiport [22]			
com	MU-440			
com	USENET II	0x120	*	
com	AST	0x2a0	*	
com	Boca			
com	MOXA [10]			
aimc	Chase Research [2]	0x180	*	0xcc00
bheat	Connect Tech	*	*	
cy	Cyclades Cyclom-Y [2]	-	10	0xd4000
digi	Digiboard PC [34]	0x220	*	0xd0000
digi	Digiboard PC/X [41]	*	*	0xd0000
eqnx	Equinox SST [2]	0x220	*	0xd0000
ms	Maxpeed	-	-	0xd4000
rc	SDL RISCom/8	0x220	*	
rp	Comtrol [2]	0x180	-	
si	Specialix [2,11]	-	*	0xd8000
stl	Stallion	0x2a8	15	
stli	Stallion	0x2a8	15	

[2] Contributed driver, limited support. See *Device Recommendations*.

[10] Compatible with another supported device, but untested. See *Device Recommendations*.

[11] Data corruption (of disk data) has been observed when using the Specialix HOST card with an Adaptec 1542CF in a DX2/66.

[12] These devices may not be supported past the first two serial ports in the GENERIC kernel.

[22] When using serial "COM" devices, FIFO support (16550) is strongly recommended.

[41] The IRQ must be configured explicitly for DigiBoard cards with jumper-selectable IRQ. Multiple DigiBoard cards can use the same iomem address. The DigiBoard (isa) example can be used with any supported ISA cards: PC/4e, PC/8e, PC/16e, PC/8i, PC/16em.

Mice

Bus Mice			
Dev	Model	Port	IRQ
bms/lms	ATI Ultra, Ultra Pro		
bms	Microsoft	0x23c	5
lms	Logitech	0x23c	5
pcaux	PS/2 style	*	12

Serial Mice

Standard serial (com port) mice:

Logitech
Microsoft
MMSeries
MouseMan
MouseSystems

SCSI Controllers

Adaptec SCSI host adapters

aha:	1540B, 1542B, 1540C, 1542C, 1540CF, 1542CF, 1542CP (154xCx supports Fast SCSI-II)
aic:	2740, 2740W, 2742, 2742W, 2740AT, 2742AT (one channel only)
aic:	2940, 2940W, 2940UW, 2944W
aic:	on-board/other controllers based on AIC-7850, AIC-7870, AIC-78804
eaha:	1740A, 1742A (support Fast SCSI-II)
sa:	on-board/other controllers based on AIC-6260, AIC-6360
sa:	1520, 1522

Adaptec SCSI host adapters			
Dev	Port	IRQ	DRQ
aha [4,36]	0x330, 0x334	*	*
aic	*	*	-
eaha	*	*	-
sa	0x340	*	-

[4] The Adaptec `aic` driver does not currently support the 284x VLB cards or the AIC-7770 on the motherboard. Only one channel is supported on the 274xAT twinchannel cards; this is not expected to change.

The 274xAT twinchannel cards work by multiplexing two SCSI controllers over a single interface. Since the controllers do not run in parallel, performance can be expected to be approximately equivalent to a single-channel host adapter – and considerably worse than two independent single-channel host adapters. Therefore BSDI does not anticipate supporting these cards in dual-channel mode.

The Plug and Play option must be disabled on Adaptec 1542CP host adapters.

[36] The Adaptec 1740 family of SCSI host adapters can be configured in either "standard" (1540B-compatible) or "enhanced" (EISA) modes. BSD/OS supports both modes; enhanced mode should give better performance. Use the EISA Configuration Utility (via DOS) for your system and the configuration floppies provided with your 1740 to set your adapter to the desired mode. Sometimes the default transfer rate of 10 MB/s in enhanced mode won't work for your particular SCSI bus; if at first you don't succeed, try a lower rate.

Mylex/Buslogic/BusTek SCSI host adapters

bfp: 930 (narrow/ultra) and 950 (wide/ultra)
bha/aha:BT-445S, BT-445C, BT-542B, BT-545C, BT-742, BT-747, BT-747C, BT-946C, BT948, BT-956C, BT-956CD (differential), BT-958

Mylex/Buslogic/BusTek SCSI host adapters		
Dev	Port	IRQ
bfp	*	*
bha/aha [9,40]	0x330, 0x334	*

[9] Disks on some Buslogic BT-946C adapters, and possibly other Buslogic models, might fail to boot when using the BIOS bootstrap, printing the error message `no bsd label`. If this happens, check the adapter setup using the BIOS program by following the on-screen instructions as the adapter initializes after a power-up. If the BIOS memory area is set to a value below 0x220, change it to 0x220. If this fails to solve the problem, boot from the installation floppy as described in *Additional BSD/OS Notes*. Then use `disksetup` with the `-B` option to install the `aha` bootstrap.
For the BT-948 and BT-958, the card's "ISA compatible port" must be set to *Primary* to use the aha bootstraps. You need not run these cards in ISA compatible mode if using the BIOS boot blocks.
The aha bootblocks expect the controller to respond to address 0x330. This means that even if you have a PCI controller responding to another I/O address, it should also respond to port 0x330. This can be set under 'Advanced options', 'Set ISA compatible IO port = Primary'.
Also, the aha bootblocks expect the bootdisk to be at SCSI target 0, LUN 0. Note that DOS accepts a different setup, but the BSD/OS aha bootblocks need target0, LUN 0) to boot.

[40] Multiple cards of this type are supported by the GENERIC kernel.

Other SCSI Host Adapters

dpt: Distributed Processing Technology
SmartCache Plus ISA: PM2011, PM2011B, PM2011B1
SmartCache Plus EISA: PM2012A, PM2012B, PM2012B2
SmartCache III ISA: PM2021
SmartCache III EISA: PM2022, PM2122

SmartCache III PCI: PM2024, PM2124
SmartCache IV ISA: PM2041
SmartCache IV EISA: PM2042, PM2142
SmartCache IV PCI: PM2044, PM2144
SmartRAID ISA: PM3021
SmartRAID EISA: PM3122, PM3222
SmartRAID PCI: PM3224, PM3334

ncr: NCR Adapters based on the Symbios 810, 815, 820 (wide), 825(wide), 860 (narrow ultra), and 875 (wide ultra)

Other SCSI Host Adapters			
Dev	Port	IRQ	DRQ
dpt [2]	0x170	*	5
ncr [5]	*	*	-

[2] Contributed driver, no BSDI support. See DPT for support.

[5] Some NCR SCSI host adapters determine the geometry of the disk from the FDISK table. This implies that the FDISK table must be created and must be in the proper format. Pre-existing FDISK tables on disks used with other adapters must be zeroed before beginning. The `disksetup`(8) program will automatically create partitions of the proper format (which is to make the ending cylinder/head/sector values end on a cylinder boundary). The relative offset + length might extend to include a partial cylinder.

In order to be used as boot devices, some Symbios adapters require BIOS support, which is not available on all motherboards.

SCSI Disk

sd: Any standard SCSI hard disk
sd: Iomega and Syquest Removable SCSI disks

RAID Devices

cr: Compaq RAID logical disk
crc: Compaq Smart-2 RAID controller
sd: Any external RAID systems emulating standard SCSI disk(s)
sd: Adjile and Mylex RAID sub-systems [7]

[7] Not all combinations of SCSI controllers and RAID sub-systems from Adjile and Mylex work. Mylex DAC960S known to work with the BT956/BT958.

Disk

wd: IDE hard disks (including enhanced IDE)
wdc: IDE disk controller
wdc: IDE/ESDI/ST-506/RLL/MFM controllers
wdc: PnP IDE ports (such as those on sound cards)

Some caching/raid EIDE controllers have been reported to work.

Dev	Port	IRQ
IDE/ESDI/ST-506/RLL/MFM controllers	0x170, 0x1f0	*
PnP IDE ports (such as those on sound cards)	*	*
Primary IDE disk controller	0x1f0	*
Secondary IDE disk controller	0x170	*

SCSI Tape

st: Any standard SCSI tape drive (except the CCS version of the Exabyte EXB-8200)

ISA QIC Tape

Dev	Tape	Port	IRQ	DRQ
wt	Archive Viper QIC-02	0x300	*	1
wt	Everex EV-811, EV-831, EV-833	0x300	*	1
wt	WangTek 5150PK QIC-02	0x300	*	1
wdpi	(ATAPI SFF 8020 R2.6) [35]	-	-	-

[35] BSDI has received reports from customers that read/write ATAPI devices (such as tape or removable rewritable drives) operate using the supplied ATAPI driver. BSDI does not have any of these devices available for testing and debugging, however we will accept support requests if a sample of the failing hardware is made available to us for a short time.

CD-ROM

Dev	Name	Port	IRQ	DRQ
mcd	Mitsumi ISA CD-ROM LU002S [14]	0x334, 0x340	2/9	3
mcd	Mitsumi ISA CD-ROM LU005S, FX001S, FX001D [14]	0x334, 0x340	2/9	6
sr	Any standard SCSI or IDE CD-ROM (ATAPI SFF-8020 R2.6 or earlier)	-	-	-
sr	IDE CD-ROM changer devices (SFF-8020I R2.6 compliant)	-	-	-

[14] The factory configuration might **not** conform to this table. It will probably have to be reconfigured.

Magneto/Optical

sr: Any standard SCSI Magneto Optical drive

CDR

sr: Philips or Yamaha command set compliant CDR devices (such as the HP SureStore 6020i or Ricoh 1420).

Successful operation of a CDR device depends heavily on system load and configuration. The software used to drive CDR devices in write mode and extract red book audio are contributed software and are not supported by BSDI.

Ethernet

Because of the number of supported ethernet devices, this section has been broken up into several parts (by manufacturer) to aid in finding devices. Please check through this section for the device by manufacturer first.

3Com ethernet cards

Dev	Name	Port	IRQ
eb	3C900/3C900B PCI (EtherLink XL)	*	*
eb	3C905/3C905B PCI (Fast EtherLink XL)	*	*
ef	3C509, 3C509B (EtherLink III) [25,37]	0x250, 0x260	*
ef	3C579, 3C592 EISA (EtherLink III) [39]	*	*
ef	3C590 PCI (EtherLink III)	*	*
ef	3C595 PCI (Fast EtherLink) [15]	*	*
ef	3C597 EISA (Fast EtherLink) [15]	*	*
el	3C507 (EtherLink 16)	0x310	*
ep	3C505 (EtherLink Plus)	0x240, 0x320	*
red	3C508 [29]	*	*
we	3C503 (EtherLink II) [37,39]	*	*
eo	3C501 (EtherLink) [3]	0x320	*

[3] Supplied, but unsupported driver. Please use with caution. See *Device Recommendations*.

[15] If an Ethernet card capable of 100 Mb/s and 10 Mb/s operation is set to the wrong speed for your Ethernet, the network might jam (100% collisions). The 3C595/597 initializes its speed to the value stored in EEPROM when it is first powered up; run the `efsetup`(8) utility to set the default network speed before connecting to your network.

[25] BSD/OS does not use the I/O port / IRQ values which can be set using the bundled DOS setup program. To change the value of port or irq, please enter the values at the bootstrap prompt (boot: -dev ef0 port=0x300 irq=5) or change the kernel configuration.

[29] If 3C508 and 3C509 are installed on the same machine, both cards must be forced to use PnP.

[37] The iomem addresses for 3C503 and 3C509 Ethernets are automatically

configured by BSD/OS – no special hardware configuration by the user is necessary. The port number for the 3C509 is also set by the kernel. Note that the 3C509 also uses port 0x100 during autoconfiguration – this can conflict with another device there. Port 0x100 can be changed - see `ef`(4).

[39] BSD/OS supports the 3C503 at 0x280, 0x2a0, 0x2e0, 0x300, 0x310, 0x330, and 0x350.

SMC (or WD) ethernet cards

Dev	Name	IRQ	iomem
de	8434T (2 port 10mbit), 9334BDT (2 port 100mbit) [6]	*	*
de	9332BDT (replaced 9332DST)	*	*
de	EtherPower and EtherPower 10/100 PCI 8432T, 8432BT, 8432BTA, 9332DST	*	*
se	9432TX EtherPower II EPIC-based 10/100 mbit PCI ethernet cards [26]	*	*
we	8003 [1,38,44,48]	*	*
we	8013 EtherCard PLUS/Elite Series [38,44,47,48]	*	0xd0000
we	Ultra and EtherEZ Series 8216 Ultra, 8416 EtherEZ [48]	*	0xd0000

[1] Performance is inadequate for use on server machines. See *Device Recommendations*.

[6] This device contains an on-board PCI-PCI bridge and will only work on systems with a BIOS that initializes PCI-PCI bridges correctly.

[26] An early revision of the 9432TX (using device 83C170 rev E or prior) may exhibit performance problems in certain switched environments when running in full-duplex mode. This problem can be circumvented by configuring the card to work in half-duplex mode, or by calling SMC Customer Support at (800) SMC-4-YOU for a replacement product.

[38] BSD/OS supports WD/SMC Ethernet at 0x280, 0x2a0, 0x2e0, 0x300, 0x320, 0x340, 0x360, 0x380, 0x3a0, and 0x3e0, including the 8003/8013 Elite series, the 8216 Ultra and the 8416 EtherEZ.

[44] The iomem addresses for WD/SMC Ethernets are automatically configured by BSD/OS – no special hardware configuration by the user is necessary.

[47] Supports 16-bit mode and 16K of memory.

[48] Must be connected to a network during boot to be recognized.

DEC (Digital Equipment) ethernet cards			
Dev	Name	Port	IRQ
de	DEC Many cards based on DEC DC21040, 21041, and 21140 chips [31]	*	*
de	DEC PCI Ethernet DE435, DE434, DE450, DE500, DE500-XA, DE500-AA	*	*
df	DEC Cards based on DEC DC21143 chip [31]	*	*
df	DEC DE500-BA	*	*
di	DEC EtherWorks II and III Ethernet Controllers DEPCA, DE100, DE101, DE200, DE201, DE202 (all ISA), and DE422 (EISA) [2]	*	*

[2] Contributed driver, limited support. See *Device Recommendations*.
[31] Not all other cards based on the DEC chips will work.

ZNYX ethernet cards	
Dev	Name
de	ZNYX ZX314 (4 port 10mbit), ZX345-N (1-port, 100mbit), ZX346I (4-port 100mbit) [6,13]
df	ZNYX 345-Q (1-port, 100mbit)
df	ZNYX 346-Q (4-port, 100mbit) [6]
df	ZNYX 348-Q (2-port, 100mbit) [6]

[6] This device contains an on-board PCI-PCI bridge and will only work on systems with a BIOS that initializes PCI-PCI bridges correctly.
[13] Other 21140-based members of the ZNYX family using the DEC chips might work but only the 314 and 346I have been tested.

NE-1000 and NE-2000 ethernet cards	
Dev	Name
ne	NE-1000 [1,3]
ne	NE-2000 and compatibles (ISA) [1,8]
ne	NE-2000 and compatibles (PCI) [1,23]

[1] Performance is inadequate for use on server machines. See *Device Recommendations*.
[3] Supplied, but unsupported driver. Please use with caution. See *Device Recommendations*.
[8] Some NE-2000 clones that use software configuration utilities rather than jumpers fail to work when a midi port is also configured (such as that on a SoundBlaster card).
[23] There are many different manufacturers making these cards; unfortunately all of them use different PCI IDs. It is possible that a particular card is not recognized. In that case, it is required to edit /sys/i386/isa/if_neconf.c and recompile the kernel. (This means that initial installation using these cards

may not be possible). BSDI engineering invites people who have found NE2000-PCI cards with new PCI-IDs to submit the ID to BSDI for inclusion in a future release.

Xircom Ethernet cards

Dev	Name	Port	IRQ
pe	Xircom PocketEthernet II (parallel port)	0x378, 0x3bc	7
xir	Xircom Pocket Ethernet	0x378, 0x3bc	7
xir	Xircom PocketEthernet III (parallel port) [18]	0x378, 0x3bc	7

[18] The Xircom PE-III does not always restart properly on laptop machines after entering and exiting "standby mode". Try

```
# ifconfig xir0 down
# ifconfig xir0 up
```

If that doesn't work, you'll have to reboot.

On some laptops, the Xircom PE-III driver is unable to determine and/or use the default parallel port configuration correctly. If you try the `xir` driver on a PE-III and get a boot-time message to the effect that the selected hardware configuration is unavailable, you should use your SETUP floppy or your ROMBIOS "setup" mode to change the printer port configuration. On some machines, you must change it to "bidirectional" and on others you must change it to "output-only". Please send email to support@bsdi.com to let us know if you have trouble with this and what you finally did that solved it; this way we can share your experiences with other customers who have the same kind of laptop. Some Toshiba models (in particular the T4500) are known to require that the printer port be set to OUTPUT (rather than Bidirectional) with the TSETUP DOS utility in order to operate properly with the `xir` driver even though the PE3TEST program works in Bidirectional mode.

Intel Ethernet cards

Dev	Name	Port	IRQ
ex	Intel EtherExpress 16 [1]	0x260	*
exp	Intel EtherExpress Pro 100B, 100+	*	*

[1] Performance is inadequate for use on server machines. See *Device Recommendations*.

Other Ethernet cards

de:	AsanteFast 10/100 [10]
de:	Cogent EM100 [10]
de:	Kingston EtheRX [10]
df:	Embedded 21143-controller in USR Edgeserver Pro
hpp:	HP EtherTwist PC LAN Adapter/16 Plus, 27247B PC LAN Adapter/16 TP Plus [AUI/UTP], 27252A PC LAN Adapter/16 TL Plus [AUI/BNC] [2]
re:	Allied Telesis RE2000/AT-1700 Series [2,43]

tl:	Compaq Most NetFlex / Netelligent ThunderLAN-based with 10baseT or 100baseTX port [27]
tl:	RACORE M8148 with M8149-010 (10baseT) module or M8149-030 (100baseTX/10baseT) module [24]
tl:	RACORE M8165 (10baseT/100baseTX)
tnic:	TNIC 1500 Transition Eng Fast ISA busmaster DMA NIC
ar:	Aironet Wireless Communications radio network card [3]
pcnet:	AMD PCnet PCI based cards

[2] Contributed driver, limited support. See *Device Recommendations*.

[3] Supplied, but unsupported driver. Please use with caution. See *Device Recommendations*.

[10] Compatible with another supported device, but untested. See *Device Recommendations*.

[24] Support is not included for the M8149-050 (100baseFX) module at this time. Please contact BSDI support for details.

[27] Only PCI-based Ethernet cards or onboard Ethernet is supported at this time.

[43] BSD/OS configures Allied Telesis Ethernet at 0x240, 0x260, 0x280, 0x2a0, 0x300, 0x320, 0x340, and 0x380.

FDDI

Dev	Name	Port	IRQ
fea	DEC EISA FDDI (DEFEA-DA, DEFEA-SA, DEFEA-UA)	*	*
fpa	DEC PCI FDDI (DEFPA-DA, DEFPA-SA, DEFPA-UA)	*	*

Token Ring

Dev	Name	Port	IRQ	iomem
te	SMC TokenLink	0x280	*	0xd8000
tr	3Com TokenCard (alt)	0xa24	*	0xd8000
tr	3Com TokenCard (pri) [33,46]	0xa20	*	0xd0000
tr	3Com TokenLink Velocity ISA [46]	0x280	*	0xd8000
tr	IBM TRA 16/4 (pri) [32,33,45]	0xa20	*	0xd0000
tr	IBM TRA16/4 (alt)	0xa24	*	0xd8000

[32] Due to hardware limitations, only two cards supported per machine. Please ensure that you disabled RAM shadowing in the BIOS settings when using these cards.

[33] Because of hardware restrictions, max 2 cards per machine.

[45] IBM token ring cards and compatibles require two memory areas: one for shared RAM, one for MMIO; see `tr`(4) for details. Be sure that RAM shadowing is disabled for these memory areas! Because of hardware

restrictions, only two cards are supported per machine: on 0xa20 (primary address) and 0xa24 (alternate)

[46] 3Com ISA token ring cards are supported in IBM compatibility mode on 0xa20 (primary address) and 0xa24 (alternate). BSD/OS does support the larger selection of IRQ's that these cards provide. As with the IBM Token Ring cards, these cards require two memory areas: one for shared RAM, one for MMIO; see `tr`(4) for details. Be sure that RAM shadowing is disabled for these memory areas!

High Speed Synchronous

These adapters support the PPP, Frame Relay & Cisco HDLC protocols:

ntwo:	RISCom/N2 single or dual port Up-to-T1 [42]
ntwo:	RISCom/N2csu 1 T1 Port with built in CSU plus 1 Up-to-T1 port [42]
ntwo:	RISCom/N2dds 1 56K Port with built in DDS plus 1 Up-to-T1 port [42]
ntwo:	RISCom/N2pci single or dual port Up-to-T1 [21,42]
rh:	RISCom/H2 dual port @ Up-to-T1
rn:	RISCom/N1 single port @ Up-to-T1

Dev	Port	IRQ	DRQ	iomem
ntwo	0x300	*	-	0xe0000
rh	0x240	*	6	-
rn	0x220	*	-	0xe0000

[21] The N2pci card has experienced significant problems with DMA contention in BSDI's testing. Before trying a new configuration, contact the vendor (SDL) to ensure that the problems have been resolved.

[42] Multiple ISA N2 cards (including CSU and DDS models) may share the same iomem address.

Video Capture

Dev	Name	Port
qcam	Connectix QuickCam (mono)	0x378, 0x3bc

Scanners

sg: HP Scanjet IIC, 3c, 3p, 4c, and 4p SCSI

Scanner support requires third party products (such as XVScan, http://www.tummy.com/xvscan) that are not shipped with BSD/OS.

Sound

The SB Wavetable 16 works, but is only an 8-bit card (not recommended). The SB AWE-64 Value and Gold work as 16-bit cards. These are all PnP devices. Avoid the SB16 Vibra (or WavEffects), this is not really a 16-bit card.

Dev	Name	Port	IRQ
gus	Gravis UltraSound [3]		
mpu	Generic MPU-401 midi interface [3,8]		
mss	MS Sound System [3]		
pas	Pro Audio Spectrum [3]		
sb	SoundBlaster Original [1,3]	0x220	5
sb	SoundBlaster/16 [3]	0x220	5
sb	SoundBlaster/16 PnP [3]	*	*
sb	SoundBlasterPRO [1,3]	0x220	5
uart	6850 uart midi interface [3]		

[1] Performance is inadequate for use on server machines. See *Device Recommendations*.

[3] Supplied, but unsupported driver. Please use with caution. See *Device Recommendations*.

[8] Some NE-2000 clones that use software configuration utilities rather than jumpers fail to work when a midi port is also configured (such as that on a SoundBlaster card).

PC Cards (PCMCIA)

This release contains our release of PCMCIA and APM code. This code has proven itself to be sufficiently stable in our previous release. We do not consider it to be production quality, especially in the areas of configuration and ease of use. Problem reports regarding this software will be treated as beta problem reports and possibly dealt with at a lower priority than other fully supported software.

Combo cards (e.g. modem/Ethernet combo's) are not supported at this time.

PC Card Controllers

pcic:	Cirrus Logic CL-PD6710, CL-PD672X
	IBM KING chip [2]
	Intel i82365 and (almost) fully compatibles
	Intel i82365SL (and compatibles)
	Ricoh RF5C296/396 [2]
	Vadem VG468, VG469
tcic:	Databook DB86082(TCIC-2)/DB86082A(TCIC-2/N) [3,17]

[2] Contributed driver, limited support. See *Device Recommendations*.

[3] Supplied, but unsupported driver. Please use with caution. See *Device Recommendations*.

[17] The hardware support described is of an experimental nature and only available with a source code license.

PC Card Serial/Modem

com: Serial/modem cards ([10] applies to many of these)
B.U.G. Linkboy D64K (64Kbps ISDN Adapter Card)
Caravelle PS-T9600
E-TECH C1414AX
Fujitsu FC14F (MBH10213)
Hayes Optima288
Hitachi HT-4840-40
IBM Serial IR
IPS-5000 (GPS)
Intel Faxmodem 144/144
MEGASOFT Inc. DATA/FAX MODEM SFI-1414
MegaHertz XJ1144
MegaHertz XJ2144
MegaHertz XJ2288
Microcom TravelCard
NEC AtermIC20
NEC TM150
NTT DoCoMo PCMCIA DATA/FAX ADAPTER 9600
NTT IT ThunderCard
PACY-CNV10 (GPS)
Panasonic IR Card CD-JRA101
Most newer serial/modem cards

[10] Compatible with another supported device, but untested. See *Device Recommendations*.

PC Card ATA disks (including flash)

wd: Generic support, the following have not been tested by BSDI[10]:
ACE Technologies DoubleFlash+ 3 MB/6 MB Flash/ATA (IBM 17JSSFP)
BONDWELL Pocket Disk B232
DEC Digital Mobile Media CD-ROM
EPSON EHHD260 260 MB PCMCIA/ATA
EPSON Flash Memory 10 MB(ATA)
EPSON SDP5A-20(SunDisk SDP5)
EPSON SDP5A-40(SunDisk)
HAGIWARA SYS-COM, FLASH ATA CARD 5.2 MB, Sundisk
I/O Data, 40 MB Type III Hard disk (Maxtor OEM)
IBM Flash Memory 40 MB(ATA)
IBM Intelligent Flash Memory 40 MB (IBM 17JSSFP)
Maxtor MobileMax 105 MB PCMCIA/ATA
Maxtor MobileMax 170 MB PCMCIA/ATA
MiniStor 42 MB PCMCIA/ATA

[10] Compatible with another supported device, but untested. See *Device Recommendations*.

PC Card Ethernet

cce: Fujitsu MB86960 based
ef: 3Com 3C589, 3C589B, 3C589C, 3C589D (TPO and Combo)
mz: MegaHertz XJ10BT/XJ10BC/(CC10BT)
ne: NE2000 compatibles

The following cards (based on ne, cce, ef, or mz drivers) are reported to work but are not currently supported:[2]
Accton UN2216, EN2216
Allied Telesys CentreCOM LA-PCM
Cardwell LAN PCMCIA Card (ACCTON compatible)
Contec C-NET(PC)C Series
D-Link DE-650, DE-650CT
Dayna Communication Ethernet Adapter
Digital Ethernet Adapter
E-Card E-Card
Farallon EtherMac (3Com 3C589B compatible)
Farallon EtherWave (3Com 3C589 compatible)
Fujitsu MBH10301, MBH10302, MBH10304
Fujitsu LAN Card (FMV-J182)
Hitachi HT-4840-10, HT-4840-11
IBM CreditCard
IC-Card IC-Card
LinkSys E-Card
Matsushita Electric Industrial CF-VEL211J
MegaHertz XJack Ether XJ10BC/XJ10BT
NCR WaveLAN
NCR WaveLAN
National Semiconductor INFO MOVER NE4100 and IBM compatible Version
NextCom NC5310
PLANET SMART COM CREDITCARD/2000
PMX PE-200
Ratoc REX5535 (MBH10302 OEM)
Socket EA+
TDK DF2814, LanCard CD-021
Xircom Credit Card (Corporate series)

[2] Contributed driver, limited support. See *Device Recommendations*.

PC Card Video Capture

fvc: I/O Magic Focus Video Capture

PC Card SRAM

These cards are supported as DOS filesystems.
mc: Generic memory card devices [3]

[3] Supplied, but unsupported driver. Please use with caution. See *Device Recommendations*.

PC Card SCSI

sa: 152x compatibles
sa: Adaptec SlimSCSI-II APA-1460 [3]
sa: New Media Bus Toaster [49]

[3] Supplied, but unsupported driver. Please use with caution. See *Device Recommendations*.
[49] FCC Id J9MBTSCSI only supported.

Requiring DOS for Hardware Configuration

Occasionally, you'll find you'll have to use the DOS software (supplied with your peripheral card) to set up WD/SMC Ethernet cards, ATI graphic controller cards, and maybe some others.

CMOS Setup

The BSD/OS kernel uses the CMOS settings to distinguish between 3.5" and 5.25" floppy drives. If the CMOS setting is wrong, the kernel will boot but the floppy drive will not work. The configuration output from the kernel when it locates the floppy drives will show you what the CMOS entry says so you can verify that it is correct. If it is incorrect, change the CMOS settings to match the actual hardware configuration.

For BSD/OS, the 3.5" floppy drive must be configured as drive `A:` (fd0). As some laptops do not report the drive type, an unknown type of drive is assumed to be a 3.5" 1.44 MB floppy.

Interactions Among Parallel Ports, pe, and xir Devices

Because Pocket Ethernet adapters are generally not attached at all times, when included in a kernel configuration, pe (Xircom PE-2) and xir (Xircom PE-3) devices are "found" during autoconfig when the relevant parallel port is found. PC Card devices are similarly announced if a slot controller is found (but not immediately), even if no card is present. The configuration message is not an error. Of course, one must attach a PE-2 or PE-3 adapter to use the configured network interface. Since the QuickCam attaches in the same manner, it will always be recognized.

Typical SCSI Bus Problems and Solutions

SCSI problems are almost always configuration problems. Hardware items to check are cables and SCSI termination (only the very ends of the SCSI bus should be terminated; so you should have exactly two terminators on the entire SCSI bus). SCSI BIOS configuration options to try are: disable sync negotiation, disable "fast SCSI", set the DMA transfer rate to 5.0 MB/s, make sure that if parity checking is enabled on your host adapter that all devices have parity enabled.

Even if the SCSI bus is terminated "properly", you could still have termination problems; try an active terminator. External devices often cause problems and cables are very important for external devices; make sure they are short and SCSI-II compliant (many are not). Total length of all cables should be less than 5 feet for best results (and keep in mind that external boxes can have cables inside the box, sometimes 6 inches or more).

It's best if you can swap hardware around to try to isolate problems. Remove things to make the bus simpler.

Fast and Ultra SCSI subsystems are very demanding of the SCSI cables and termination. The majority of SCSI problems that we see are related to poor quality or overlong cables and improper or inadequate termination.

Note: even if "it works under DOS", that doesn't mean there isn't a SCSI bus problem. Different drivers exercise different paths in the system. BSDI's high speed drivers make SCSI hardware work much harder than DOS and Windows drivers.

Configuration Conflicts

We've found that some PC hardware configurations are mutually exclusive and cannot be used in the same machine. In other cases, some hardware combinations might limit the address and interrupt configuration possibilities of a particular card. While we've made every effort to make BSD/OS as flexible as possible in these situations, there are cases where technology fails and old fashioned trial and error succeeds. We urge you to be patient and persistent when configuring interface cards in your machine. We suggest configuring devices in the following order, keeping a map of I/O ports and IRQs you use along the way:

1. Video card
2. Floppy drive controller
3. Fixed drive controller(s)
4. COM ports 1 & 2
5. Ethernet controller
6. Mouse (if applicable)
7. COM ports 3 & 4 (if applicable)
8. Other devices

We've also experienced cards that play tricks on us: These devices have the option of reading their configuration from EEPROM, but sometimes this EEPROM information becomes corrupted by another card's power-on initialization sequence and is reset due to a port/IRQ conflict, etc. If you think this is happening to you, verify that you can boot DOS (from power off) multiple times and have the card retain the same configuration as reported by its configuration utility. If not, you might try using the card at its default addresses or, on some cards, configure the card with jumpers rather than the EEPROM.

IDE CD-ROM problems

We have received a number of problem reports from users attempting to configure an IDE CD-ROM as an IDE slave device with no master device on the cable. This is not a configuration defined by the ATA or ATAPI (SFF 8020 R2.6) standard; since the BSD/OS IDE CD-ROM driver must operate with a wide variety of different CD-ROM devices we do not feel it would be prudent to attempt support of this configuration. The DOS/Windows drivers distributed by manufacturers of specific CD-ROM drives frequently support this slave-only configuration, however those drivers are not required to support devices from different manufacturers and so might use non-standard methods that might interfere with the operation of other standard compliant devices.

Another typical problem occurs when the CDROM is attached to the 2nd IDE port on a PIIX3 or PIIX4 (Intel Chipset) based system. The problem is due to a bug in some BIOS'es (notably: Award and AMI) that doesn't route IRQ15 properly (it leaves it disabled). Many DOS mode drivers do not use interrupts so they work fine in this mode. There is no simple workaround to this problem, the easiest suggestion is to put the ATAPI CDROM on the first IDE port as the slave (and accept the somewhat reduced performance this might cause). Newer BIOS'es (such as those found on Pentium II motherboards) do not seem to have this bug.

Power-saving features

Machines with power-saving ("green") features can occasionally confuse BSD/OS. If you're seeing mysterious problems (e.g., "cannot reboot without pressing reset") try disabling those features or enabling APM support (see *Configuring Power Management*).

X Server Hardware

The software that manages the graphical user interface under BSD/OS is called the *X Window System*. It consists of a server program, which displays graphics on your screen, and client programs, which generate graphical data to send to the server program.

In BSD/OS 4.0, there is now a choice of X server programs, each of which may support a video display card and monitor a little differently. (See the section *Configuring the X Window System* for details on selecting an X server.) BSDI supplies the Xmetro server from Metro Link, Inc., and the XF86 server from The XFree86 Project, Inc. Both of these servers have graphical configuration programs that lead you through the steps of choosing a supported video display card and selecting the appropriate monitor characteristics. Sometimes a particular video card or monitor is not explicitly listed, but you may be able to find a compatible card or monitor configuration instead. If you don't know of a compatible card or monitor, you may select the lowest common denominator, the IBM VGA or generic VGA configuration, which every recent PC video card can emulate.

Xmetro Video Cards

For a comprehensive list of video cards supported by Xmetro, please see *Appendix A*.

Xmetro Monitors

For a comprehensive list of monitors supported by Xmetro, please see *Appendix A*.

XF86 Video Cards

For a comprehensive list of video cards supported by XFree86, please see *Appendix B*.

XF86 Monitors

The XFree86 graphical configuration utility lists a few sample monitor configurations, but it really supports a wide variety of multifrequency monitors of various sizes. If you know the range of frequencies for your monitor's horizontal sync (usually given in kHz) and vertical refresh (in Hz), you should enter them by hand.

Note: *Specifying too high a frequency can actually cause damage to your monitor*, so take care to use the correct values.

Debugging Hardware Setup and Booting

Before You Start

In case of problems with certain boards, please verify that the board works correctly using the board's diagnostic floppy (usually under DOS), *if* the board comes with such a floppy.

Some ISA boards use shared memory to communicate with the CPU. We have seen problems if *RAM shadowing* is enabled. We recommend that shadowing be disabled whenever possible.

Using Boot Information

It is possible to boot a kernel with options to obtain additional information about the autoconfiguration process (e.g., to find out why a device does not configure). It is also possible to control some aspects of the autoconfiguration process, such as inhibiting some devices from being probed or specifying alternate port addresses, memory addresses, and IRQs for specific devices.

Interrupt the normal boot process by pressing the <Space> bar or <Enter> key when you see:

```
  Press any key to interrupt boot sequence
```

prompt. The system will print a `Boot:` prompt and wait for your input.

Autoconfiguration Debugging

The first thing to try when autoconfiguration fails is to disable BIOS shadowing from the CMOS setup menu of your hardware (if possible).

To enable autoconfiguration debugging or to modify the level of output from the kernel autoconfiguration process, enter the `−autodebug` command followed by the desired autodebug flags. Specifying any autodebug flags at the `Boot:` prompt will override any flags set in `/etc/boot.default`. The following flags are supported:

−a Ask; specifying this will cause the kernel to stop at each probe and ask whether the probe should be performed. If you answer y (yes, perform the probe), the result of the probe will be printed. This is useful if you suspect a particular probe routine of causing another device to fail.

−d Debug; this flag causes additional debugging information to be displayed about each device probe that is attempted.

−q Quiet; this flag stops all but the most basic autoconfiguration information from being displayed. This flag is typically set in `/etc/boot.default`.

−v Verbose; this flag causes additional configuration information to be displayed, such as SCSI bus parameters.

For example, to enable both verbose and debugging information when booting from the floppy, use:

```
  Boot: −autodebug −v −d
```

```
Boot: /bsd.gz
```

If you are debugging problems before the system has been installed (you are booting from the install floppy) you have two opportunities to interrupt the boot sequence. The first opportunity is right after the boot loader is started, at this point neither the kernel nor the install filesystem have been loaded into memory; stopping the boot process here is appropriate for debugging problems that are preventing the kernel from finishing autoconfiguration and mounting its root filesystem. The second opportunity comes after the install filesystem and kernel have been loaded; interrupting here is appropriate if you need to make changes to successfully install (e.g. to make an Ethernet or CD-ROM operational). After making the changes, the `start` command is used to continue as illustrated below (with some lines folded to two lines for presentation purposes):

```
loading /boot...........
+------------------------------------------------------+
| Welcome to BSD/OS.  Please wait while the system is |
| loaded into memory.                                  |
+------------------------------------------------------+
press any key to interrupt boot sequence 4 3 2 1 0
basemem = 640K, extmem = 15360K, total 16384K
1065388+59140+99628 start 0x1000
Set up ramdisk of 2560K @ 13816K
Loading filesystem from filesys.gz

Loaded.

        Pausing before booting kernel.

press any key to interrupt boot sequence 4 3 <Enter>
Boot: [enter overrides here]
[...]
Boot: -start -
Copyright (c) 1982, 1986, 1989, 1991, 1993
[...]
```

Changing Parameters for a Device Probe

When faced with the problem that the BSD kernel is looking for a particular device at one address and the device is actually configured at some other address, the best solution is usually reconfiguration of the device. In some cases this might not be convenient or possible. It is possible to change where the kernel will look for a given device through commands to `/boot`.

The command to change a device specification is `-dev`. This command enables you to modify the same set of parameters for a device as are normally specified in the kernel configuration file. For example, to tell the kernel to search for the bha0 (first BusLogic host adapter) device at address 0x334 instead of the default 0x330 address, use the command:

```
Boot: -dev bha0 port=0x334
```

You will then be returned to the `Boot:` prompt so you can enter other commands. Once you're done typing in all the commands, tell `/boot` the name of the file you want to boot (usually `/bsd` for the hard disk, `/bsd.gz` for the installation floppy, or just `-start` – if the kernel is already loaded).

The valid parameters you can specify with the `-dev` command are: `port`, `iosiz`, `maddr`, `msize`, `irq`, `drq`, and `flags`. Wild card or symbolic names for ports are not permitted in `/boot` in the manner they are used in the kernel configuration file; you must use the actual unit numbers instead.

If the probe of a particular device is causing problems you can force it not to be probed by using a port address of –1. For example:

```
 Boot: -dev we0 port=-1
```

would prevent device `we0` from being probed at any of its defined addresses.

Memory Sizing Options

Other commands accepted by `/boot` change the way it sizes memory. These options should not be needed in normal installations, but can be used to override the default behavior in cases where the default behavior causes problems.

- `-memsize` *N* – force memory size to N (N is decimal or hex, with an optional trailing M for megabyte or K for kilobyte); precede hex numbers with ***0x***
- `-basemem` *N* – force size of base memory to N (e.g., to 640K)
- `-cmosmem` – limit search for memory to the CMOS value
- `-extendend` *N* – limit search for memory to this address (*Note:* This is extendend, not extended!)

Resource reservation

On some machines it might be desirable to prevent the kernel from automatically allocating certain resources. The kernel will manage conflicts between devices it knows about, however some machines might have devices that are either unrecognized by the kernel or are not configured into the GENERIC kernel (for example: the sound hardware built into many laptops will claim interrupts and I/O space but the GENERIC kernel does not have sound drivers configured into it). This situation sometimes arises with PCI devices that are not configured into the kernel or are not supported (in this case only the IRQ are an issue).

If you know or suspect that a range of I/O addresses, an IRQ, or an ISA I/O memory region will not work for some reason (such as being wired to a non-recognized device), you may mark these resources as reserved so the kernel does not attempt to use them. The commands are:

- `-iomem reserve` *`addr size`* – Reserve the given ISA I/O memory range
- `-ioport reserve` *`base count`* – Reserve the listed range of I/O port addresses
- `-irq reserve` *`irqno`* – Reserve the given IRQ

Additional Information

This is not a complete list of the options for `/boot`. See the `boot`(8) manual page (included at the back of this manual) for more information in addition to trying the `–help` and `–show` options. All commands for `/boot` can be placed in the file `/etc/boot.default` and they will be executed as if they had been typed at the `Boot:` prompt.

Disk Space Requirements

BSD/OS loads its software in modules called "packages". As such, you can customize which software you load and use. For minimal operation, you can get away with a very small disk usage.

The MB listed for disk consumption shown below might change slightly in the final 4.0 release.

MB	Package (Destination Directory)
0.4	Core `/var` (`/var`) *Minimal `/var` directory; required*
6.0	Core root (`/`) *Minimal root utilities; required*
37.4	Core /usr (`/usr`) *Minimal utilities; required*
7.6	Additional `/usr` (`/usr`) *Remainder of regular utilities; desirable*
15.0	Networking (`/usr`) *Networking utilities; desirable*
21.0	Development (`/usr`) *Compilers, libraries, include files; desirable*
16.6	Manual Pages (`/usr/share/man` & `/usr/contrib/man`) *Preformatted manual pages; desirable*
2.5	Groff (`/usr` & `/usr/share`) *Troff/nroff text processor; desirable*
46.0	Contributed binaries (`/usr/contrib`) *Contributed software; desirable*
59.6	X11 Server (`/usr/X11R6`) *X11 windows server; desirable*
22.6	Core X11 Clients (`/usr/X11R6`) *X11 windows clients (`xterm`, `twm`); desirable*
10.5	X11 Development (`/usr/X11R6`) *X11 libraries and include files; desirable*
4.9	X11 Manual (`/usr/X11R6/man`) *X11 manual pages; desirable*
25.3	Additional X11 binaries. (`/usr/X11R6`) *X11 contributed software; desirable*
12.5	Supplementary docs (`/usr/share/doc`) *Additional system documentation; desirable*

MB	Package (Destination Directory)
5.0	Games (`/usr/games` & `/var`) *Games; desirable*
6.7	Ghostscript (`/usr/contrib`) *PostScript interpreter/preview; desirable*
3.5	Mh (`/usr/contrib/mh`) *MH mail handler; desirable*
24.7	Emacs (`/usr/contrib`) *GNU Emacs editor; desirable*
60.8	TeX (`/usr/contrib`) *TeX text processor; optional*
2.9	Hylafax (`/usr/contrib` & `/var/spool/fax`) *Fax modem software; desirable*
25.6	Kernel Objects (`/usr/src/sys`) *Compiled kernel objects (for reconfiguration); desirable*
3.9	Interviews C++ Graphics Library (`/usr/contrib/interviews`) *Interviews development environment; optional*
35.2	Kernel Sources (`/usr/src/sys`) *Kernel source code; desirable; extra cost option*
15.1	Man Source (`/usr/share/man` & `/usr/contrib/man`) *Nroff sources for manual pages; optional*

If you choose to load the sources and intend to build software, remember to reserve enough disk space to hold the object (i.e., .o) files and the linked binaries.

The sources to the contributed binaries are available on a CDROM included in your BSD/OS shipment.

The tables show approximately how much space each of the distribution pieces requires. These packages are all stored in the `/PACKAGES` directory on the distribution CD-ROM.

Installing Server Software

This chapter outlines the procedures for installing BSD/OS 4.0.

Check Manifest

Begin by checking that the BSD/OS installation kit was complete. It should have contained at least the following:

1. BSD/OS 4.0 INSTALL 3.5" floppy (everyone)
2. BSD/OS 4.0 BINARY (Domestic or International) CD-ROM
3. BSD/OS 4.0 CONTRIBUTED SOURCE CD-ROM (everyone)
4. BSD/OS 4.0 SOURCE CD-ROM (source customers only)
5. BSD/OS 4.0 License Key for this machine

The source packages are an extra cost option. The domestic BINARY CD-ROM contains packages that can not be exported due to licensing restrictions.

Typing on the Console

Type on the BSD/OS console as you would on any other computer console. Use backspace to erase a single character or <Control-U> to erase the entire current line (in the case of a major typing error). Terminate programs by typing <Control-C>.

Installing BSD/OS 4.0

BSD/OS 4.0 may be installed on a computer in one of two ways. For computers that currently have BSD/OS 3.0 or 3.1 installed, BSD/OS 4.0 software may be installed by an *Upgrade Installation*. The Upgrade Installation is designed to preserve the system's current configuration information, upgrading system software and configuration files in place.

For other systems, including computers that have never had BSD/OS installed on them, those running even older versions of BSD/OS, or when there is absolutely no information from the old version that you wish to retain, the software should be installed using the *New Installation* process.

The remainder of this chapter describes the new software installation process. The next two chapters detail the Express and Custom installation procedures, and the following chapter describes the process for upgrading existing systems. If you are upgrading an existing system, you might skip to the chapter *Upgrading BSD/OS*.

New Installation Outline

The installation of BSD/OS 4.0 is quite automated. This section explains the general steps, and subsequent sections detail the specific operations. Feel free to scan the next chapters on installation to learn about installation options before beginning.

A new installation includes the following steps, many of which can be performed automatically:

- Booting from floppy disk

- Mounting a local or remote CD-ROM (and configuring the network if performing remote CD-ROM install)
- Dividing disk(s) into partitions for use by BSD/OS
- Initializing BSD/OS filesystems
- Installing BSD/OS software packages
- Booting from the freshly-installed hard disk
- Customizing system configuration

BSD/OS 4.0 supports two methods for New Installation: *Express Installation* and *Custom Installation*.

Express Installation

Express Installation is the easiest way to install BSD/OS 4.0. Express Installation replaces the contents of the computer's primary disk, optionally leaving room for DOS to be (re)installed at a later time. It divides a single disk into partitions using a standard allocation. It requires that at least 450 MB of disk space be assigned to BSD/OS 4.0. All *Required* and *Desirable* packages (see the manifest) that come with BSD/OS will be automatically installed, including the X11 Window System.

The Express Installation will work when the distribution CD-ROM is accessed over the network. However, for simplest operation, we recommend that you use a CD-ROM drive local to the computer on which BSD/OS is being installed.

Installation on any wd disk (ESDI or IDE drives) that does not have perfect media requires custom installation (perfect media means "no bad tracks").

Custom Installation

Use Custom Installation when finer control over the installation process is necessary or desirable. Custom Installation should be used for computers with small disks, when installing on a disk other than the system's primary disk, or when portions of the primary disk must not be overwritten. Custom Installation also enables user selection of which packages are to be installed. It can also install on multiple disks. However, Custom Installation requires that the installer make many more choices, such as how the disks are to be partitioned, and potentially to provide geometry information for the disks.

The custom installation procedure will ask if disksetup should be ***`automatic`*** or ***`custom`*** as well as if the software selection/installation should be ***`automatic`*** or ***`custom.`*** If ***`automatic`*** is selected for both of these, the installation will be nearly the same as an Express Installation with the exception that a remote installation may be performed even if a local CD-ROM drive is present.

If you are considering a custom installation, you should scan the section *Automatic Disk Configuration With disksetup* for information about disk partitioning.

Prepare for Installation

First: **backup all your data** if you have any contents of value on any disks on the system. The installation process replaces the contents of part or all of one or more disks, and a small error could destroy other data.

Once the data on the computer has been backed up, about 15 minutes will be required to answer installation questions, followed by 8 to 20 minutes while the system is installed from scratch from the CD-ROM (the speed depends on your hardware's speed and very much on the speed of the CD-ROM drive). Upgrades, because they require synchronous disk operations, take 20-90 minutes. Once the system is loaded, you must answer a few questions about the configuration of the computer. This takes between 5 and 15 minutes.

Ensure Computer Is Configured Correctly

Ensure that your hardware is supported and configured according to the rules as described in the *Supported Hardware and Hardware Configuration* chapter.

Most modern disks are pre-formatted. If you have an ESDI disk, perform a low-level format on it by following the directions that came with the disk. If you have one of the rare SCSI disks that requires a low-level format, follow your vendor's directions for formatting it. If BSD/OS is not going to share the disk with DOS, format with native mode (no DOS cylinder or head mapping) and with no interleave.

If you are installing on a disk that was previously labeled by BSD/OS, make sure the label accurately reflects the disk. If the "c" partition extends beyond the end of the disk, the installation procedure will have difficulties configuring the disk, as it believes existing labels to have the correct information.

When you fix the label on a badly configured disk, you will **lose all data stored anywhere on that disk**. **Backup data before doing this.** Fix badly configured disks that **only run BSD/OS** by running the following (*Be sure to use the right disk name!*):

```
# disksetup -W sd0
# dd count=1 seek=1 bs=512 if=/dev/zero of=/dev/rsd1a
```

Fix badly configured disks that **run BSD/OS and some other operating system** by running the following (*Be sure to use the right disk name!*):

```
# disksetup -W sd0
# dd count=2 bs=512 if=/dev/zero of=/dev/rsd1a
```

This destroys your FDISK partition table and thus all data on the disk.

The hard-disk that BSD/OS will boot from must be unit 0 (`C:`). The 3.5" floppy used during installation must be the `A:` unit.

Collect Information for Installation

A few things need to be checked or facts verified prior to installing BSD/OS 4.0:

- The system must have a 3.5" 1.44 MB floppy diskette drive, configured as `A:`. The system must boot from the `A:` drive if a floppy is present in the drive.

- If the system does not have a local CD-ROM drive, it must have a network card that is supported by the default BSD/OS kernel. (See *Supported Hardware and Hardware Configuration* for more details.)
- You must know the login name of the user who will be administering this system (you get to pick this).
- If the system does not have a supported local CD-ROM drive, then the installation must be performed remotely via a supported network card, and you will need:
  1. A second computer, accessible over the network by this computer, with a local CD-ROM drive, and which supports the `rsh` protocol. This is the "remote computer" and is where the distribution CD-ROM will be mounted. The user whose account will be used for remote access must put the hostname or IP address of the machine to be installed into the user's `.rhosts` file, which probably must have permissions like 0600; set them using this command:

     ```
     # chmod 600 .rhosts
     ```
  2. The Internet (IP) address (something like `12.34.56.70`) for the computer upon which the system is to be installed (the "local computer").
  3. The Internet (IP) address for the remote computer.
  4. The directory pathname on the remote computer where the BSD/OS 4.0 BINARY CD-ROM will be mounted.
  5. The login name of a user on the remote computer whose configuration allows `rsh` commands to be executed from the computer to be installed (i.e., their `.rhosts` file is correctly configured).

General Procedures and Requesting Help

During installation, screens of information will be presented to you with the following line near the bottom of the screen:

```
Press <ENTER> to continue:
```

Please read the information carefully. When you are ready to proceed, press `<Enter>`.

Occasionally, the system requires a response and will ask you a question. Most questions you are asked will have a default answer shown in square brackets (e.g., `[express]`). You can answer with the default answer by pressing `<Enter>` without typing anything else. For example, given the question:

```
Use Express or Custom Installation? [express]
```

pressing `<Enter>` is the same as entering ***express*** followed by the `<Enter>` key. To type in some other answer, type it (in full) and then press the `<Enter>` key, (e.g., ***custom***`<Enter>`).

At many of the prompts, additional help is available by typing ***help***`<Enter>`. When you do this, more information will be displayed. Pressing `<Enter>` will then redisplay the original question.

Requesting a Shell Prompt During Installation or Configuration

Under normal circumstances, it should never be necessary to use a command line shell prompt during BSD/OS installation or configuration. If, for some reason you need one, enter ***`!shell`***`<Enter>` at any of the prompts (including the `Press <ENTER> to continue:` prompt). When you exit the shell process, installation or configuration will continue from where you left off.

Notation

The steps below contain both specific instructions (e.g., `Load this disk`) and explanations. Steps that require an action on the part of the installer are marked with the symbol "$\boxed{\rightarrow}$".

Boot From the "INSTALL" Floppy Diskette

If this machine is a notebook and will be installed over an Ethernet PC Card (PCMCIA):

$\boxed{\rightarrow}$ Insert the supported Ethernet PC Card into a PC Card slot.

In all cases:

$\boxed{\rightarrow}$ Place the "INSTALL" floppy diskette in the `A:` drive.

$\boxed{\rightarrow}$ If there is a supported CD-ROM on this computer, insert the BSD/OS 4.0 BINARY CD-ROM in the CD-ROM drive. Power on the computer (or press reset). The system should then display something like:

```
loading /boot...............
+--------------------------+
|  Welcome to BSD/OS.     |
+--------------------------+
press any key to interrupt boot sequence 4 3 2 1 0
loading /bsd.gz
basemem = 640K, extmem = 15360K, total 16384K
1047676+58724+99512 start 0x1000
Set up ramdisk of 2560K @ 13816K
Loading filesystem from filesys.gz
2368K
```

This might take some time as nearly the entire contents of the floppy disk are being read. Once the system has loaded the kernel (indicated by the entire line `1047676+58724+99512 start 0x1000` being printed), it loads a compressed 2.5 MB filesystem into memory. The good news is that once this filesystem is loaded into memory, the floppy disk is no longer needed, except at the point the BSD/OS kernel and bootstrapping program are copied from the floppy to the hard disk.

The numbers will not match exactly, but the lines displayed will be similar.

The `basemem`, `extmem`, and `total` memory information will vary from computer to computer depending on the actual amount of memory found, and whether or not the BIOS requires memory to be reserved for its own use. The `total` is not the sum of available base and extended memory, but rather the total addressable memory (including areas reserved for the BIOS and devices). Don't be concerned if `basemem` is 636K or so instead of 640K.

If the system prints a line like this:

```
NOTE: extended mem (32762K) found differs from
CMOS amount (15296K)
```

then the installation procedure's memory sizing algorithm found more or less memory than the CMOS setup menu indicated was present. By default, the system will use the amount it finds regardless of what the CMOS indicates. It might be possible to update the CMOS by running the CMOS setup program (or the EISA configuration utility on EISA machines). Some computers (notably Dell systems) will always return 16 MB instead of the amount actually present in the system. It should always be safe to ignore this "NOTE" if the memory found is about the right size. If the memory size is wrong, see the memory-sizing options listed in `boot`(8), found at the back of the manual, for possible workarounds.

Once the kernel and root filesystems are loaded from floppy (a few minutes), the screen will display a banner like:

```
Copyright (c) 1991, 1992, 1993, 1994, 1995, 1996
       Berkeley Software Design Inc.
Copyright (c) 1997,1998 Berkeley Software Design Inc.
All rights reserved.
```

This is followed by a (potentially lengthy) report of the devices the kernel found on the system (presented in white letters on a blue background, if possible).

⇒ If you have multiple disks, note the device names for each drive so that you can later tell the system which drives will store which data. You will want to look for devices such as `sd0` or `wd0`. It is not necessary to record the rest of the information; it will be made available to you later.

Several screens of information will then be presented to explain how to respond to the questions asked by the installation procedure. Take your time and read the information carefully. Once you're finished, press `<Enter>` to continue.

You are then asked to enter the type of console keyboard:

```
Which keyboard do you have? [US]
```

Possible keyboard choices include:

BE	Belgium
BG	Bulgaria
DE	Germany
FR	France
JP	Japan
NO.8859	Norway (ISO 8859)
RU	Russia
SE	Sweden
SP	Spain
UK	United Kingdom
US	US
USJP	Japan 106-key switchable with US

⇥ Press `<Enter>` to select the US (standard) keyboard type. Some versions of the keyboard maps make the 'CAPS LOCK' key behave as if it were the 'CONTROL' key. Those versions have '.ctl' as a suffix (e.g., `US.ctl`). Change keyboard mapping later by copying the appropriate file from `/usr/libdata/keyboard` to `/etc/keyboard.map` and rebooting.

All installations will then proceed with the following question:

```
BSD/OS now supports licenses to help administrators
determine if they are exceeding their licensed number
of users. Licenses from BSDI also include a hostid
that is unique among all BSDI licensed machines.
While a license is not required to install and use the
BSD/OS operating system, entering your license now
will prevent several warning messages from being
printed each time the system is brought up as well as
when more than a single unique user is using the
machine. To install without a license, enter the
license of "none".

Please enter the BSD/OS license:
```

⇥ Enter the license provided with the distribution followed by `<Enter>`. The license key is 12 case-sensitive characters and looks something like:

```
Please enter the BSD/OS license: 1+jK pluh G17-
```

(This is an invalid license key, used for example purposes only.) If the license key has been misplaced, enter ***none***`<ENTER>` to continue without a license. The configuration procedure (later) will again ask for the license key in this case.

The next question asked by the installation procedure is:

```
Use Express or Custom Installation? [express]
```

⇥ Press `<Enter>` to start an Express Installation. Type ***custom***`<Enter>` for a Custom Installation and skip forward to that section in these notes.

Special note for notebook installations

If all of the following are true of the machine being installed:

- It is a notebook system
- It has a local CD-ROM drive from which the installation will take place.
- The floppy disk drive and CD-ROM drive use the same bay so only one of the two might be installed at one time.
- The notebook allows "hot-swapping" from the floppy disk drive to the CD-ROM drive.
- A remote installation is not possible

then the boot up procedure is slightly modified. When the system displays

```
Loaded.
        Pausing before booting kernel.
press any key to interrupt boot sequence 5
```

press the <SPACE> bar. The prompt

```
Boot:
```

should be displayed. Swap the floppy drive and CD-ROM drive. At the

```
Boot:
```

prompt type ***-start -***<ENTER>.

The system should now continue to boot. If this is not possible, or the system does not recognize the CD-ROM drive, a local CD-ROM installation is not possible on this notebook.

Continue with the rest of the installation normally.

Express Installation

Express Installation Setup: Local CD-ROM

Skip this paragraph if installing from remote CD-ROM or using custom install.

The Express Installation procedure will automatically determine whether there is a supported local CD-ROM drive and whether the correct CD-ROM media is inserted. If a CD-ROM drive is discovered but the CD-ROM media is not loaded correctly (or an incorrect CD-ROM media is loaded), a message will be displayed asking you to insert the the proper CD-ROM. After you have inserted the correct CD-ROM media, press `<Enter>`.

Express Installation Setup: Remote CD-ROM

Skip this section if installing from local CD-ROM or using custom install.

If Express Installation does not find a local CD-ROM drive, then your system must have a supported network card installed, and there must be a CD-ROM drive on a computer on the network that is accessible from the system to be installed.

⇒ First, mount the BSD/OS 4.0 BINARY CD-ROM on the remote computer, probably using its standard `mount` command. Remember the name of the directory upon which the CD-ROM filesystem is mounted.

  If BSD/OS only finds a single network card in the system, it will automatically be selected. If more than one network card is found, the Installation script will display the list of available cards.

⇒ In the case of multiple cards, select the card that can communicate with the remote computer on which the CD-ROM was mounted.

Once the interface is selected, the Installation script will prompt for the following information:

⇒ The IP address of this computer.
⇒ The netmask for this network.
⇒ The IP address of the remote computer.
⇒ The name of a user account on the remote computer that allows rsh requests from the `root` account on the local computer.
⇒ The directory on the remote machine on which the BSD/OS 4.0 BINARY CD-ROM is mounted.
⇒ The IP address of a gateway machine, if one exists. If the remote machine is on the same local network, no gateway address will be needed.

The Installation script will then attempt to contact the remote computer and find the CD-ROM. If something goes wrong, an error will be reported to you and the above questions will be asked again. This process will repeat until a CD-ROM is found.

Disk Configuration and Software Installation

If both IDE/ESDI and SCSI disks are present, Express Installation might ask which disk is your boot device once it has found the correct CD-ROM. The installation procedure will prompt for the type of drive that is the boot device (that is, the name of the drive from which the system will normally try to boot if there is no floppy in the `A:` drive), usually known as the `C:` drive.

Once Express Installation has identified the drive from which you wish to boot, it will check the size of the drive. The drive must have at least 450 MB of space for Express Installation. If the disk is smaller, you must perform a Custom Installation. If there is more than 450 MB of disk space, some amount of disk space can be reserved for DOS.

Express Installation will ask if a DOS partition is desired. If you answer ***yes***`<Enter>`, then Express Installation will ask you how much space to reserve for DOS. Your answer should be in units of megabytes. To assign 40 MB of disk space, type ***40***`<Enter>`.

The following warning and prompt will be displayed (the second line will be different if the boot drive is not SCSI):

```
Express Installation will now reconfigure your
SCSI sd0 hard drive.

THIS WILL ERASE ANY DATA STORED ON THE DRIVE.
IF YOU HAVE NOT BACKED UP ANY DATA YOU WISH SAVED
THEN YOU SHOULD REMOVE THE FLOPPY DISK FROM THE
DRIVE AND REBOOT YOUR COMPUTER NOW.

Please enter "erase" to continue on, or "no" to
halt the installation.

Erase and reconfigure drive for BSD/OS?
```

If there is any question as to which disk is about to be reconfigured, or **if all useful data on this system has not yet been backed up, remove the floppy diskette from the floppy drive and turn the system off.**

Please read this message carefully! If the correct disk has been chosen **and** you are certain there is no data on that drive that needs to be saved (it is always best to have backed up all the drives, just to be safe!) then
⇒ Type: ***erase***`<Enter>`.

The installation process will reconfigure the primary disk and load appropriate packages.

Once the the packages have been installed, the installation procedure will request the root password:

```
Changing local password for root.
New password:
```

Enter the new password for the root account on this machine and press `<Enter>`. The password will not be displayed as it is entered. The `passwd` program will prompt a second time to verify that it was not mistyped. If the first and second entries do not match, it will once again ask for the password. This repeats until the same password is entered twice.

After all the packages have been installed and the root password has been set, the installation procedure will prompt you to remove the INSTALL floppy from the drive.

Make sure the installation floppy has been removed and press `<Enter>`; the system will then automatically reboot. In rare circumstances, you might have press the "reset" button.

If you chose to reserve space for DOS, the machine will prompt for which system to boot. Initially there is no default; press `<F2>` to boot the BSDI system.

Proceed to the section named *Configuring Your BSDI System*.

Custom Installation

Skip this section if you intend to use Express Installation.

A Custom Installation enables (and requires) dramatically more interaction than Express Installation. If possible, we recommend that you use a local CD-ROM to install BSD/OS. However, it is possible to perform the installation using a CD-ROM mounted on another computer that supports the `rsh` protocol.

⇒ First, choose what type of `disksetup`:

```
Automatic or Custom disk setup? [automatic]
```

Selecting ***automatic*** directs the setup program to use the same disk configuration method used by the Express Installation (see previous section). Selecting ***custom*** will direct the setup program to use the disk configuration method described in the following section.

⇒ Next, choose the way software is selected:

```
Use Automatic or Custom software selection? [automatic]
```

Selecting ***automatic*** directs the setup program to install the most common and desired software automatically. This is preferred for the majority of installations. If you select ***custom***, you will be prompted to select which packages to install. Users installing on disks with a `/usr` partition smaller than 360 MB should use this option.

Custom Installation: Setting Up for Local CD-ROM Loading

Skip this section if loading from a CD-ROM located on a remote machine.

⇒ Load the BSD/OS 4.0 BINARY CD-ROM into the local CD-ROM drive. Skip the next section.

Custom Installation: Setting Up for Remote CD-ROM Loading

Skip this section if loading from a CD-ROM on the local machine.

⇒ For remote CD-ROM access, mount the BSD/OS 4.0 BINARY CD-ROM on the remote computer using its standard `mount` command. Remember the name of the directory upon which the CD-ROM filesystem is mounted.

If BSD/OS finds only a single network card in the system, it will automatically be selected. If more than one network card is found, the Installation script will display the list of available cards.

⇒ If multiple cards are displayed, select the card that can communicate with the remote computer on which the CD-ROM was mounted.

Once the interface is selected, the Installation script will prompt for the following information:

⇒ The IP address of this computer.
⇒ The netmask for this network.
⇒ The IP address of the remote computer.
⇒ The name of a user account on the remote computer that allows rsh requests from the `root` account on the local computer.

⇨ The directory on the remote machine on which the BSD/OS 4.0 BINARY CD-ROM is mounted.

⇨ The IP address of a gateway machine, if one exists. If the remote machine is on the same local network, no gateway address will be needed.

The Installation script will then attempt to contact the remote computer and find the CD-ROM. If something goes wrong, an error will be reported to you and the above questions will be asked again. This process will repeat until a CD-ROM is found.

Custom Installation: Disk Configuration

The Custom Installation procedure will list all disks found on the system. For each disk, it will ask you if this disk should be configured. Custom Installation configures disks by calling the `disksetup`(8) program in interactive mode. This mode assists you in setting the geometry for the disk, partitioning the disk, for both BSD/OS and potentially other operating systems, setting up the BSD/OS partitions, and installing boot code for BSD/OS. Note that you should not use the default configuration for disks other than the primary disk, as the default configuration will create root and `/usr` filesystems on multiple disks (which is undesirable). Each filesystem partition should have a unique pathname. See the `disksetup`(8) manual page located in the reference section of this manual for detailed information on configuring disks.

Successful installation normally requires at least the `/` (root) and `/usr` partitions. The `/` (root) partition is nearly always located on the boot drive of the computer. (The BIOS always boots from the "`C:`" drive. There are two possible ways to configure that drive to run BSD/OS from another drive. One method is to use `bootany` on the primary drive to boot a partition on the "`D:`" drive; see `bootany`(8) for the procedure. It is also possible to boot from a drive not known to the BIOS by placing a small pseudo-root partition on the boot drive. This partition requires only two files, `/boot` and a `/etc/boot.default` file that contains a command to boot from a root partition on another drive; see `boot`(8) for details.)

Once all disks have been configured, the Custom Installation procedure will create a `/etc/fstab` filesystem configuration file for the system.

Custom Installation will also ask you to specify a size, in megabytes, for a memory-based (MFS) `/tmp` filesystem. Using an MFS will improve the overall performance of your system (unless you are very short on RAM), and it is recommended if you have at least 16 MB of RAM. Answer with `0` in the unlikely event you do not want `/tmp` to be configured as an MFS.

If you later determine that you desire a larger MFS, change its `-s` option in `/etc/fstab` and reboot.

⇨ Choose a size for your MFS `/tmp` filesystem (try 16 MB if you have no better guess).

If the location of the `/` and `/usr` partitions cannot be automatically determined, the Custom Installation procedure will prompt for the name of an editor to run with a prototype `/etc/fstab` file (i.e., editors like `pico` or `vi`).

→ If asked, choose an editor.

→ When asked, supply correct entries for `/` and `/usr` then write the file and exit the editor.

For each of the wd (IDE or ESDI) drives on the system, the Custom Installation procedure will check for the existence of a bad sector table using `diskdefect`(8). If one does not exist, it will ask if you want the disk be scanned for bad blocks. Modern IDE drives provide “perfect media” and do not need to be scanned. Older IDE drives and all ESDI drives should be scanned.

→ Answer ***`yes`*** or ***`no`*** appropriately.

The Custom Installation procedure will then check or create the filesystems on the disks. It will first ask if `all`, `some`, or `none` of the filesystems should be initialized.

→ If there are no special system requirements and all the filesystems are new, respond with ***`all`***. This will cause the command

`newfs /dev/r`*disk*

to be executed for each filesystem, where *disk* is replaced with the appropriate entry, e.g., sd0a.

→ If some or all of the filesystems need special attention (e.g., if their contents are to be preserved), answer ***`some`***. In this case, Custom Installation will prompt for each of the filesystems.

→ For each filesystem prompt, to execute the `newfs` command, answer ***`yes`***. Answer ***`no`*** if that filesystem’s contents are to be preserved. For special handling, respond with ***`!shell`*** and at the command line prompt execute the appropriate `newfs` command (or `fsck` if the file system already exists).

→ If you do initialize the filesystem by hand by using ***`!shell`*** to request a command prompt, answer ***`no`*** to the subsequent question of whether the filesystem should be automatically initialized. Note that swap partitions should never be initialized.

The final option, ***`none`***, should only be used if the filesystems were all pre-existing or if you used the ***`!shell`*** command to initialize all of the filesystems by hand.

Once all the file systems have been initialized, the filesystems will be mounted and swap space (from the drive that hosts the root partition) will be enabled. If there is no `/var` filesystem, the Custom Installation procedure will create a `/usr/var` directory and symlink it to `/var`, although this can be overridden by specifying a different path name to the question

```
Where should the /var data be placed? [/usr/var]
```

Once the disks have been set up, the following actions will be performed:

- The directory `/var/db` will be created.
- The license key, if any, will be written to the file `/etc/license`.
- The appropriate keyboard map, as selected, will be written to `/etc/keyboard.map`.
- The file `/etc/fstab` is copied into place.
- The kernel `bsd.gz` from the cdrom is installed as `/bsd`.
- The `/boot` file on the cdrom is installed on the hard disk.
- The `/etc/boot.default` file is created with the lines:

  ```
  -echooff
  -autodebug -q
  ```

  If the installation is performed from a serial console, the following lines are included as well:

  ```
  -echo booting from serial console com0
  -console com0
  ```

  If the root device is on removable media, the following lines are included as well:

  ```
  -echo booting from removable media (sr0)
  -rootdev sr(0,0)
  ```

  If the installation was performed on a machine that had a supported PCMCIA controller, the following lines are included as well:

  ```
  # some notebooks need wait states inserted.
  -echo setting up pcmcia services
  -pccard_wait on
  # Uncomment the following two lines to enable apm
  #-echo setting up for apm
  #-apm
  ```

  You might wish to uncomment the `-apm` line to enable advanced power management, if the machine supports it, though in rare circumstances some machines have had problems with this and will not boot if apm is enabled.

  See the `boot`(8) manual page for more information.

The installation procedure is now ready to run the `installsw` program. The required packages **must** be installed, and most of the desirable packages should be installed as well. The `?` command to `installsw` displays information about a package and can be used to decide whether to install that package.

⇒ Answer the `installsw` questions appropriately.

Once the the packages have been installed, the installation procedure will request the root password:

```
Changing local password for root.
```

```
New password:
```

Enter the password for the root account on this machine and press `<Enter>`. The password will not be displayed as it is entered. The `passwd` program will prompt a second time to verify that it was not mistyped. If the first and second entries do not match, it will once again ask for the password. This repeats until the same password is entered twice.

The installation is now complete. You will be asked to remove the INSTALL floppy and reboot the machine.

⇨ Remove the floppy and reboot the machine.

Proceed to the section named *Configuring Your BSDI System*.

Upgrading BSD/OS

Skip this section if you are installing BSD/OS from scratch.

Upgrading BSD/OS to version 4.0 from version 3.0 or 3.1 is a simple process. This chapter describes the upgrade procedure. This procedure works *only* with systems running BSD/OS 3.0 or 3.1; the procedure will refuse to operate on any other release. **You should read this entire section, including the compatibility issues, before beginning an upgrade.**

First: **backup all your data**. While our testers have never lost data when upgrading BSD/OS V (to our knowledge), it is only prudent to be prepared. In particular, it is strongly recommended that a separate copy of the `/etc` directory be made, as well as copies of any other configuration files you might have (e.g., `/var/www/*`) and mailboxes (`/var/mail/*`), even though the upgrade script should cause no harm to these files. As all binaries will be upgraded, any locally-modified utilities in standard locations (like `/usr/bin`) will be replaced. You should also make sure that there is at least 3 MB of free space on the root filesystem.

A few of the packages in BSD/OS have dramatically changed in size since the 3.0/3.1 releases. In general, if you are not installing any of these packages and you have 15 MB free in the `/usr` filesystem, you should have more than enough disk space for upgrading. If you are installing any of the following packages, you should allow additional space:

Package Name	Increase in size
Misc. additional binaries	+ 10.1 MB
Developer	+ 4.1 MB
Supplementary Manuals	+ 2.1 MB
Manual Page Sources	+ 1.2 MB
Networking	+ 4.5 MB
Kernel Objects	+ 7.3 MB
TeX	+ 21.1 MB
Core X11 Clients	+ 10.2 MB
X11 Developer	+ 3.1 MB
X11 Server	+ 43.8 MB

Note that if you have removed portions of the 3.0/3.1 packages, they will be re-installed and thus use more disk space. If you have installed non-standard programs in places like `/bin` or `/usr/bin`, they will be replaced and your disk space requirements will change. The `installsw` program will help you keep track of disk space you'll need.

Help! I don't have enough disk space!

If your `/usr` filesystem does not have enough space for you to upgrade, you have several options:

- Backup all disk drives; reinitialize the disks; reload the files
- Remove packages that you never use

- 'Move' one of the following three large directories to some other disk partition and then use a symbolic link to connect the two pieces:
  - /usr/contrib
  - /usr/X11R6
  - /usr/src

  Moving any other directory from `/usr` is liable to break lots of things. Don't move any other directory off the `/usr` filesystem!

The BSDI service department can help you with these items.

Upgrade Paths

There are two possible upgrade paths: Express Upgrade and Custom Upgrade. Both support upgrade from local or remote CD-ROMs. Of course, using a local CD-ROM is simpler and easier.

Use the Express Upgrade when you want to upgrade the existing packages on the system and you do not wish to add any additional packages. Otherwise, use the custom upgrade procedure.

Use the "local" upgrade when you have a local CD-ROM drive; otherwise use the "remote" upgrade.

⇒ The first step in upgrading the system is to shut down the computer to single user mode. After informing your users, enter this:

```
# shutdown now "Upgrade to 4.0"
```

This will shut down the system immediately and logoff any currently logged in users.

⇒ If this is to be a local upgrade, mount the BSD/OS 4.0 BINARY CD-ROM on the directory `/cdrom`. If this is a remote upgrade, insert the BSD/OS 4.0 BINARY CD-ROM on the remote machine and mount it on a directory; be sure to remember the name of the directory.

⇒ Insert the BSD/OS 4.0 INSTALL floppy diskette into the `A:` drive and execute the command:

```
# installfloppy
```

You will be presented with several screens of information. Read these screens carefully and continue only when you are sure the requirements presented have been met and you want to perform the actions described.

The system will install some preliminary pieces and a new kernel and will then prompt you to reboot. After the reboot, the upgrade procedure will continue.

⇒ Choose either the "Express" or "Custom" Upgrade. Choose "Express" if you want the upgrade procedure to choose automatically which packages to upgrade. If you choose the Custom path you should stop for a moment and read the "DESCRIPTION" section of the `installsw`(8) manual page located in the reference section of this manual. Using the Custom upgrade option, `installsw` automatically marks the packages that appear to be installed from a previous release, but these selections can be changed.

⇒ Choose "remote" or "local" upgrades. Choose local if the CD-ROM is mounted on the system being upgraded; choose remote if the CD-ROM is mounted on a remote machine. If remote is chosen you will need to provide:
   - The name or IP address of the remote machine.
   - The login name of a user on the remote computer that allows rsh commands as root from the computer being installed.
   - The directory path on which the remote CD-ROM is mounted.

   Once the CD-ROM is found, the preliminary UPGRADE package will be extracted. This includes the latest versions of the BSD/OS 4.0 shared libraries and new versions of the `installsw`, `ifconfig`, `route`, and `pax` commands. The 4.0 shared libraries will be installed in `/shlib`, and if any of the 3.0/3.1 shared libraries in `/shlib` is not the latest version, all outdated libraries will be upgraded.

   The system will install a new kernel from the CD-ROM and will prompt you to reboot. Once the system reboots, the upgrade procedure will continue.

   The Upgrade Installation procedure will now run the `installsw` command in Upgrade mode. As part of installing the upgraded software, a number of configuration files, generally found in `/etc`, will be modified. If, because of local modifications to those files, the file cannot be automatically upgraded, a new version of the file will be installed with a `.bsdi` suffix, e.g., `/etc/rc.bsdi`. You will be informed whenever this happens, and once `installsw` has been run, you should merge the two versions of the file by hand. Failure to do so can result in BSD/OS being unable to reboot properly and complete the upgrade procedure.

It is important to realize that any individual package installation might cause files already on disk to be removed when you add the new package, i.e., files from a previous version of a package that are not used by the current version of the package will be removed during installation.

Your upgrade is now almost complete. The system will need to `reboot` once again so that it's running with all the new binaries and libraries. When `fsck` runs after the system is rebooted, you might see comments about unreferenced files that are being removed from your disk since programs like `init` that were running during the installation have been replaced. In the case of this reboot, these messages are normal and no cause for concern.

Your upgrade is now complete.

Hint for Easing Future Updates

When you first install a 4.0 system, the installation procedure creates the file `/.base-4.0` before changing anything. Thus, it is possible to generate a list of every file on the system changed since the installation by running the command:

```
 # find -x / /usr -newer /.base-4.0 -print
```

This is useful not only for performing upgrades to future versions of the BSD/OS software, but also for generating your own install procedures for large sites. As previous releases of BSD/OS generated a file named /.base-<version>, you can use the file as described above to determine if there are configuration files on your system that should be saved before you install the new release.

The Disk Clean Flag

BSD/OS supports the filesystem "clean flag". When a filesystem is unmounted or the system is shut down properly, the filesystems are marked 'clean.' When the system reboots, 'clean' filesystems do not require a filesystem check. If the filesystem is not clean, it should be (and, during reboot, is automatically) checked with `fsck`(8) before the filesystem is mounted (or, in the case of the root filesystem, before the filesystem is mounted read-write). The `fsck`(8) program will mark filesystems clean once they have been checked. It is possible to force a check of a clean filesystem (for example, if you suspect disk problems) by specifying the -f option to fsck.

A filesystem that is not marked clean cannot be mounted read-write, although it can be mounted read-only. The system prints a warning message when a dirty filesystem is mounted read-only or when an attempt is made to mount a dirty filesystem read-write. In an emergency, it is possible to mount a dirty filesystem read-write by mounting it read-only and then upgrading to read-write:

```
# mount -r /dev/xxxx /path
# mount -uw /path
```

Note that modifications to a dirty filesystem can corrupt the filesystem and cause system crashes. The system panics if an inconsistency is found in a filesystem mounted read-write so that the problem will be corrected during the filesystem check after reboot. However, inconsistencies found in a read-only filesystem are simply logged.

BSD/OS 4.0 Compatibility Issues

This section describes changes in BSD/OS 4.0 that affect compatibility with previous releases.

ELF

In BSD/OS 4.0, the default executable binary file format and core file format is ELF. Programs will fail if they assume that executable binary files must be either a.out or COFF format. ELF binaries, object files, core files and libraries built under BSD/OS 4.0 will not work on earlier versions of BSD/OS.

The system loads ELF binaries at different addresses from a.out binaries. Programs that make assumptions about the order or addressing of the text, data or stack segments may fail.

The following BSD and GNU programs have been replaced by newer GNU programs that support ELF:

- cc (GCC 1)
- ar
- as
- gprof
- ld
- ranlib

These utilities are mostly compatible with previous versions of BSD/OS, but there are some differences. Check the manual pages for details. In particular, `ld`(1) is now much more complex. Note that `cc`(1) now produces dynamically-linked binaries by default, rather than statically-linked, unshared binaries.

Also, the input language that is accepted by `as`(1) is now ELF assembly language. Many, or most, old assembly language files will not build correctly under the new version of `as`(1), although the differences in usage are relatively small. We distribute the old assembler as `/usr/old/as`. The a.out object files that it produces may be linked into ELF programs.

See the *Programmer's Notes* section for information on changes that are due to ELF.

Netstat

The output format of the `netstat`(8) program has changed radically:

- **`netstat -i`**
  now omits statistics.
- **`netstat -is`**
  displays statistics, and omits network and MTU information. It looks more similar to the old **‘`netstat -i`’**, while
- **`netstat -isv`**
  now looks like the old **‘`netstat -is`’**.

- **`netstat -r`**
  display has changed to omit `References` and `Use` fields and include the `MTU` field.

The **`-O`** flag provides compatibility with the old flags and display formats. See the `netstat`(8) manual page for details.

Network Interface Ioctls

The following `ioctl`(2)'s will no longer work in a 4.0 system unless the kernel is rebuilt from source with **`'options COMPAT_IFIOCTL'`**:

SIOCSIFADDR
SIOCSIFBRDADDR
SIOCSIFDSTADDR
SIOCSIFNETMASK

The SIOCAIFADDR and SIOCDIFADDR ioctls should be used instead. Also, the following `ioctl`(2)'s are deprecated:

SIOCGIFADDR
SIOCGIFBRDADDR
SIOCGIFDSTADDR
SIOCGIFNETMASK

Use `getifaddrs`(3) instead.

Ftpd

The standard ftpd provided with BSD/OS 4.0 is completely different from previous versions of ftpd. See the section *Configuring Anonymous FTP* and `ftpd`(8) for details.

Directory Linking

It is no longer possible to add or remove hard links to directories using `link`(2) or `unlink`(2), respectively. Only `mkdir`(2) and `rmdir`(2) adjust hard links to directories.

Automatic Disk Configuration With disksetup

Skip this section if you have already set up your disks or don't care to learn the internals of BSDI's filesystem configuration scheme.

Although `disksetup`(8) attempts to be self explanatory (via prompts and help screens), it is useful to understand the basics of how the interactive mode works and the goals for configuring disks.

Hints for Using disksetup

When requesting information about a disk's geometry, or when editing FDISK and BSD/OS partition tables, `disksetup` uses a full screen display/edit mode. Use the <Tab> key to move among fields. Generally the <Esc> key will cancel the current changes. When editing the partition tables, press the "?" key for context-sensitive help. You might want to read through these help screens as they contain other valuable tips on using `disksetup`.

Goals in Setting Up a Disk

The two goals of `disksetup` are:

- To partition a disk into filesystems for use by BSD/OS.
- If necessary, to partition the disk into logical disks to enable BSD/OS to be co-resident with other operating systems (i.e., share a hard disk with other operating systems).

Since co-residency affects nearly every part of partitioning a disk, it is the first question that is asked. Setting up disks for co-residency is more challenging than dedicating them entirely to BSD/OS.

If you are not going to use co-residency, you might want to skip the next section and continue on to *Selecting Geometries*. References to the FDISK partition table can generally be ignored in this case. An FDISK partition table is created with a single BSD/OS entry for the entire disk.

Goals of Co-residency

Skip this section if you do not wish to share a single disk between BSD/OS and another operating system.

When a disk is to be set up for co-residency, an FDISK partition table (read by DOS and others) is created to divide the disk into sections. The FDISK partition table might describe up to four partitions, each of which is treated as a logical disk. In order to install BSD/OS, a partition must be assigned to BSD/OS. Typically there is no need to create more than a single BSD/OS FDISK partition. The `disksetup` utility eases the creation of the FDISK table when the choice is to install DOS and BSD/OS. When you assign some space for DOS, BSD/OS will be assigned the rest of the space. Share a swap partition between DOS and BSD/OS by assigning some space when asked. See *Sharing a Swap Partition* below for more information on this feature.

Before partitioning the disk, you should know approximately how much space you will be assigning to each FDISK partition, and you should ensure there is enough space assigned to BSD/OS for installation of the software. The amount of space required by BSD/OS can range from under 100 MB to many hundreds of megabytes depending on the features selected. Typical systems require 350 MB for the BSDI software plus whatever space users require for their files.

BSD/OS is most often placed at the end of the disk. This is convenient for leaving space for the bad sector table on non-SCSI disks. Non-SCSI disks store the bad sector table at the end of the disk and require 126 sectors plus the number of sectors in one track. This is typically reserved automatically by `disksetup`.

The standard Master Boot Record (MBR) found on most systems will always boot the operating system marked "active" in the FDISK partition table. Although you may use the `bootany`(8) command to alter the active partition from BSD/OS, it is typically more convenient to use the MBR provided with BSD/OS called `bootany.sys`. The `bootany.sys` MBR, which `disksetup` installs by calling the `bootany` utility, enables booting of up to four operating systems from either of the disks recognized by the BIOS (often called `C:` and `D:`). During the standard installation process, you will only have the option of booting from the first disk (`C:`). See the `bootany` manual page (at the end of this document) for more information on booting from the second drive.

Selecting Geometries

Due to the complexity of modern drives (particularly SCSI and large IDE drives) and DOS's initial design for smaller disks, the geometry specified for those disks (number of heads, sectors per track, and total number of cylinders) is often largely fictitious (though the total capacity will match). To calcualte the total number of sectors on you disk, use the following formula:

```
  total-sectors = heads * sectors-per-track * cylinders
```

Also note that:

```
  heads = tracks-per-cylinder
```

When used with co-residency, `disksetup` operates with two potentially different geometries. One is used for producing the FDISK table while the other is used when describing BSD/OS partitions. Note that the FDISK table can only describe partitions contained within the first 1024 cylinders. The BSD/OS geometry does not have this limitation. When `disksetup` first examines a disk, it determines up to four geometries associated with a disk. Some of these will be used in producing an FDISK partition table, and others might be used for the BSD/OS geometry:

- **CMOS:** This geometry is reported by the INT41 and INT46 interrupt vectors. This is assumed to be the same geometry that is displayed under the CMOS setup screen for the drive. Not all disks will have this geometry reported. Normally, only IDE, ESDI, and similar disks have this geometry, and only if enabled in the CMOS setup screen. The data displayed might have been slightly altered by the BSD/OS boot straps in order to reduce the number of heads to a value of 16 or less. IDE drives that claim to have more than 16 heads are considered "mapped" and BSD/OS will report the unmapped geometry instead.

- **BIOS:** This geometry is reported by the BIOS and is used by DOS and any other operating system that uses the BIOS to access the disk. The FDISK partition table is almost always expressed in terms of this geometry. Up to two disks may report BIOS geometries. The BIOS geometry does not always agree with the CMOS geometry. SCSI disks with no CMOS geometry may have a BIOS geometry if the SCSI adapter BIOS "attaches" them as the `C:` or `D:` disk.
- **FDISK:** This geometry is derived from an existing FDISK table. The total number of cylinders might be less than the actual number of cylinders the drive contains. This geometry is only defined if there was a pre-existing FDISK table on the disk. Due to various work-arounds that different OS's use to handle large disks, it might not always be possible to determine a valid geometry from the FDISK table. This should not cause problems for `disksetup` as it will assume the BIOS geometry is correct and use that regardless of what the FDISK geometry claims to be.
- **BSD/OS:** This geometry is retrieved from a previous BSD/OS label on this drive. This geometry is only defined if there was pre-existing BSD/OS label on the disk.

When determining the geometry to be used by BSD/OS, several mechanisms are available:

- **Probe:** Try to determine the geometry by probing the actual disk. On SCSI disks that are zone recorded (the size of a cylinder varies over the disk), this will always return 2048 sectors per track, one head.

  Probing of large disks might take many minutes, so don't be in too much of a hurry to reset the machine.
- **Internal:** Use the geometry that BSD/OS is currently using for the disk. This geometry might not be found on the disk and might be grossly inaccurate on a new disk.
- **File:** Use the geometry described by an ASCII disk label previously produced by the `disksetup` program.
- **Disktab:** Use the geometry found by searching for a name in the `/etc/disktab` file.
- **Manually:** Enter the geometry by hand. This is often the choice used for more troublesome drives.

The geometry used by BSD/OS should almost always be the geometry determined by probing the disk. Please see the warning above on probing disks. The BSD/OS geometry must not contain more than 16 heads on IDE or ESDI disks. A mapped geometry with fewer heads must be selected in cases where the other geometries specify more than 16 heads. The BSD/OS bootstraps will identify this situation, if possible, and adjust the CMOS geometry for the disk to be a usable geometry.

Note that the FDISK table will be examined by non-BSD/OS systems that use the BIOS to access the disk. This means that the disk geometry used (number of heads, sectors per track, and cylinders) by the BIOS must be the same as that used by `disksetup` when building this table.

Goals of BSD/OS Partitioning

Once the FDISK partitions have been assigned (or if the entire disk is to be used by BSD/OS), the space assigned to BSD/OS must then be partitioned into filesystems. The `disksetup` utility can automatically split the disk into the required partitions for installation. These partitions are the `/` (root) partition, the swap partition, and the `/usr` partition.

- **`/`:** The `/` (root) partition contains the operating system, some configuration files, and the programs required to boot the system. While this can be made as small as 5 MB or 6 MB for special small configurations, 10 MB is a more common value (it eases future upgrades). This partition might need to be made larger to accommodate large password or other configuration files. This partition should always be at the start of the disk or BSD/OS FDISK partition.

  This partition is small so it's easy to perform full backups of `/` every night; this in turn makes recovery from disk crashes less painful in the event that something goes wrong. As almost all the configuration files are kept on the root partition (including the password file), full root backups should be performed frequently.
- **swap:** The `swap` partition is used when the virtual memory system exhausts real memory (RAM). The amount of swap space needed can vary widely, based on use of the machine. Typically a minimum of 48 MB will be sufficient, though the X Window System and C++ compiler can consume large amounts of space. It is always recommended that swap be at least as large as main memory so that in the unfortunate event that the system panics, a copy of main memory can be saved in the swap partition for retrieval (for diagnostic purposes) when the system is brought back up.
- **`/usr`:** The `/usr` partition contains the bulk of the system programs and data used in normal system operations. `/usr` is designed so that it can be shared among machines (via NFS), although in the standard system it is assigned all the disk space not used by root or swap. You will need to split up the `/usr` partition if you plan to share a single copy of it among several BSD/OS systems. (See below.)

  `/usr` needs to be between 200 MB and 300 MB depending on the features installed.

While `/`, swap, and `/usr` are the standard partitions, many sites choose to have more. Common examples are:

- **`/var`:** Used to contain files specific to this machine, such as log files and spool files. Files in the /var system are expected to grow and shrink over time (e.g., user's mailboxes). In the standard setup, /var is actually a symbolic link to /usr/var and these files are stored there.

BSD/OS requires at least 4 MB for this partition. However, once you add /var/tmp (for editor files), /var/mail (for email files), and /var/spool (for mail, printer, and uucp queues), you can easily push the requirements up to 50 MB or beyond.

Don't let /var run low on disk space; many system functions require disk space in /var to maintain normal operation. Be sure there is at least 10 MB of "breathing room" on the /var partition.

- **/usr/src:** Since sources take up so much space, often they are placed in their own partition. One common approach is to mount /usr/src from the CD-ROM at this location in the filesystem tree.
- **/usr/obj:** During a build from BSD/OS sources, object files are normally placed in the /usr/obj directory. Creating a separate partition keeps these files from cluttering the /usr partition.
- **/usr/home:** By default, user accounts are created in /usr/home. To separate user accounts from the rest of the system, a separate partition may be created for them. Using /usr/home is not a hard and fast rule. Other possibilities often seen are /home, /home/your_hostname, /u1, and /u2 (when using multiple filesystems for home directories).
- **/usr/local:** There are often local customizations to a system. Upgrades are simplified if changes are concentrated into as few places as possible. /usr/local need not be an actual filesystem but could be a symbolic link to some other directory (maybe one that is not on the /usr filesystem). The size of the /usr/local partition will vary with your site, of course.
- **/var/news:** Usenet news tends to create a very large number of smaller-than-average files. A full news feed can consume an enormous amount of disk space (not to mention network bandwidth). You will probably need to alter the default newfs(8) parameters when creating the filesystem to get the best results.

  By default, newfs allocates one inode for every four disk fragments. A fragment is the smallest unit of disk allocation while a block is the largest unit of I/O. The default is 1K/8K, but for news filesystems you probably want to use 512/4K.

  One inode is required per allocated file; when you run out of inodes, it's like running out of disk space (only it's more annoying because you have disk space left). For filesystems containing news, the average file size is only about 3K so you will need to increase the total number of inodes available (by decreasing the bytes per inode value) or else you will probably run out of inodes.

  The following command will create a filesystem with a fragment size of 512 bytes and allocate an inode for each 3072 (3K) bytes of data on the filesystem. This should be reasonable for most news partitions:

```
# newfs -f 512 -b 4096 -i 3072 /dev/rsd0e
```

Of course, you'll want to substitute the correct disk partition name.

- **`/tmp:`** Machines with at least 16 MB of RAM probably want to use a memory based filesystem (MFS) for `/tmp`. See `mfs`(8). Using an MFS for `/tmp` can improve system performance since it speeds access for the many small files that are created and destroyed during normal system operation. You will want to allocate extra `swap` space if you plan to use an MFS.

  More active systems will want `/tmp` to be at least 8 MB (more likely 16 MB) and hence will most likely not want `/tmp` to be on the root partition (where it is by default). Smaller systems, such single user systems with small memory, can usually leave `/tmp` on the root filesystem if there is enough free space.

  It should always be assumed that all the data in `/tmp` will be lost across reboots. Do not store important files there!

There are a few conventions that should always be followed when setting up BSD/OS partitions. The 'a' partition should always be the / (root) filesystem. The 'b' partition should always be a `swap` partition. The 'c' partition should always reflect the entire disk, even when using co-residency. If you wish to use the entire disk as a single partition, do not directly mount the 'c' partition (except on SCSI disks where it's OK). Rather, create a 'd' partition that spans the disk (excluding the bad sector area) and use it. It is important that the bad sector area not be included in any partition other than the 'c' partition. Normally, `disksetup` will not allow you to assign such partitions.

The remaining BSD/OS partitions are named 'e', 'f', 'g', and 'h'.

If you are using co-residency, you might find it convenient to assign BSD/OS partition letters to some of your FDISK partitions. This is very useful when the FDISK partition represents a DOS FAT filesystem that BSD/OS can mount. You may do this by using the `[I]mport` command when assigning BSD/OS partitions. By convention, these partitions start with the 'd' partition and number upwards.

Note that extended DOS partitions are actually like separate disks themselves and start with an FDISK partition table. The `disksetup` utility offers no way to edit this recursive table (use DOS's FDISK command for that). Since the typical scenario is to have a single DOS partition inside the extended DOS partition, the actual DOS filesystem will start one track beyond the start of the extended DOS partition. To do this, after you have imported the partition, decrease the size of the partition by one track by typing:

```
-1t<Enter>
```

Then use the `<tab>` key to move over to the starting sector and increase it by one track with the keystrokes:

```
+1t<Enter>
```

Sharing a Swap Partition

When using co-residency, particularly on notebooks with small disks, it is often desirable to share disk space for the MS Windows swap file and the BSD/OS swap partition. This is accomplished by creating an Extended DOS FDISK partition that is one cylinder larger than the size desired for the BSD/OS swap partition. When assigning BSD/OS partitions, the b partition should be imported from the Extended DOS FDISK partition. The start of the b partition should be one cylinder in from the start of the Extended DOS FDISK partition. You will need to use the DOS FDISK utility to install and then format the Extended DOS partition. You should never create a subdirectory in this DOS partition as BSD/OS will likely destroy it when it uses the swap space. Any file put in this DOS partition will likely be destroyed by BSD/OS. You should, however, assign a temporary swap file in this DOS partition for use by Windows. You may assign all the available space to the Windows temporary swap file.

Sample Partitioning Screens

Partition screens will have a one line map of the disk displayed. For example, when setting up the FDISK partitioning you will see a line such as:

```
|.1------2-------3-----------------------------------------------|
```

The “|” characters represent the start and end of the disk. Free space is represented by the “.” character. Space assigned to a partition starts with a digit, such as “1” and is optionally followed by one or more “-” characters. The above line has a bit of free disk space at the front of the disk followed by two smallish partitions with a third large partition.

When setting up the BSD/OS partitioning you will see a line similar to the following:

```
|.d------2b-----a-h--------------------------------------------|
```

In addition to the FDISK partitioning information, the BSD/OS partitioning information is overlaid on top. BSD/OS partitions are represented by a single character between a and h, inclusive. Again, they might optionally be followed by one or more – characters. The above line shows a slight amount of free space at the start of the disk. The d partition is most likely an imported DOS partition. The b partition (swap) starts slightly into the second FDISK partition. (This is what a shared swap partition would look like.) The a partition (root) uses a small amount of disk space and the h partition (`/usr`) consumes the remaining part of the disk.

The following sample screen shows a 500 MB disk partitioned by `disksetup` using the defaults and using the recommended shared swap partition.

```
    FDISK Partitioning                             wd0 Device to Partition
                                                    63 Sectors/Track
    <?> for help                                  1008 Sectors/Cylinder
                                                    16 Heads
                                                  1015 Cylinders
 |.1------2-------3-------------------------------------------------------|
    |    Type of    |  Length of Partition in   | Starting|  Ending| Sector
 P# |    Partition  |  Sectors ( MBytes    Cyls)|   Sector|  Sector|    Gap
 ---|---------------|--------------------------|---------|--------|-------
 1  | 01 DOS-FAT12  |    81585 (   39.8    80.9)|       63|   81647|
 2  | 05 DOS-EXTEND |    97776 (   47.7    97.0)|    81648|  179423|
 3* | 9F BSD/OS     |   844515 (  412.4   837.8)|   179424| 1023938|
    |               |                          |         |        |
    |  Size of Disk |  1023939 (  500.0  1015.8)|        0| 1023939|
 ---|---------------|--------------------------|---------|--------|-------

    At this point you should define, alter, or delete the partitions for
    this disk so that you have enough space reserved for each of your
    operating systems. Press N when you are done or X if you wish to cancel.

    If you do not understand this screen, press X and then Y to confirm
    the cancel.  When asked if you want the standard setup, answer YES.

    [A]dd [D]elete [E]dit [N]ext [X]Abort [*]Mark Active [?]Help
```

Sample BSD/OS Partitioning with above FDISK Partitioning

Using the above FDISK partitioning, the following screen shows the standard BSD/OS partitioning that `disksetup` would use. Note that the swap partition is overlaid on the DOS-EXTEND partition, but starts one cylinder in. This allows preservation of the FAT used by DOS.

```
    BSD Partitioning                           wd0 Device to Partition
    <?> for help
 |.d------2b-----a-h------------------------------------------------------|
   FS    Mount     |  Length of File System in | Starting|  Ending| Sector
   Type  Point     |  Sectors ( MBytes    Cyls)|   Sector|  Sector|    Gap
 -----------------|---------------------------|---------|--------|-------
 a 4.2   /         |    20160 (    9.8    20.0)|   179424|  199583|
 b swap            |    96768 (   47.2    96.0)|    82656|  179423|
 c -----           |  1024128 (  500.1  1016.0)|        0| 1024127|
 d msdos /dos      |    81585 (   39.8    80.9)|       63|   81647|
 e -----           |                           |         |        |
 f -----           |                           |         |        |
 g -----           |                           |         |        |
 h 4.2   /usr      |   824355 (  402.5   817.8)|   199584| 1023938|
 -----------------|---------------------------|---------|--------|-------

    At this point you should define, alter, or delete the file systems.
    BSD/OS requires at least a root filesystem (/), a swap partition and
    a /usr filesystem.

    If you do not understand this screen, press X and then Y to confirm
    the cancel.  When asked if you want the standard setup, answer YES.

    [I]mport [A]dd [D]elete [E]dit [N]ext [X]Abort [?]Help
```

Sample BSD/OS Partitioning Using the Whole Disk

The following screen displays the standard partitioning of a 500 MB disk when the entire disk is used by BSD/OS. Note that final BSD/OS partition (partition h) does not use the final 189 sectors of the disk. This is to leave space for the bad sector table of 126 sectors plus the size of one track, 63 sectors in this sample.

```
    BSD Partitioning                         wd0 Device to Partition
    <?> for help
|a-b-------h-------------------------------------------------------------|
  FS    Mount     |  Length of File System in | Starting|  Ending | Sector
  Type  Point     |  Sectors ( MBytes    Cyls)|   Sector|  Sector |    Gap
------------------|---------------------------|---------|---------|-------
a 4.2   /         |    20160 (     9.8    20.0)|        0|   20159 |
b swap  swap      |    98784 (    48.2    98.0)|    20160|  118943 |
c -----           |  1024128 (   500.1  1016.0)|        0| 1024127 |
d -----           |                           |         |         |
e -----           |                           |         |         |
f -----           |                           |         |         |
g -----           |                           |         |         |
h 4.2   /usr      |   904995 (   441.9   897.8)|   118944| 1023938 |
------------------|---------------------------|---------|---------|-------

    At this point you should define, alter, or delete the file systems.
    BSD/OS requires at least a root filesystem (/), a swap partition and
    a /usr filesystem.

    If you do not understand this screen, press X and then Y to confirm
    the cancel.  When asked if you want the standard setup, answer YES.

    [I]mport [A]dd [D]elete [E]dit [N]ext [X]Abort [?]Help
```

Configuring Your BSDI System

Configuring BSD/OS 4.0 is the same for both Express and Custom Installations. The first time the system boots BSD/OS, you will be prompted to answer several configuration questions. You will only be asked these questions the first time the system boots. Each of these questions will be preceded by an explanation of what information is expected.

The goal of the initial system configuration phase is to configure the system's time-of-day, network connection, Domain Name System and license information. In addition, you will create an administrative user account.

The first menu gives you the three choices for system configuration:

- MaxIM, which runs under the X Window System and uses a web-based interface to help you configure your system. This option requires that you have a VGA-capable display and a mouse on the machine being configured. You can run MaxIM at any time by running the `maxim`(8) command explicitly.
- Manual configuration, which helps you configure the basic information (license key, timezone, and basic networking) but runs without requiring you to configure the X Window System.
- No configuration assistance (for experts only).

If you select MaxIM, the first thing you will do is configure the X Window System. If you know the kind of video card and mouse you have, X configuration is easy. Otherwise, it's a trial-and-error process. The configuration program will cycle you through configuring X and then checking to see if your configuration worked before moving on.

If all else fails, you can use standard 640x480 VGA mode. This is best with at least 8-bit displays; if you can only do 4-bit color you should select StaticGray visual as the default when you configure X, so you can see the different colors in your browser screen.

Once all is well with X, the system will bring up Netscape under X, using the locally running MaxIM web server.

MaxIM will prompt you for a login and password; you must login as `root` with the password you specified during system installation.

You will now be viewing the `Initialize System` page. This page helps you configure several miscellaneous system parameters. You can select `Help` at any time to get further information on the items you are configuring. When done viewing the `Help` page, select `Back` on the browser display to return to the page from which you came. The `Back` button can also be used at any point to retrace your steps and make new selections. Note, the initial configuration pages give you an option to `SKIP THIS STEP`. You should not skip steps unless you are sure it's okay or you have no alternative.

The first configuration item is the `Fully Qualified Hostname` of the machine you are configuring. That means you must include your domain name (e.g., "testhost.bsdi.com", not "testhost"). You can also configure a default route and set the License Key (if it was not previously set during system installation).

You should also create a system administrator's account for yourself while on this page. The account you create will be added to the `maxim` group so that you have access to MaxIM and to the `wheel` group so that you have access to the superuser account (using `su(1)`).

The next two configuration pages are for selecting your default timezone and for setting the system clock.

After you've added the administrator's account and configured these system parameters, the next step is network configuration. MaxIM will list the Ethernet, FDDI, and token ring interfaces found on this system for you to select the interface that you wish to configure. Then you should set the interface's IP Address and potentially other network options. You can use the `Back` button to revisit this page as many times as needed to configure your network interfaces (or configure only your primary interface now and configure the secondary interfaces later).

Finally, you must configure DNS for the local server. The DNS Server Configuration page gives you quick options for declaring the local machine to be a primary DNS server for your domain and for configuring a Forwarder. Of course, there should only be one primary DNS server for your domain – do not configure yourself as a primary server if someone else is already acting as the primary server.

Configuring a Forwarder means that the local machine will forward DNS requests to the machine you specify instead of attempting to resolve the name locally. This might be a requirement (e.g., you are behind a firewall and do not have direct access to the Internet), or it might be the correct choice if you have a local DNS server used by all of your site's systems. If you're not sure which configuration is right for your system, ask your local network administration people or your Internet bandwidth provider for help.

If your machine has direct internet access you can leave Forwarder blank. Of course, you can always use MaxIM to change this option later.

After the last configuration page, MaxIM will give you an option to configure more things from the main MaxIM menu or to reboot.

Once you've finished all of the configuration that you want to do during initial installation, return to the Reboot page and let MaxIM reboot your system (note that exiting the X window system will also cause the system to reboot). After the system reboots, you will get a `login:` prompt. Basic configuration is complete, your system is now up (and on the net, if you're connected) and ready to go! You can login as either `root` (superuser) or the user name you designated as the system administrator.

For information on configuring specific system services, see the chapter entitled *Configuring System Services*.

Third Party Software

BSD/OS 4.0 includes several software products from commercial vendors. Among these software products are the following:

- Netscape's Communicator web browser
- Metro-X from MetroLink, Inc.
- Novell 3.x compatible networking and file services by NetCon.

At some time after the initial release of BSD/OS 4.0, BSDI will release a third-party CD. For more detailed information on the included products and the availability of this CD, please see the World Wide Web page:

```
http://www.bsdi.com/products/internet/third-party
```

Netscape's Communicator web browser

Netscape's Communicator 4.04 is part of BSD/OS 4.0's core distribution. It is loaded with all installations. Netscape's World Wide Web page can be found at:

```
http://home.netscape.com
```

Metro-X

The accelerated X server from MetroLink is included. Further information can be found in *X Server Hardware*. MetroLink's World Wide Web page can be found at:

```
http://www.metrolink.com
```

Novell 3.x compatible services

Further information about NetCon's Novell 3.x compatible networking and file services can be found in *Configuring IPX/SPX Services*. NetCon's World Wide Web page can be found at:

```
http://www.netcon.com
```

Products Removed from BSD/OS 4.0

Notably missing from release of BSD/OS is the Netscape FastTrack Server. Our FastTrack compatibility status will be based on results of testing by BSDI engineers at some future date.

Backing Up Your System

If you are accessing this chapter because you are interested in backing up your system before upgrading, please read the entire chapter. You will probably want to do a full backup as described in the section *Full Daily Backups* below. You should perform the backup in single user mode. After the backup is completed, any modifications to your system will not be able to be restored from the backup.

It is important to create a recovery floppy in case of a catastophic problem. please see the chapter on `Disaster Recovery` for details on making a recovery floppy.

You should make frequent (daily) backup copies of your system's data to prevent data loss in case of catastrophe. BSDI recommends using the `dump`(8) utility and a 4mm (DAT), 8mm, or QIC tape. Some people prefer using `pax` (which supports the historic `tar`(1) and `pax` interfaces) for backups. Their respective manual pages discuss these interfaces and the mechanisms they use. However, the `dump` program provides easy incremental backups, and the corresponding `restore`(8) utility features greater flexibility for retrieval, such as an interactive mode for perusing backup tapes and restoring single files or directory hierarchies.

Dumps

There are numerous strategies for performing backups with `dump`(). Your chosen strategy depends on the nature of filesystem updates, the type of backup device, and local taste. There are at least two general strategies worth considering.

Full Daily Backups

If your system is quiescent for part of the day and you have a tape drive that holds enough data, it might be best to perform full dumps onto a single tape and then save the tapes in rotation. Here's an easy script that performs full dumps:

```
# cat /etc/dump.daily
dump 0ufB /dev/nrst0 10000000 /
dump 0ufB /dev/nrst0 10000000 /usr
  (and so on for all your filesystems)
mt -f /dev/nrst0 rewind
```

In this example, there are some magic names and numbers. The first argument (`0ufB`) tells dump four things:

- The zero (**0**) option is the dump level, indicating a *full* dump of all filesystem contents. There are other dump levels, see the `dump`(8) manual page for an explanation of the other levels.
- The **u** option tells `dump` to make a record of the dump in the `/etc/dumpdates` file. This file is human-readable and is a useful record of

the dumps that you've performed.

- The **f** option tells `dump` that you will be specifying the dump device. Since it is the first such option, the first argument following this argument is the device that will be used for the dump, in this case `/dev/nrst0`. If it does not match your system setup, you should replace it with the device name for the correct, non-rewinding tape drive on your system.
- The **B** option tells `dump` that you will be specifying the "length of the tape in 1K blocks". Since it is the second such option, the second argument following this argument specifies the tape length, in this case the number 10000000. The simplest approach to the problem of using the right value is not to attempt to figure out how much data will actually fit on your tapes, but rather to use a tape on which a full dump will easily fit and then specify a preposterous number as its length. The `dump` program responds correctly to a detected end-of-tape condition on most SCSI tape drives, but, given current tape technology, the "one dump, one tape" approach is simpler and less error prone.

Incremental Backups

Another reasonable strategy is to use periodic full dumps when the system is quiescent, ideally when running in single-user mode without system daemons, and to perform daily incremental dumps using a scheme such as that described in the `dump`(8) manual page. This scheme is better than daily full dumps if one or more of the following conditions is true:

- The system is seldom quiescent.
- Full dumps of commonly-used filesystems do not fit on a single tape.
- Daily full dumps take too long.

Ideally, filesystems being dumped should not be modified during the backup. If that is not possible, an incremental dump completes more quickly and is thus less likely to have problems due to changes during the dump. It is possible to use incremental dumps in regular files on a different disk, with periodic full dumps or higher-level incremental dumps on tape to prevent the daily backups from becoming too large.

Other Backup Strategies

BSDI recommends performing dumps on all of your commonly-modified filesystems daily, and, while less-used filesystems might need to be dumped less frequently, you should dump all of your filesystems regularly unless you are mounting them read-only or frequently verify that their contents haven't changed.

Always store your dump tapes for at least a week. Every other week or so you should take a set of dump tapes off-site and store them for months or years. (This is so that a fire in your computer room won't destroy both your machines and your backups!) In addition, you should always take the time to verify that dump tapes that you intend to store for a long period are readable, by displaying their table of contents using the `-t` option of the `restore` utility.

When establishing your `dump` regimen, be sure to check your procedures by double checking that you can restore files from the dump tapes. Carefully check your tapes (by restoring them or at least listing their file names) after the first few days as a safeguard.

Finally, always label your tapes *clearly* and *correctly*.

NOTHING can replace a good backup, except *LOTS* of work. A bad backup is rarely better than no backup at all. Make sure your backups are good!

Running Backups

In order to ensure that your backups get performed on a daily basis it is recommended that you use the cron command (see the man pages for `cron`(1) and `crontab`(5)). This will automatically run your dump script at whatever time you specify.

If you are using the `dump.daily` script above you can make an entry in the `/etc/crontab`:

```
 0 1 * * *       root /bin/sh /etc/dump.daily
```

To use the incremental backup methodology you would need a more complicated script or simply a different script for each day. On Sunday you would do a level 0 dump, Monday a level 1 and so on. To accomplish this you will need a different dump command in each script:

```
 dump 0ufB /dev/nrst0 10000000 /
```

for a level 0 dump and:

```
 dump 1ufB /dev/nrst0 10000000 /
```

for a level 1 dump and so on.

In this scenario you MUST change tapes everyday or you can easily lose data. If you are performing your backups during the night and you do not have anyone at your site on Saturday and Sunday then you might schedule dumps only for Monday through Friday - Monday's dump going to the tape inserted on Friday. Change 'time' portion of the cron entry to:

```
 0 1 * * 1-5     root /bin/sh /etc/dump.daily
```

The 1-5 specifies Monday (day 1) through Friday (day 5). Sunday can be specified as either 0 or 7.

Restores

The `restore` program will allow you to extract files archived using the `dump` program. The manual page for `restore`(8) gives plenty of information on its use. It is often useful to use the interactive mode (by using the **`i`** option) as it helps when perusing backup tapes and restoring single files or directory hierarchies.

Use `restore` to verify the contents of your backup. The **`t`** option will list the table of contents of the tape. You may wish to redirect the output of this command to a file to use for quick access to the table of contents as well as to verify a good backup.

Automating Backups

Amanda is contributed software that may be used to automate backups on one or more machines. See `amanda`(8) for more information..

Configuring System Services

Subsequent chapters describe configuration for many BSD/OS services. This chapter briefly describes these services, in the same order as the following chapters.

Adding and Removing Users

How to add and remove users from your system. BSD/OS requires that each user be "authorized" by an entry in the "password file".

Configuring Login Classes

BSD/OS 4.0 includes login classes as well as support for per-user resource management and several methods of authentication. Additional information on configuring how users access your system.

Configuring the X11 Window System

How to configure the X Window System for BSD/OS. Window systems provide a graphic user interface and multiple system windows on one console.

Configuring Serial and Parallel Devices

How to configure modems, printers and other similar devices.

Configuring Dialout IP Services (PPP and SLIP)

How to configure dialout PPP or SLIP connections to your network provider.

Configuring a Custom Network

The automatic BSD/OS configuration sets up your network for a local Ethernet with an optional default route to an external router. This chapter covers more specialized requirements, e.g., PPP, SLIP, alternate routing protocols and synchronous interfaces such as SDL communications cards.

Configuring IP Version 6

How to set up IPv6 on your system, now that BSD/OS can now provide Version 6 IP services.

Configuring the Domain Name System (BIND, DNS, named)

The automatic BSD/OS configuration sets up your initial DNS information. This chapter covers more specialized requirements, e.g., remote caches and secondary servers.

Configuring IPX/SPX Services

Configure your server to recognize traffic from the IPX/SPX network.

Configuring IP Filtering

BSD/OS IP Filters can be use to filter IP packets based on any aspect of the packet, including the full contents, the source, destination and/or return interfaces, and the mbuf flags.

Configuring Virtual Private Networks

BSD/OS 4.0 supports the ability to set up a Virtual Private Network (VPN). In BSD/OS, a VPN is the ability to set up an authenticated and/or encrypted tunnel between separate networks.

Configuring a Web Server

No special configuration is required for the standard Web clients (NetScape and NCSA Mosaic). This chapter covers more specialized information about configuring your system to be Web aware.

Configuring Virtual Hosts

How to configure your system to support network virtual domains.

Configuring an HTTP Proxy (Squid)

How to configure your the Squid caching HTTP proxy and accelerator.

Configuring an NIS client (YP)

How to configure your BSD/OS system to act as an NIS client.

Configuring Sendmail

The automatic BSD/OS configuration sets up `sendmail`(8) to handle standalone systems or systems on a local Ethernet or the Internet. In addition, BSD/OS is pre-configured to start a POP3 server via the inetd facility. This server supports both the normal "PASS" command and RPOP (via `.rhosts`) authentication. This chapter covers more specialized requirements.

Configuring Netnews (USENET)

How to configure and read netnews. Netnews provides access to hundreds of thousands of articles of information every day.

Configuring Anonymous FTP

The automatic BSD/OS configuration sets up FTP for incoming non-anonymous FTP and outgoing FTP. This chapter covers more specialized requirements, e.g., anonymous uploads.

Configuring UUCP

How to configure UUCP for file and mail network transfer.

Configuring gopher

How to obtain the gopher server and text-based client.

Configuring TeX

How to install and configure the TeX text processing system.

Configuring SCO/iBCS2 Emulation

How to use SCO/iBCS emulation. The ability to execute third party software (e.g., commercial databases) dramatically increases BSD/OS's usefulness.

Configuring NFS

How to configure your BSD/OS system to act as either a NFS client or server. The Network File System (NFS) allows the sharing of filesystems with other systems.

Configuring Filesystem Quotas

How to configure your filesystems to restrict users to some specified amount of disk space.

Configuring RADIUS

How to configure the Livingston RADIUS server.

About Contributed Software

How to use and exploit the contributed software in BSD/OS.

Configuring an Additional Disk

How to install and configure additional disk drives on your system.

Configuring PC Card (PCMCIA) Devices (Wildboar)

How to install and configure PCMCIA Devices for your system. (The BSDI PCMCIA software is also known as the Wildboar software.)

Configuring Power Management

How to configure Advanced Power Management on your system.

Adding and Removing Users

User accounts can be added, removed, and modified using `maxim`(8), which provides a web-based interface to account management.

MaxIM has two modes for creating accounts (also called Views in MaxIM, you can alternate between Views on the page using the links at the bottom). The first, simpler mode assumes many defaults, making account creation quick and easy. The second enables you to individually customize the account in great detail.

The default configuration files installed for a new user are found in the `/usr/share/skel` directory. The installation of these skeleton files is controlled by the `Makefile` in that directory.

The historic interfaces for adding users and groups (`chpass`(1), `addgroup`(8), `adduser`(8), `rmgroup`(8), `rmuser`(8) and `vipw`(8)) remain available in BSD/OS. Generally, these programs are useful when programmatically adding, deleting, or modifying accounts (especially in large numbers), as they support command-line arguments for the user's configuration information. See the specific manual pages for more information.

Configuring Login Classes

Sometimes, simple password-based authentication is inadequate. Packet sniffers make logging in remotely over the Internet a less secure operation if you are sending your password in clear text back to your home site. Other authentication mechanisms like 'Secure Cards' mitigate this problem.

BSD/OS 4.0 supports BSD Authentication, which provides several authentication mechanisms in addition to traditional passwords. Security considerations – particularly across the network where packet sniffers might be operating – make these mechanisms attractive to thwart hackers. Among them are authentication calculators that support challenge-response authentication (so that the "bad guys" cannot break into a system because they have obtained a password by watching the network). Many sites also wish to set resource limits and other parameters differently for different groups of users.

BSD/OS 4.0 uses the concept of "login classes" as well as the ability to provide multiple methods of authentication (potentially restricted to certain classes of users). The class name for each user is set in the field after the group ID in the file `/etc/master.passwd`. This scheme utilizes the `pw_class` entry in the password file database. The value of the `pw_class` entry is the same as the name of an entry in the `/etc/login.conf` file, which specifies the authentication method to use for each user (among a large number of other parameters). The `default` class is used for users with no class name in `/etc/master.passwd`.

The `/etc/login.conf` configuration file is organized as a standard capability database (i.e., in the traditional `termcap` file format; see `getcap`(3) for more information).

Authentication is no longer performed directly by commands such as `login`(8), `su`(1), and `ftpd`(8). A set of programs residing in the `/usr/libexec` directory provide different methods of authenticating users. For example, the traditional login authentication method is implemented by the `/usr/libexec/login_passwd` program.

The various authentication programs may be called without having any special (superuser) privileges. Of course, the authentication programs themselves might require certain special privileges in order to access secure databases. Programs wishing to use the BSD Authentication mechanism need not be setuid root simply to determine if a user knows a password.

Logging In With an Alternate Authentication Method

Users logging in to the system using an alternate authentication method add the name of the method and a colon after their login name as follows: `spike:activ`. This means that the user is logging into the account named `spike`, and wishes to use the the authentication method `activ` (which is used with ActivCard brand cryptographic calculators).

This same mechanism can also be used when FTP'ing to a BSD/OS 4.0 machine.

When using a special authentication protocol with the `su` command, the `-a` option is used: `su -a activ spike`. The first authorization type listed in the `auth=` entry for a class is used by default (when no `:type` is specified).

Controlling Authentication Via User Classes

The default `/etc/login.conf` file contains at least two entries similar to the following:

```
auth-bsdi-defaults:auth=passwd,activ,crypto,snk:
default:\
    :tc=auth-bsdi-defaults:
```

The first entry should **never** be altered, as it contains the list of the standard authentication types. (See the `login.conf`(5) manual page for additional information on the available authentication types.)

The second entry is the default entry used if the user has not been assigned a class in the password database or if their class is specified as "default". In this case, the standard authentication types listed for the `auth-bsdi-defaults` entry in the `/etc/login.conf` file are used.

Authentication methods for `ftp`, `su` and other services may be specified separately by including entries for

```
:auth-ftp=...:
```

and

```
:auth-su=...:
```

in the `/etc/login.conf` file. Furthermore, the `/etc/ttys.conf` file may contain entries of the form `:auth=`*type*`:` to specify the authentication types that may be used, on a per-tty basis. The pty entries in `/etc/ttys`, which are marked as "network", will use the `:auth-network=...:`, type from the user's class entry, if it exists. This is due to the fact that the standard entry for "network" is `:auth=network:`. When a user logs in on a terminal for which an authentication type is specified, the `login` program will first search in `/etc/login.conf` for a corresponding `:auth-`*type*`=...:` list of authentication types. If no such list is found, `login` will use the entries in the default `:auth=...:` list. This can be useful for network users. For example, by adding an entry like:

```
:auth-network=activ,crypto,snk,token:
```

to the `default` entry in `/etc/login.conf`, only non-password methods be used to login when accessing the system via a pty (presumably from the network).

See the `tokeninit`(8) manual page for more information about these authentication types.

Classify Scripts

The entry `:classify=`*path*`:` defines a classify script to be used by `login`(8). This is only useful if specified for the default class, since it is used before the system knows who is attempting to log in.

The purpose of a classify script is to evaluate the session and determine what type of authentication to use, presumably based on the remote IP address. This is useful to allow normal password authentication for sessions originating within the local network, while requiring some sort of challenge-response for all sessions originating from outside the local network. Of course, this would not detect someone logging into another local system and then, from that system, logging into another system on the local network, as the second system will see the request as originating from the local network. For this reason, you should not depend on a classify program for security. It is largely a helpful reminder to use some secure method of authentication when entering the local network from outside.

See `login.conf`(5) for additional information regarding classify programs.

Approval Scripts

The entry `:approve=`*path*`:` in the `/etc/login.conf` defines an approval script for users of a specific class. The `path` references a program that requires two arguments: the user's login name and class name. If the account `jones` has the class of `musician`, and `jones` has been previously authenticated, the program listed as `path` would be invoked with `jones` and `musician` as its arguments. The approval program can then examine system load, time of day, or anything else it chooses to determine if the user may log into the system. If the program exits with a zero status, the user is allowed to log in. If the user is not approved, the approval program might write a message to the standard output file stream to explain why access was denied.

Long Passwords

The maximum and minimum length of passwords can be set in the `/etc/login.conf` file. Historically, only the first eight characters of the password were used to authenticate the user – additional characters were ignored. If the `widepasswords` entry is listed in a user's class entry, the first 128 characters (or all characters if shorter than 128 characters) of the password are used to authenticate users. For obvious reasons, these longer passwords (often referred to as "pass phrases") are far more difficult to crack by mechanical means than the historic passwords. In addition, pass phrases are usually more easily remembered than complex passwords: the pass phrase "Let me log in!" is easily remembered, and practically impossible to crack by mechanical means. Of course, it's now a demonstration password and would be a very poor choice for any of your users.

The minimum length of a password has traditionally been six characters. This can be changed by the `minpasswordlen` entry in the `/etc/login.conf` file. Commonly, password cracking programs assume a password is no longer than eight characters. Adding the entries;

```
  :widepasswords:minpasswordlen=9:
```

to the `/etc/login.conf` file will guarantee that users have passwords that could not have been used on a historical system.

Note that when either `widepasswords` is set or `minpasswordlen` is larger than 8, the encrypted form of the password stored in the `/etc/master.passwd` file will not be portable to any system that does not support the 4.4BSD extended password format.

Resource Limits

Resource limits, such as CPU time, memory use, etc., may also be altered by adding entries to the `/etc/login.conf` file. The values need not be limited to the system default. You might want to have a class of users who have no datasize limit maximum but start with a soft limit of 64 MB. Adding the entry:

```
  :datasize-cur=64M:datasize-max=infinity:
```

to their class entry accomplishes this. You can also restrict both the current and maximum values to 24 MB, for example, by adding:

```
  :datasize=24M:
```

to the class entry.

Further Information

There are several other resources and behaviors that can be modified using the `/etc/login.conf` file. See the `login.conf`(5) manual page for additional information.

Configuring the X Window System

The X window system provides a graphical user interface on your computer's display. It consists of a server program, which displays graphics on your screen, and various client programs, which send graphical data to the server. Client programs may be local to your system, or may run remotely on another system that supports X. Client programs provide features like virtual terminal services, document editing and previewing, image manipulation, and so on. See the `X(1)` manual page for more details.

BSD/OS 4.0 includes two distinct X servers, one from Metro Link, Inc., and one from The XFree86 Project, Inc. You may configure your system to use either server; different servers support different sets of video cards and may have different performance.

Normally, the X window system is already installed (express installations install it automatically). Most sites configure the X window system during installation, before starting `maxim`(8) configuration.

The X window system packages must be installed in order to use the X window system. If you have not already installed the X packages, insert the BSD/OS BINARY CD-ROM, mount the CD-ROM, rerun `installsw`(8), and install the packages.

You need not extract X source or XFree86 packages unless you require them or you will be doing active development or complex debugging of the X Window System itself; an X tree with compiled objects and binaries can require 200 MB of disk space or more.

Motif Availability

BSD/OS does not include the OSF/Motif package. Motif packages are available – contact BSDI for information on purchasing Motif.

Mice

The X window system supports most serial mice, Microsoft- and Logitech-compatible bus-mice, and PS/2-style mice (which use the auxiliary port on the PS/2 style keyboard controller).

There is often confusion over the difference between a "bus-mouse" and a "PS/2" mouse since both use similar (but actually different) connectors. In general, if the socket for the mouse is on the motherboard, it's probably a "PS/2" mouse and it uses the device named `/dev/pcaux0`. If the socket is either on a card by itself or is part of the video card, it's probably a bus-mouse. PS/2 mice are common on newer machines. Bus-mice are quite rare.

When configuring the X server for a PS/2 mouse, "PS/2" is most likely the correct protocol selection, but this is not always the case. If the mouse doesn't behave correctly with the "PS/2" protocol, try the others – many PS/2 mice work with the "Logitech" protocol and others might require different selections.

Note that many bus-mouse controllers must be configured before they can be used. For example, to get the ATI Ultra bus mouse to work, you'll need to boot the ATI configuration utility (included with that card on a DOS diskette) and enable the mouse port at the primary address (0x23c) and IRQ 5.

NOTE: ATI bus mouses can be either of two different device types: Some are type "bms"; some are type "lms".

X Window System Configuration

You will normally configure X during the installation process. (Note: To avoid installing or configuring X, you must use custom installation and explicitly request that X not be loaded or configured.) You may also configure X by running the `config_X` program, even if you have already tried some other X configuration at installation time or at another time. To run `config_X`, you must not currently be running an X server and you must be acting as the super-user.

When you configure X, you will be given the opportunity to select an X server from a list of available servers, and then you will be placed in the graphical configuration utility for that server. You can then tell the system about the video card, monitor and mouse that you have. After the configuration utility finishes, you get the chance to test your new configuration; you may elect to accept it, change it or quit. If you decide to change it, you will get another chance to select a server and run the server-specific configuration utility.

Running X

Once you have configured the X server, make sure `/usr/X11/bin` is in your search path. To check it:

```
  % echo $PATH
```

If `/usr/X11/bin` is not present, be sure to fix your `.profile` (for `bash` and `sh` shells) or `.cshrc` (for `csh` and `tcsh` shells) file so that you can access X Window programs directly.

Start the X window system like this (example for csh):

```
  % set path=($path /usr/X11/bin)      if necessary
  % startx                             starts an xterm window
```

Handy hint: Use <Control-alt-backspace> to cause the X server to exit immediately (especially if things are not going well).

If the server fails to reset the screen correctly on exit, you can often get the hardware to cooperate by running the `vreset(1)` program. Find out what errors the X server generated (even if you were forced to reboot) by looking in the `.xinit.log` file in the home directory of the user who attempted to start the server.

Once you configure the X server, you can adjust "app-defaults" files and tweak other X options. These enable you to change the "look and feel" of the X applications. Applications should run without any customization but it can be fun, on a rainy afternoon perhaps, to set up xterm with a chartreuse background and lime green letters. See `/usr/share/skel` for some starter `.xinitrc` files and other configuration files.

See the `X`(1), `Xserver`(1), `startx`(1), and `xinit`(1) manual pages for more information.

Starting X Automatically

Many sites like to start the window system automatically at boot time. The `/usr/X11/bin/xdm` program (the X Display Manager) will do this for you. To run `xdm`, uncomment the sample lines in `/etc/rc.local`. When you boot, `xdm` will automatically start the X server and display a login box for X. When a user logs in, `xdm` will automatically run their `~/.xsession` file to start all their windows, windows manager, etc. When they exit the X window system (via the window manager, or via the <ctl><alt><backspace> key sequence, `xdm` will automatically restart the server and display a new login box.

X Security

The default X window system configuration uses `Xauth` authentication, so it is relatively secure. You should use the `xauth` command to add 'cookies' (see MIT-MAGIC-COOKIE-1 in the manual page `xauth`(1)) to other hosts to maintain that security rather than using `xhost`. If you do use `xhost`, note that anyone can connect from the hosts you allow (potentially any host) and therefore anyone on those hosts might be able to capture your keystrokes!

The other security concern with X Windows involves the `xlock` program. By default, the X server allows you to force the server to exit with the <control><alt><backspace> key sequence. If you use `xdm` to start the server automatically, executing this sequence will simply cause the server to exit (and the person logged in to be logged out) so there is no security implication. If you use `startx` or `xinit` from a regular user shell, someone using this sequence to cause the X server exit (even from a locked screen) will get back to the regular shell X was started from. To avoid this problem, you should exec the X server (so that you will be logged out when exiting X) by using a command like: `exec startx`.

Configuring Serial and Parallel Devices

Attaching and configuring a serial device requires several steps:

- Configure kernel (for extra serial ports or certain high speed serial devices)
- Ensure the `/dev` entries for the devices exist
- Configure `/etc/ttys.conf.local` (not always necessary)
- Configure `/etc/ttys`
- Look in the `/etc/rc.hardware` directory for the `README.digi`, `README.equinox`, and `README.iopro` files if you have a serial port card from Digi, Equinox, or IOPRO. You will need to create a file in `/etc/rc.hardware` if you have one of these boards.
- Signal the `init` process to recognize the new devices (or reboot)
- Signal the `gettyd`(8) daemon to recycle the serial ports.

Serial Port Hardware Configuration

The GENERIC kernel shipped with BSD/OS supports the standard PC COM ports (COM1: and COM2:). BSD/OS can support additional COM ports and/or a host of other asynchronous cards if they are first configured into the kernel. The GENERIC kernel configuration file included with the release contains commented out entries for the various types of serial cards BSD/OS supports. See the sections on configuring and rebuilding kernels to configure and build a new kernel that will identify these cards.

The serial port controllers must be identified properly by the kernel at boot time before they can be used. If the kernel doesn't find the card, either the card or the kernel is misconfigured, the configuration conflicts with some other device, or the card is broken or not of a supported type. If the card is not found properly at boot time, you must solve that problem before continuing.

See the manual pages in section 4 of the manual for specific information about the various types of supported cards. The introductory page can be especially helpful (***`man 4 intro`***).

Creating Serial Port Entries in `/dev`

The `MAKEDEV`(8) program is used to create device entries in the `/dev` directory. The device entries in `/dev` provide a name for each device (in this case each serial port). The `MAKEDEV` program shipped with the release includes information on how to make device entries for the many types of serial cards supported by BSD/OS, but the device entries are not pre-made for you since that would cause the `/dev` directory to be unwieldy (due to the large number of possible combinations of serial port cards).

To create the `/dev` entries for a device:

```
# cd /dev
# ./MAKEDEV deviceN
```

Where *`device`* is the name of the device you wish to create (usually the same as the device name in the kernel configuration file), and *N* is the number of the device. For example, to create the COM3: device (called `com2` in the kernel configuration file since BSD/OS starts numbering devices at 0), execute the commands:

```
# cd /dev
# ./MAKEDEV com2
```

This creates the device file `/dev/tty02`, which is the name for the third COM port device.

Most of the serial cards supported by BSD/OS provide more than one port per device (COM ports are the exception to the rule). In the case of these devices, a single `MAKEDEV` command will create all of the devices associated with a card. For example, the commands:

```
# cd /dev
# ./MAKEDEV rc0
```

will create the eight device names associated with the first RISCom/8 card: `/dev/ttya0`, `/dev/ttya1`, `...`, `/dev/ttya7` (where the last digit is the port number on the card, again starting with 0).

See the table below for the devices and their associated device file names. The `digi` devices are special in that they use two numbers after the name (separated by a period), the first of which indicates the board number and the second of which indicates the "module" associated with that board (the modular DigiBoards support up to 64 ports on each card in groups of 16 ports per module). For example,

```
# cd /dev
# ./MAKEDEV digi0.0
```

will create the first 16 ports associated with the first Digi card. The argument `digi0.1` specifies the second 16 ports associated with the first card, while the argument `digi1.0` specifies the first 16 ports associated with the second card.

Device	Description	/dev names
com*	standard PC COM ports	tty[01][0-9]
bheat*	Blue Heat PCI multi-port serial card	tty[01][0-9]
rc*	SDL Communications RISCom/8	tty[a-f][0-7]
si*	Specialix SLXOS serial adapter	tty[ab][0-31]
		tty[ab][0-31]_xp
	Specialix control device	si_control
ms*	Maxpeed SS-4/2, SS-4+, etc.	tty[h-o][0-7]
digi*	various DigiBoard products	tty[A-D][00-3f]
	digi or digi0 or digi0.0	ttyA[00-0f]
	digi0.1 ports 16-31	ttyA[10-1f]
	digi0.3 ports 48-63	ttyA[30-3f]
	digi1.0 second digi board	ttyB[00-0f]
	digi3.0 fourth digi board	ttyD[00-0f]

Device	Description	/dev names
cy*	Cyclades Cyclom-Y serial cards	ttyy[0-3][a-p]
	cy or cy0 first Cyclades board	ttyy0[a-p]
	cy1 second Cyclades board	ttyy1[a-p]
	cy2 third Cyclades board	ttyy2[a-p]
	cy3 fourth Cyclades board	ttyy3[a-p]
rp*	Comtrol RocketPort cards	tty[RSTU][0-31]
	rp0 first RocketPort card	ttyR[0-31]
	rp1 second RocketPort card	ttyS[0-31]
	rp2 third RocketPort card	ttyT[0-31]
	rp3 forth RocketPort card	ttyU[0-31]
	rp0.4 first card with 4 ports	ttyR[0-3]
stl*	Stallion Technologies cards	tty[E-H][00-64]
	stl0 1st Stallion dumb card	ttyE[00-64])
	stl1 2nd Stallion dumb card	ttyF[00-64])
	stl2 3rd Stallion dumb card	ttyG[00-64])
	stl3 4th Stallion dumb card	ttyH[00-64])
	stli0 1st Stallion smart card	ttyI[00-64])
	stli1 2nd Stallion smart card	ttyJ[00-64])
	stli2 3rd Stallion smart card	ttyK[00-64])
	stli3 4th Stallion smart card	ttyL[00-64])
	Stallion control devices	stliomem[0-7])

See `/etc/rc.hardware/README.equinox` for creating equinox devices. See `ioprod`(8) for information on Chase Research devices, which are supported by the vendor, not BSDI. Chase research devices are often referred to by the names ‘iopro’ or ‘ioprod’.

Remember the names of the devices you’re creating – you’ll need to know them later when you go to add them to the `/etc/ttys` file.

Some cards (e.g., Digi, Specialix, and Equinox cards) require you to run an initialization program before they can be used. These initialization commands are typically executed from scripts in the `/etc/rc.hardware` directory. See the `README` file in `/etc/rc.hardware` along with the device specific `README` files in that directory for more information.

RS-232 Connection Hardware

Once the hardware is installed and configured, and the device entries have been created, the next step is to actually connect the devices.

Probably the hardest part of installing new serial devices is correctly wiring the RS-232 connector (although modem configuration runs a close second). You must match male plugs to female sockets and match the DTE (Data Terminal Equipment) signal set to its DCE (Data Communications Equipment) counterpart (i.e., the computer’s transmit line must be attached to the terminal or modem’s receive line and so on for the other signals). The easiest way to debug RS-232 connections is to use a break-out box. A simple break-out box enables you to see which signals are active on the cable; fancy break-out boxes tell you exactly what you need to do to properly connect the devices. You can purchase these at computer electronics stores like Radio Shack.

Standard PC serial ports are wired as 25-pin male DTE; if you have the 9-pin AT-style serial ports you can get cables that either convert them to the standard 25-pin RS-232 connector or that connect directly to certain devices (e.g., 9-pin to 25-pin modem cables are common). Most modem ports are wired as a 25-pin female DCE; use a straight-through cable to connect them to the PC serial port.

On the other hand, terminals are generally wired as 25-pin male DTE (just like the PC serial port) and require a "Null Modem" cable to connect them. Null Modem cables convert the DTE signal set to a DCE signal set by crossing (exchanging) the transmit and receive pairs: TD/RD (pins 2 and 3) and RTS/CTS (pins 4 and 5), DCD/DTR (pins 8 and 20). As a general rule, a DTE to DCE connection requires a straight through cable and a DTE to DTE (or DCE to DCE) connection requires a Null Modem cable. Not all equipment strictly follows the DTE/DCE model so some devices require customized cables. A break-out box is probably required to fathom the correct connections in those cases.

Some of the more expensive cables enable you to rewire them easily and therefore ease the connecting of equipment. Also available are "smart cables" that configure themselves and "gender-benders" for connecting two same sex connectors together.

For more information, see the `rs-232`(7) manual page. Also see Evi Nemeth, et al.'s *UNIX System Administration Handbook* ('the red book') for information on "Yost connectors" and RJ-45 wiring techniques.

Decisions for Serial Port Configuration

Choose whether serial port dialins (with users logging in through the serial port) will be allowed or not. If dialins are allowed, decide if the device is to be used for both dialins and dialouts ("bidirectional" mode).

Bidirectional connections enable you to use a serial port for users dialing in and users dialing out (one at a time, of course). Bidirectional serial ports can be a security problem since a user with access to the dial-out serial ports can use the serial port to spoof users into thinking they are logging into the system when they are not actually doing so. This enables a malicious user to acquire other user's passwords.

The `gettyd` daemon can prevent this type of security problem by properly configuring the port and modem. The modem's `hinit` configuration string (which is sent to the modem just before it places any call) must disable auto-answer (S0=0) and make sure that CD is only high during a connection (&C1). The modem's `hcondition` string must set the modem for auto-answer (S0=1). Both the `hinit` and `hcondition` string must disable the modem's escape (+++) character sequence (S2=255). The AT commands above (that correspond to the modem configuration options) might be different for your modem(s). You should check your modem manual to verify that the correct AT commands are being used for your modem(s). The modem configuration strings in the default `/etc/ttys.conf` have all of these options set. The serial port configuration must not be set to `clocal`, and only the `gettyd` daemon should be allowed to do the dialing. For more information on setting modem configuration strings and configuring serial ports with the `gettyd` daemon see the appropriate sections later in this chapter.

If these conditions can not be met, you should probably not use bidirectional serial port connections (use extra serial ports instead), unless you trust your users.

Serial Port Management with the Gettyd Daemon

Historically, each service that used serial ports had to be independently configured and manage both the serial ports and the modems connected to them. It was the System Administrator's responsibility to ensure that the different services did not conflict with each other when using serial ports and modems. It was a common problem for one service to attempt to use a serial port while in use by another service, e.g., attempting to use `tip` to connect out through a serial port that has a `getty` running on it. It was also a common problem for one service to leave a modem in a state that was incorrect for the next service wanting to use it.

BSDI has created a centralized serial port management service called the `gettyd` daemon to alleviate some of the problems associated with the previous methods. The `gettyd` daemon manages access to serial ports, similar to the service `inetd` provides to IP services. A service requiring access to a serial port requests access to a serial port from the `gettyd` daemon. The `gettyd` daemon then assigns a serial port to the requesting service and configures the serial port and modem for the requesting service.

The advantage of this approach is that all services that require serial port access do not have to be independently configured with a list of available serial ports, nor does a service need specific knowledge of how to configure the serial port and modem. Instead, the requesting service only needs to ask the `gettyd` daemon for access in a well defined manner. The `gettyd` daemon maintains all the information required to select a serial port and configure it and the modem to meet the service's request.

Enabling the gettyd daemon

When the `gettyd` daemon starts, it reads the `/etc/ttys` file. Any entry that does not have a flag of `on`, `off`, or `network` will be controlled by the `gettyd` daemon. The `/etc/ttys` entries that have an `on` flag will be managed by the `init` program, as in previous releases. The `init` program will execute the specified command, usually `/usr/libexec/getty`. Entries with the flag of `network` will be available for remote logins and other purposes, as well.

A serial port can be configured as any one of in, out, or bidir. The gettyd daemon will control these ports. The gettyd daemon will monitor serial ports identified as type in and type bidir, and when an incoming connection is made, the gettyd daemon will execute a command, typically the login command, and pass control of the serial port to that command. Serial ports identified as out or bidir will be placed on a list of available out devices. Services that require the use of a serial port can make a request for access to a serial port with specific characteristics.

The gettyd daemon will assign a serial port from the out device list that matches the characteristics of the requesting service. The gettyd daemon will make the necessary configurations of both the serial ports and the attached modems, and then pass control of the serial port over to the requesting service. Serial ports identified as bidir will also be placed on the out list just as ports marked as out only. The gettyd daemon will also monitor any bidir serial port and wait for a connection, identical to ports identified as in. If a connection comes into a bidir serial port before the port is assigned to a requesting service, the indicated command will be executed. If a service requests a bidir port before a connection is established, the bidir serial port will be assigned to the requesting service. Here is a sample /etc/ttys file:

```
# This entry is controlled by init as usual

console  "/usr/libexec/getty std.9600" ibmpc3 on secure

# These ports are not controlled by the gettyd daemon or init
# (This port happens to have a mouse plugged in to it)

tty00  off

# These ports are controlled by the getty daemon
# (Refer to the /etc/ttys.conf file for configuration
#   parameters of these devices)

#DEVICE BAUD     TYPE TERM        MODEM      FUNCTION
#------ -------- ---- ----------- ---------- --------
tty01            com  term=dialup Generic288 bidir
tty02   std.9600 com  term=vt100             in
tty03            com              Generic288 out
tty04            com              Generic288 ppp-only in

# These pseudo-devices are available for network logins

ttyp0  network
```

The console port uses the historic method of login, having the init program execute a getty program on the port. The tty01, tty02, tty03, and tty04 entries are controlled by gettyd. The tty01 port has a 28.8K baud modem attached to a standard com port and is bidirectional. The TERM environment variable will be set to dialup. The tty02 serial port is connected directly to a vt100 terminal and a fixed baud rate of 9600 will be used for that connection. The tty03 serial port has a 28.8K baud modem attached for use for outgoing connections only. The tty04 serial port has a 28.8K baud modem attached and is dedicated to PPP connections only. This port will execute the ppp program when a connection is made, rather than the more standard login command. As a result, users with interactive accounts would not be able to log into the machine through tty04.

In general, an /etc/ttys entry that is to be controlled by the gettyd daemon will have the following format:

```
line [baud rate] [serial-type] [terminal-type] [modem-type] [flags]
```

The fixed baud rate entry overrides gettyd's default action of determining the baud rate by selecting the fastest baud rate that both the serial port and the modem have in common. Typically, you will want gettyd to automatically determine the fastest baud rate common to the serial port and the modem, but if you want to set the baud rate to a fixed speed, precede the serial-type with the appropriate std.<*speed*> setting.

When adding a new port to be controlled by the gettyd daemon, it is important that the correct properties for any given serial port device be selected. If no properties are defined, the baud rate of the device will be undefined. Further, the serial-type entry is used to limit the choices for the baud rate to those values that are available for the device. An outgoing line, that will only be used by tip(1) and for which /etc/remote defines the baud rate, does not require a serial-type entry (though it can still be provided). Below is a list of the supported serial port devices and the corresponding serial-type selection:

Serial-type	Device	Serial Port Description
pccons	pccons	(special case, it is both a modem and a port entry)
com	com	the "standard" com port (16550 UART, buffered)
ast4	com	AST-4 4-port async card
bheat	com	Blue Heat PCI multi-port Serial Card
boca	com	Boca 4 port async card
cyclom	cy	Cyclades Cyclom-Y Multiport Serial Cards
digiboard	digi	DigiBoard intelligent serial controllers
equinox	eqnx	Equinox SuperSerial Technology host controllers
iopro	aim	Chase Research IOPRO
ms_ss4plus	ms	Maxpeed serial controllers (ss4 plus)
ms_ss8	ms	Maxpeed serial controllers (ss-8/2)
mu440	com	MU-440 4-port async card
rocketport	rp	Comtrol Rocketport Intelligent Serial Port Cards
siplus	si	Specialix Intl SLXOS 8-32 (XIO/MTA terminal adapters)
specialix	si	Specialix Intl SLXOS 8-32 (SI/TA terminal adapters)
stallion	stl,stli	Stallion Technologies multiport serial controllers
u16450	com	unbuffered com port
usenetII	com	USENET Serial Board II

The `term=type` field allows you to set a default terminal type for a dialin port. If the port is connected to a modem you will probably want to use dialup as a terminal-type, i.e., `term=dialup`. When adding a new entry in the `/etc/ttys` file, the correct modem type should be selected to indicate what type of modem is connected to the serial port. Lines that are directly connected to a terminal or some device other than a modem, should not define a modem-type entry. The default `/etc/ttys.conf` file comes with the following modem-type descriptions:

Modem-type	Modem
Generic144	Modems with Hayes AT command set at 14.4 kbps
Generic288	Modems with Hayes AT command set at 28.8 kbps
Generic336	Modems with Hayes AT command set at 33.6 kbps
zoom144	Zoom 14.4 kbps modems
deskporte288	Desk Porte 28.8 kbps modem
sportster336	US Robotics Sportster 33.6 kbps modem

The possible flags that can be used for any given entry are:

Flags	Meaning
in	Dedicated for dialin use
out	Dedicated for dialout use
bidir	Used for both dialin and dialout
network	Used by telnet for logins
secure	Root can directly login
ppp-only	Execute the ppp command rather than the login command
on	Use `init` to control the line as in the past
off	Port not used

To add a digiboard to the system and use its first port as a dialin port with a 28.8 modem, you would only need to add the following entry to the `/etc/ttys` file:

```
ttyA00  term=dialup digiboard Generic288 in
```

To add a Cyclades Cyclom-Y Multiport Serial Card port for dialout only with a 14.4 kbps modem, the following entry can be entered in the /etc/ttys file:

```
ttyy0a  cyclom Generic144 out
```

Once the `/etc/ttys` file has been changed with all the correct port settings, the `init` process needs to be sent a HUP signal so that it will re-read the `/etc/ttys` file. You can then send the `init` process a HUP signal with the following command:

```
# kill -HUP 1
```

You will also want to have the `gettyd` daemon recycle any ports that it currently has control over. This will send any newly defined initialization strings to the modems. The following command will send a restart command to the `gettyd` process, forcing each port to be recycled, which will send any new initialization strings:

```
# gettystat -r
```

If there are users currently connected on ports under the `gettyd` daemon's control, the above command will disconnect them. You might want to specify only the ports that have been changed:

```
# gettystat -r ttyA00 ttyy0a
```

Serial Port Definitions

A serial port definition is quite simple. Typically it is nothing more than the list of possible baud rates at which the serial port can operate. An example of one of the default serial port entries in the `/etc/ttys.conf` file is the `digi` entry:

```
digiboard|digi:\
     :dte-speeds=115200|76800|57600|38400|19200|...:
```

(where the ellipses indicate additional speeds in the field).

The digiboard entry consists of only one field called the `dte-speeds` field. The `dte-speeds` field is a list of the possible speeds to which the digiboard serial port can be set. Normal operation of the `gettyd` daemon is to compare the `dte-speeds` field of the serial port entry and the `dce-speeds` field of the modem entry and select the first speed supported by both the serial port and the modem. The order of the `dte-speeds` and `dce-speeds` fields should start with the most desirable speeds, typically the fastest speeds. If you wanted to pick a fixed baud rate you would then want to set the serial-type to a single fixed speed that the modem can support. To select a serial port speed of 9600 baud for the serial port you can select `std.9600`, one of the default fixed speed types in `/etc/ttys.conf:`

```
std.9600:\
     :dte-speeds=9600:
```

This would of course require you to change the `/etc/ttys` file for the port to use the `std.9600` serial-type rather than the `digi` serial-type.

It might be that a serial-type is required that is not covered by the default `/etc/ttys.conf` file. In this case you need to make your own serial-type definition in the `/etc/ttys.conf.local` file. The file named `/etc/ttys.conf.local` is read before the `/etc/ttys.conf` file. Entries in the `/etc/ttys.conf.local` file take precedence over any entry with the same name in `/etc/ttys.conf`. You should take care not to use a name for an entry you create in the `/etc/ttys.conf.local` file that already exists in the `/etc/ttys.conf` file. This can cause some confusion if other entries reference the entry with the common name. The `/etc/ttys.conf` file should **NOT** be modified. Future updates might change the `/etc/ttys.conf` file and any local modifications to it might be lost. Some possible changes for serial-type entries would be to set specific `stty` settings. You might have a situation where you want to use odd parity at a fixed rate of 9600 baud. You could add the following entry to the `/etc/ttys.conf.local` file:

```
odd.std.9600:\
      :stty-modes=parenb parodd:
      :tc=std.9600:
```

The ports that need to use odd parity at 9600 baud would then need to be updated in the `/etc/ttys` file.

Modem Definitions – Hardware

Modem-type definitions are more complicated than serial-type definitions. There are generally two parts to a modem definition. The first part is the hardware specific part. The second part is primarily the software specific part. Below is the hardware portion of a modem entry for a 28.8 baud modem:

```
 Generic288:\
      :modemtype=Generic Hayes Compatible 28.8K:\
      :compression=v.42bis|mnp5:\
      :dce-speeds=115200|57600|38400|19200|9600|4800|...:\
      :modemcontrol:\
      :flowcontrol:\
      :error-correction=v.42|mnp4|lapm:\
      :flow=hw|xon:\
      :modulation=bell103|bell212a|v.21|v.22|...:\
      :speed=28800|26400|24000|21600|19200|16800|...:\
      :speaker=off|on|carrier:\
      :volume=low|medium|high|off:\
      :tc=hayes:
```

(where the ellipses indicate additional values or speeds in the field).

These fields illustrate the different hardware settings that are possible for the modem. For specific information about each type of field you should refer to the manual page for the `ttys.conf`(5) file. The `dce-speeds` and `speed` fields require some special mention. The `speed` field indicates the possible speeds at which the modem can communicate with the other modem over the telephone line. The `dce-speeds` field is a list of speeds at which the modem can communicate to the computer's serial port. The `gettyd` daemon will select the first speed that is on the serial port's `dte-speeds` list and the modem's `dce-speeds` list. The most desirable speeds should come first in the `dce-speeds` field, usually the fastest speeds.

The selected DCE speed might be different than the `speed` at which the two modems communicate. The modem must compensate for the differences in speed by providing buffering of the data when it is transferred from the fast serial port interface to the slower telephone line. This feature is generally referred to as *Speed Buffering* although the documentation for some modems might refer to it as *Locked DTE* rate. This feature is enabled by a modem software setting that is critical for proper operation (regrettably not standard among modems). It is also important to enable `flowcontrol` to enable hardware flow control (likewise not standard). This feature aids the modem in preventing data loss. The `modemcontrol` option allows the computer to recognize that the modem has hung up. It also should be enabled for any modem connection.

Modem Definitions – Software

Modems also have software settings. These software settings are set by sending commands to the modem from the serial port. The most common commands understood by modems are the 'Hayes AT Command Set'. Any modem that recognizes the Hayes AT command set should use the `tc` field to select the `hayes` command set for the software portion of the modem setting:

```
hayes:\
     :hcondition=ATE0&D2&S1&C1S0=1S2=255:\
     :hdial=ATDT$1:\
     :hhangup=ATH:\
     :hinit=AT&D2&C1S2=255:\
     :hquiet=ATQ1:\
     :hreset=AT&F:\
     :condition=/etc/getty/hayes-condition:\
     :dialer=/etc/getty/hayes-call:\
     :hangup=/etc/getty/hayes-hangup:\
     :map=<FLASH>!|<TONE>T|<PULSE>P|<DIALTONE>W|...:\
     :number:\
     :term=unknown:\
     :verbose:
```

(where the ellipses indicate additional actions in the map field).

The important fields of the modem software configuration entry above are the AT command string definitions and the scripts. The scripts are called at various phases by the `gettyd` daemon. Typically, the scripts will do little more than send the appropriate AT command string indicated in the `hayes` entry (or whichever entry is correct).

The `condition` script is called by the `gettyd` daemon to prepare a modem to receive an incoming call. It will typically send the `hreset` and then the `hcondition` AT command strings. These strings should have the correct modem commands to prepare the modem to receive a phone call. The `hangup` script is called by `gettyd` after a session is finished and sends the `hhangup` AT command string to ensure the modem has hung up the phone. This string should have the correct command for the modem to hang up the phone. The `dialer` script is called by `gettyd` when an application has requested a dialout port and provided a phone number for `gettyd` to dial. It will typically send the `hdial` AT string. This string should have the proper AT command to instruct the modem to dial a phone number. A $1 should be inserted in the AT command string where the phone number would go. The `gettyd` daemon will replace the $1 with the specified phone number. See `/etc/ttys.conf`, `ttys.conf`(5), and `hayes`(8) for more information.

Modem Configuration

Before beginning configuration, a discussion of "speed matching" is in order. Years ago, many modems were easy to configure since many 1200 bps modems would communicate with their host computer at exactly 1200 bps.

Modern modems often communicate at some speed with the remote modem and a potentially (and probably!) different speed with the computer. Of course, it is best when the computer communication speed is higher than the highest modem-to-modem speed (to take advantage of compression by modems).

The best scheme for communicating with modern modems will lock the computer↔modem interface speed (the "DTE speed") at a single rate; 57,600 bits/second is a good speed since it probably exceeds the maximum rate of modems running at 28,800 bits/second plus their compression ability.

One advantage of the `gettyd` daemon is that it is no longer necessary to pre-configure a modem. Each time the `gettyd` daemon starts a new cycle on a serial port, a conditioning string is sent to the modem. All that is required is to send the appropriate conditioning string to your specific modem. The list below shows the set of properties that should be configured on your modem (in any mode: dialin, dialout, bidirectional). You will need to double check that your modem uses the same cryptic command set as these commands. The comments are generic and should be documented in every modem's manual.

Below are the modem setting commands used for a Zoom modem and other Rockwell chipset-based modems (e.g., Twincom, Megahertz). You should set up your modem similarly, though the specific commands will be different for some options. Deduce the correct commands for your system:

The first two command codes probably need customizing

AT&Q6	Speed Buffering (locked DTE rate)
AT&K3	Hardware (RTS/CTS) Flow Control (\Q3 on Intel 14.4)
AT&D3	Data Terminal Ready (DTR) drop → modem reset (ATZ)
AT&C1	Data Carrier Detect (DCD) follows modem Carrier

Choose one:

ATS0=1	Enable auto-answer on first ring (for dialin/bidirectional)
ATS0=0	Disable auto-answer (for dialout only)

To configure the `gettyd` daemon to send the appropriate conditioning string for your modem, you will need to define a `modem-type` entry in the `/etc/ttys.conf.local` file that uses the appropriate conditioning string. Here is a sample modem-type entry that can be used in the `/etc/ttys.conf.local` file.

```
# Add new modem definitions here, such as:

example:\
     :modemtype=ACME W. Coyote 28.8 Modem:\
     :hcondition=ATE0&Q6&K3&D3&C1S0=1S2=255:\
     :hdial=ATDT$1:\
     :hhangup=ATH0:\
     :hinit=AT&Q6S0=0S2=255:\
     :hquiet=ATQ1:\
     :hreset=AT&F:\
     :tc=Generic288:
```

You will want to change the `hconditioning` string appropriately for dialin ports and the `hinit` string appropriately for dialout ports. Below is a list of the order that various AT command strings are applied.

Upon `gettyd` taking control of the port:

```
 Dialin and bidir ports:
      hreset
      hcondition
      hquiet
 Dialout ports
      hreset
      hhangup
      hinit
```

After a port has been requested by an application:

```
 Dialout and bidir ports:
      hreset
      hinit
      hdial
```

When selecting a name for an entry in `/etc/ttys.conf.local` be sure the name is not already used by an entry in the `/etc/ttys.conf` file. This can confuse the `gettyd` daemon. So, changing the entry name (from the previous, similar example) from `example` to `default` is probably a bad idea, since `default` is already defined in the `/etc/ttys.conf` file.

Pre-configuring a Modem

It might be necessary for you to pre-configure the modem. The `tip`(1) command can be used to make a terminal connection to the modem so that you can manually send AT commands to the modem. In order for the `tip` command to access the modem, the port to which the modem is connected must be configured by the `gettyd` daemon as dialout. You can use the `gettystat`(1) command to identify if the port is configured for dialout:

```
 # gettystat
 console:status=UNINITIALIZED:secure:manager=init:dialin:speed=9600:
 ttyc2:status=UNINITIALIZED:secure:manager=init:dialin:speed=9600:
 tty01:status=HANGUP:manager=gettyd:dialout:pid=8444:mdmctl=DTR,DSR,RTS:
         speed=9600:
```

The last line has been folded for display.

In the sample above the `tty01` port is available for dialout. If dialout did not appear on the gettystat output line for `tty01`, the entry for `tty01` in the `/etc/ttys` file should be changed to select either `out` or `bidir` as below:

```
 tty01   com Generic288 out
```

When the file is changed, the `gettyd` daemon needs to be made aware of the change and recycled with the following two commands:

```
# kill -HUP 1
# gettystat -r tty01
```

The output of the `gettystat` command should now show the `tty01` port as available for dialout.

The configuration file for the `tip` command, `/etc/remote`, will need to be checked to make sure there is an entry for the `tty01` serial port. The default `/etc/remote` does have an entry for `tty01`:

```
tty01:dv=/dev/tty01:br#9600:
```

or, when using `gettyd()`, add the `du` specifier to tell `tip` to use `gettyd` to control access to the line:

```
tty01:dv=/dev/tty01:br#9600:du:
```

You might want to change the baud rate to 57600:

```
tty01:dv=/dev/tty01:br#57600:du:
```

If the modem you want to configure is connected to a serial port which does not have an entry in the `/etc/remote` file, you will need to add an entry for that serial port. For example to add an entry in `/etc/remote` for the first digiboard serial port, the entry would look like this:

```
ttyA00:dv=/dev/ttyA00:br#57600:
```

Be sure to set the `dv` field to `/dev/ttyA00` as well as setting the entry name to `ttyA00`.

Once an entry for the serial port exists in `/etc/remote`, the `tip` command can be executed to make a connection to the modem:

```
# tip tty01
CONNECTED
```

The `CONNECTED` message might appear in lower case.

The `tip` program connects your keyboard to the modem. Characters that you type on the keyboard will be sent to the modem. Characters sent from the modem will appear on your screen. You might wish to be sure the modem is in echo mode by entering the `ATE1` command. This instructs the modem to echo back the characters that you type. You can now send any AT commands that are necessary to properly pre-configure your modem.

Discussion: Carrier Detect

Under BSD/OS, serial interfaces normally require the carrier detect signal (CD) to be asserted on the serial port before the serial line can be used (i.e., before `open(2)` will return successfully). Setting the `clocal` flag using `stty(1)` will enable you to open a serial device in the absence of the carrier detect signal. Note that system software (e.g., `tip` and `uucico()`) will automatically set the `clocal` flag on outgoing connections so you rarely need to do this manually.

You do need to set the `clocal` flag whenever you are connecting to a terminal or dialout device that does not properly assert carrier detect, or if you are attempting to connect to a dialin device (perhaps to change its configuration). For example, to ignore the absence of carrier detect on `tty00`, type:

```
# stty -f /dev/tty00 clocal
```

When you are finished with the device, you can restore the default behavior by clearing the `clocal` flag, e.g.:

```
# stty -f /dev/tty00 -clocal
```

Note that you can not set up a dialin connection on a device that does not properly assert carrier detect when a connection is made. With hardwired terminals this is not a big problem, just use a serial-type entry that contains `-modemcontrol` in `/etc/ttys.conf` (e.g., see the `hw.9600` entry). If you do this with a dialin modem, then the system will not be able to hang up properly when the caller disconnects or should be disconnected.

Using the Gettyd Daemon with Non-Gettyd-Daemon-Aware Programs

Note: Do not use the `gettyd` daemon with Hylafax.

Not all programs that require serial ports know how to communicate with the `gettyd` daemon. There are three ways of dealing with such programs.

The first is to simply not have the `gettyd` daemon control the ports these programs use. Of course, this also means these ports will not be available to any program that does use the `gettyd` daemon. This might be a good choice for a serial port that is dedicated to a particular program. In the `/etc/ttys` file, set the serial ports flag to `off` so the `gettyd` daemon will not attempt to take control of the serial port.

The second is to assure that daemons not aware of the `gettyd` daemon are using *uucp locking*. By default, `gettyd` also honors uucp locking, though it does not require uucp locking in order to keep track of which serial ports under its control are currently in use. Uucp locking is only provided for those programs that support it but do not know how to use the `gettyd` daemon.

The third is to use the program `gettydproxy(1)`. This program will contact the `gettyd` daemon and then pass the name of the serial port to the program not aware of the `gettyd` daemon. Typically, `gettydproxy` will be called by a shell script that masquerades as the program and that will call the real program via `gettydproxy()`. If the program closes open file descriptors when it starts up, as some programs do, `gettydproxy` will not work and one of the above two methods should be used. The advantage of `gettydproxy` is that the program can take advantage of the `gettyd` daemon's modem configuration and dialing abilities.

Running Hylafax

To use `hylafax`, edit the `/etc/ttys` file to include lines like:

```
ttyA00 "/usr/contrib/lib/hylafax/faxgetty" dialup on # fax modem1
ttyA01 "/usr/contrib/lib/hylafax/faxgetty" dialup on # fax modem2
```

Be sure to see the Hylafax manual page and the `faxaddmodem`(8c) and `faxconfig`(8c) pages for lots of information on configuring Hylafax.

Dial-out Security & Configuration

Dial-out access via `tip` is granted only to those users in the group `dialer` (as defined by the `/etc/group` file). Use your favorite text editor to add user's names to the `dialer` line in the `/etc/group` file:

```
dialer:*:117:smith,jones,brown
```

The group number 117 is arbitrary, but as many existing programs use it, you should not change it.

Note that the `ppp`(8) program uses the `netdial` group.

Attaching Printers

The print queue utilities (`lpr`, `lpq`, etc.) get print queue and configuration information from the `/etc/printcap` file. The default `/etc/printcap` file contains sample entries for dumb printers (lp), PostScript (ps), and HP LaserJet II and compatible printers (lj). Many DeskJets work with the LaserJet (lj) configuration.

You'll need to edit `/etc/printcap` to set the devices to match your system. Some things that commonly need adjusting are:

- Define the default printer by setting the printer name to `lp`. The default `/etc/printcap` file sets a dumb printer up as the default printer for the system, by virtue of the leading `lp`. To choose one of the other `/etc/printcap` entries to be the default printer, change the leading `lp` entry for the dumb printer to something else, e.g. `pl`. Then add a leading `lp` to the entry you would like to be the default printer. For example, if you wanted the LaserJet entry to be the default printer, change the line:

  ```
  lj|HP LaserJet II:\
  ```

  to

  ```
  lp|lj|HP LaserJet II:\
  ```

  Make sure when you are done there is only one printer entry with a leading `lp`.
- `lp`: the printer device name (the name in `/dev`, e.g., `/dev/lp0` for the first parallel port)
- `br`: the baud rate for serial printers
- `ms`: the stty-style mode string, for serial printers

See the `printcap`(5) manual page for specific details about configuration.

When attaching a printer to a serial port, you should make sure the port is set to `off` in the `/etc/ttys` file. Otherwise the `gettyd` daemon will send login messages to the printer and wait for it to login. See the section *Serial Port Management with the Gettyd Daemon* for examples of configuring the `/etc/ttys` file.

Parallel Ports

BSD/OS should automatically detect the standard `LPT1:`, the parallel port configured at I/O Port 0x378 or 0x3BC. Either way, it will configure as device `lp0`. If you have a second parallel port (i.e., `LPT2:`), then you will need to enable the `lp1` device in your kernel configuration file and rebuild the kernel. Here is a segment of a kernel configuration file with both parallel devices enabled:

```
# parallel printer port
# set flags to 0 for CRLF expansion, 1 for raw
lp0     at isa? port 0x378 flags 1
lp1     at isa? port 0x278 flags 1
```

You should always use `flags 1` or else you will only be able to print plain text documents.

After configuration, follow the directions for configuring, recompiling, and installing your new kernel in *Rebuilding the Kernel* in the administrative notes.

BSD/OS allows the parallel port to be used either as a printer or for a Xircom Ethernet adapter. To enable sharing, the kernel configuration line should list the port address on the `device lp0` line, plus it should have one additional line like:

```
pe0   at isa? port 0      # on lp0
or
xir0  at isa? port 0      # on lp0
```

When the system boots, the port will be configured for the printer. To start using the Xircom, issue an appropriate command:

```
# ifconfig pe0 ... up
```

or

```
# ifconfig xir0 ... up
```

This will fail if the printer is currently in use or if no adapter is plugged in. If it succeeds, attempts to open the `/dev/lp0` device will fail. To return to using the printer, run `ifconfig pe0 down` (or `ifconfig xir0 down`).

See the `lp`(4) manual page for additional information.

Configuring Dialup IP Services

Dialup IP connections enable a client computer to connect to a host computer and establish a SLIP or PPP connection for Internet-style communication. The BSDI system can act as both a client (establishing calls to remote systems) and/or as a server (answering calls from other systems and supplying IP services to them).

In the BSD/OS system, dialin PPP and SLIP are implemented on a server through the paradigm of creating a fictitious login-user (a user that does not have an interactive shell like `bash` or `csh`) for each client. Instead, this user has the `/usr/bin/ppp` program as its shell. It is best to consider this user account to be an account for your customer's machine, not an account for the customer themselves (who might or might not have an additional account).

In the BSD/OS paradigm, fictitious users have usernames that begin with capital letters and will thus not be able to receive email directly (in fact no user on the system with any capital letters in their login name can receive email). If you want the customer themselves to have an account that can collect email, you will want to give them a separate username with all lower case letters.

If the user does not login to the server and you wish them to receive email to download with a POP mail reader, assign to them a non-interactive shell (e.g. `/sbin/nologin` or `/dev/null`), so there is no question. If you want them to have a standard interactive shell, then assign one (e.g., `/bin/csh` or `/bin/sh`). The `maxim` admin tool can do all of this for the interactive account.

Of the two different types of accounts (the PPP or SLIP account for the customer's machines and the one for the customers themselves), one starts a network connection, the other starts an interactive shell. Two separate accounts also enables separate logging for connect times since some sites wish to charge different connect rates for the two kinds of accounts.

Be sure the customer understands that although they might connect their machine to your machine using one username and password combination, they might need to use a different username and password combination to collect mail or `telnet` into your machine.

BSDI systems can also act as a PPP or SLIP client when configured with a dialout connection. A dialout connection can be set up to dial on command, to maintain a continuous connection by redialing whenever a connection fails, or to dial whenever there is (outgoing) network traffic. A dialout configuration requires the phone number(s) for a server system along with authentication information required to login to that server.

Multilink PPP is an extension to PPP that enables multiple serial lines to be used for one logical connection, thus increasing performance by using more hardware resources. Each multilink session requires a PPP interface for each serial line in use and also a 'Parallel Interface' (pif) unit.

The Fast Path to PPP and SLIP Configuration

You can use MaxIM to configure PPP and SLIP both for clients dialing in ('dialin') to your computer and for your computer dialing out ('dialout') to other computers.

Requirements for configuring 'dialin' include:

- Authorizing users to connect to your system.
- Selecting the authentication method (PAP, CHAP, login/password).
- Selecting IP addresses to assign to dialin users.
- Ensuring that your system forwards packets to the Internet correctly.

Several files control these operations. The MaxIM system should be able to configure all these files for the common cases we have anticipated you will require.

Requirements for configuring 'dialout' include:

- Naming the remote system.
- Configuring the remote telephone number, account id, authorization scheme, and password.

MaxIM also configures files that describe SLIP and PPP operations.

To configure your 'dialin' and 'dialout' PPP and/or SLIP connections, follow this simple checklist:

- Invoke the graphical MaxIM administration interface.
- Find the 'DialupIP Networking' section.
- Select 'Base Configuration.
    - ○ Select the number of PPP interfaces (usually the number of modems being used for *dialin*).
    - ○ Select the number of multilink PPP sessions (often 1 – this new standard is not in wide use).
    - ○ Choose PAP and/or CHAP for authentication.
    - ○ Double check that the Local IP Address is correct.
    - ○ Submit the form.
    - ○ Go 'BACK' to the first page.
- If you foresee dialin users that will be automatically assigned IP addresses, select 'Allocate Dynamic IP Pool'.
    - ○ Follow the form's directions to configure the pool's starting IP address (of presumably contiguous addresses), PPP interface number that matches the starting IP address, and the number of interfaces to assign.
    - ○ Submit the form.
    - ○ Go 'BACK' to the first page.
- Select 'Dialin(inbound) IP Networking' for the rest of the dialin configuration.
    - ○ For each dialin user, choose an account name, address assignment scheme ('Dynamic' is easiest!), password, and protocol.
    - ○ Submit the form.
    - ○ After the last account name is entered, go 'BACK' to the first page.
- For dialout configuration, select 'Dialout(outbound) IP Networking'.
    - ○ Fill in the remote system name, account/ID, password, and system phone

  number.
  - Choose the authentication scheme (only PAP and CHAP are currently supported).
  - Submit the form.
  - Repeat for each remote system that will be called.
- Go 'BACK' to the first page or exit.

Incoming users should be able to dial in. Use

```
# ppp machinename
```

to start outgoing SLIP/PPP sessions.

Unless you have problems or are curious about the details, you should be able to skip the rest of this chapter.

Configuring a Kernel to Support PPP and SLIP

Before any PPP or SLIP interfaces can be used, it is necessary to configure support for them in the kernel. By default, the GENERIC kernel is configured with support for two asynchronous PPP interfaces (pseudo-device name appp), one SLIP interface, and one Multilink PPP interface. This will support a maximum of two PPP connections (that might be multilinked together) plus one SLIP connection at a time. This is adequate for initial testing of PPP, SLIP, and multilink interfaces, and for systems that only require one connection (e.g., a dialout connection to an Internet bandwidth provider).

The system now supports two ways to increase the number of simultaneous PPP, SLIP, and multilink PPP connections. One way requires kernel reconfiguration with an appropriate number of PPP, SLIP, and multilink PPP interfaces. Refer to the *Configuring a Kernel* section of this manual for details about how to configure a new kernel. Below is a discussion about kernel configuration specific to PPP, SLIP and multilink PPP.

The easy way is to use MaxIM (see the 'Base Configuration' selection under 'Dialup IP Networking' described on the previous page), which exploits a new `sysctl` option to change the number of interfaces dynamically.

You can test whether your kernel supports PPP and SLIP by examining the output of `netstat`(1):

```
# netstat -in
Name Index   MTU Speed Mtrc Address           Network
 [... some output omitted]
sl0*     3   308          0
ppp0*    5  1500          0
```

If you see 'sl0' and 'ppp0' (and maybe other, similar lines), then you know your kernel is ready to go.

If you need to rebuild your kernel, the following lines in the kernel configuration file determine the number of simultaneous PPP, SLIP, and multilink PPP connections:

```
pseudo-device sl   1
```

```
pseudo-device appp 2 # requires options PPP
pseudo-device pif  1 # required for PPP Multilink Prot.
```

You will want to increase these numbers to reflect the number of PPP, SLIP, and multilink PPP connections you wish to support. For example, if you had 16 serial ports, you would probably want to allow 16 PPP and 16 SLIP connections (if you don't know if a user will dial in to request a PPP or SLIP connection). Since each multilink PPP would probably take at least two PPP connections, a reasonable choice for the number of multilink PPP connections would be between 8 and 16. Modify the above kernel configuration lines to look like the following:

```
pseudo-device sl   16
pseudo-device appp 16 # requires options PPP
pseudo-device pif  8  # req'd for PPP Multilink Protocol
```

It is not necessary that the number of `sl` pseudo-devices match the number of `appp` pseudo-devices. The number of `sl` pseudo-devices controls the number of possible simultaneous SLIP connections, while the number of `appp` pseudo-devices controls the number of possible simultaneous PPP connections. The number of `pif` pseudo-devices determines the maximum number of simultaneous multilink connections. (The parallel-interface `pif` devices can also be used with point-to-point network types other than PPP; see the section *Configuring a Custom Network*.)

Also, notice that for the kernel to support the PPP protocol, the PPP option must be enabled in the kernel. You will find a line in the kernel configuration file that looks like this:

```
options PPP
```

This configures the PPP protocol modules into the kernel. Without it, you will not be able to establish a PPP connection. This option is enabled by default in the GENERIC kernel; it is best not to change it.

After you have rebooted to the new kernel, you can verify that the appropriate number of PPP, SLIP, and multilink PPP interfaces have been configured into the kernel by using the `netstat -in` command:

```
# netstat -in
Name Index   MTU Speed Mtrc Address           Network
ef0      1  1500    10M   0 00:20:af:e9:67:99
ef0      1  1500          0 192.0.0.1         192.0.0
lo0      2  4352          0
lo0      2                0 127.0.0.1         127
sl0*     3  308           0
sl1*     4  308           0
sl2*     5  308           0
sl3*     6  308           0
ppp0*    7 1500           0
ppp1*    8 1500           0
ppp2*    9 1500           0
ppp3*   10 1500           0
```

```
 pif0*   11  576           0
 pif1*   12  576           0
```

You can see that the machine above has been configured to support four SLIP interfaces, `sl0–sl3`, and four PPP interfaces, `ppp0–ppp3`, and two multilink PPP interfaces, `pif0–pif1`.

Manually Configuring PPP and SLIP on BSD/OS 4.0 for Dialin Services

Two steps are required to set up each PPP or SLIP account. The first step is to configure the authentication method and provide an appropriate account for the host that is dialing in. There are three different authentication methods available:

- standard login authentication
- PAP authentication
- CHAP authentication

Once the account for the host has been created, the connection characteristics need to be configured.

Configuring Standard Login Authentication

This method of authentication treats the PPP or SLIP host like a normal user. The dialin host must be configured to negotiate the standard "Login" and "Password" prompts. Once the user has successfully logged in, the `/usr/bin/ppp` program executes to establish the PPP connection. This is the only authentication method that can be used for SLIP hosts.

The `maxim` admin tool can be used to create a login for the PPP, SLIP, or multilink PPP user. These usernames typically begin with a "P" for PPP accounts, or "S" for SLIP accounts, so they are easy for an administrator to identify as PPP or SLIP accounts. Configure each of these users to have a primary group of 118 (the netdial group) and a shell of `/usr/bin/ppp`.

Configuring PAP (Password Authentication Protocol)

The PAP method of authentication is very similar to the standard login method. However, instead of having the dialin computer negotiate the "Login" and "Password" prompts, the username and its password are passed to the server as part of the PPP protocol. This authentication method can not be used by SLIP hosts. To enable PAP capability on the server the `ppp_direct` entry in `/etc/ppp.sys` file should be uncommented:

```
 # Uncomment the following entry to allow PAP
 # and/or CHAP dialin:

 ppp_direct:\
         :dialin:\
         :pap:\
         :chap:\
         :authretries=3:
```

When a host dials in to the server, it can immediately send a PPP Link Control Protocol Packet. When the login program receives a PPP Link Control Protocol Packet instead of an ASCII username in response to its "Login" prompt, the login program will execute the ppp program using the ppp_direct entry defined in /etc/ppp.sys. The dialin computer will send the username and associated password to the ppp program on the server. The username and password are sent out the serial port in ASCII form, just as they are when negotiating the standard login method.

The ppp program will check the username against the /etc/master.passwd file. If the username exists in that file, verification of the username and password will be attempted using the authentication method specified by the auth-pap style of authentication defined in the user's class. If no auth-pap style of authentication is defined then the auth style will be used, refer to the man page for login.conf for more information on authentication styles. If the verification is successful, the ppp connection will continue to be negotiated. If the verification is not successful the ppp connection will be terminated. For more information about the PAP method of authentication, you might want to reference RFC 1334.

If the username does not exist in the /etc/master.passwd file, The ppp program will check for a pap-default class definition in the /etc/login.conf file. If a pap-default class exists in the /etc/login.conf file, an attempt will be made to verify the username and password using the auth-pap authentication style defined for the pap-default class. For example, if a PAP packet is received with a username of *Phomer*, and *Phomer* does not exist in the /etc/master.passwd file, but there is a pap-default class defined in /etc/login.conf that looks like like this:

```
pap-default:\
        :auth-pap=rpasswd:
```

then *Phomer* will try to be authenticated using the rpasswd authentication style. If this is successful then a PPP connection will be established.

It is not necessary for a user to be defined in the /etc/master.passwd file in order to be authenticated via PAP. Of course, if there is no user defined in the /etc/master.passwd file, then that username will not be able to login by negotiating the standard "Login" and "Password" prompts. The username will only be able to login via PAP or, possibly, CHAP. If there is more then one auth-pap authentication style defined in the user's class, it is possible to specify which style of authentication is to be used when connecting via PAP by adding a ':' and the authentication style to the username. If the authentication style is listed in the auth-pap authentication list for the user's class, or the pap-default class of the username does not exist in the /etc/master.passwd file, then that authentication style will be used for verification.

The maxim admin tool can be used to create a login for the PPP, SLIP, or multilink PPP user. The maxim admin tool will create an entry in the /etc/master.passwd file for the username. If you do not want the username to be in the /etc/master.passwd file, you should not use maxim to add the PPP user. These usernames created by maxim typically begin with a "P" so they are easy for an administrator to identify as PPP accounts. It is not required that these usernames be members of the netdial group or that the shell be /usr/bin/ppp as is required for the standard login method. If you do set them to these values, the client can choose to implement the standard login authentication method.

Configuring CHAP (Challenge-Handshake Authentication Protocol)

The CHAP method of authentication is performed as part of the PPP protocol, like the PAP method, but is more sophisticated in its username and password information exchanges. To enable CHAP capability on the server the ppp_direct entry in the /etc/ppp.sys file should be uncommented:

```
# Uncomment the following entry to allow PAP
# and/or CHAP dialin
ppp_direct:\
        :dialin:\
        :pap:\
        :chap:\
        :authretries=3:
```

When a host dials in to the server, it can immediately send a PPP Link Control Protocol Packet. When the login program receives a PPP Link Control Protocol Packet instead of an ASCII username in response to its "Login" prompt, the login program will execute a ppp program using the ppp_direct entry defined in /etc/ppp.sys.

The server will send a CHAP challenge packet to the client. The CHAP challenge packet contains the name of the server and a random string known as the "challenge". The client must respond to the challenge packet with a CHAP response packet. The response packet contains the client's name and the "challenge" encrypted with the client's "secret" key. The server then looks up the "secret" key for the client in the local configuration file (/etc/chap_md5_secrets). If the client's name is not found, the connection is refused. If the client's name is found in the server's configuration file (/etc/chap_md5_secrets), the previously sent "challenge" is encrypted by the server, using the client's "secret" key, retrieved from the server's configuration file (/etc/chap_md5_secrets). If the result of the server's encryption does not produce the same result returned by the client, the connection is refused. If the server's encryption of the challenge does match the client's returned encrypted value, the connection continues. Notice that with CHAP the "secret" is never sent across the serial lines, only an encrypted value derived from the "secret". The home directory for 'Puser' is /etc/netscripts/HOME/.

Both machines need to agree ahead of time on the "secret" (usually by the machines' administrators sharing a telephone call). The "secret" is stored in the `/etc/chap_md5_secrets` file. The format of an entry is:

```
  name:format:secret:
```

The name field is the name used in the CHAP packets and the secret is the secret used to generate the encrypted value in the CHAP packets. The format field can be one of "hex" or "ascii" to identify whether the secret is hexadecimal or ASCII. The entry on the server for a client using a name of "Puser" and a secret of "midaspen2" would look like:

```
  Puser:ascii:midaspen2:
```

If you prefer hexadecimal secrets, the entry for "Puser" with a hex secret of 0x6789abcd would be:

```
  Puser:hex:0x6789abcd:
```

Notice that for CHAP, the user authentication file is **NOT** the `/etc/master.passwd` file. Therefore, the CHAP username can be used to establish a PPP connection, but that username can not be used for email, ftp, or telnet access. You might wish to add the username to the `/etc/master.passwd` file in order to grant to the PPP user the other standard user privileges on the system. This also will allow the PPP client to use either CHAP or PAP methods. If the group for the username is set to the netdial group (118) and the shell is set to `/usr/bin/ppp`, the client may choose to use the standard authentication method as well.

Configuring the /etc/ppp.sys file

Once an appropriate authentication account has been created for a PPP or SLIP user, an entry has to be made in the `/etc/ppp.sys` file for the user. An entry might be:

```
  Puser:dialin:local-addr=192.0.0.1:remote-addr=192.0.0.2:
```

or like this for a SLIP account:

```
  Suser:dialin:slip:local-addr=192.0.0.1:remote-addr=192.0.0.2:
```

The "slip" field above indicates that the SLIP protocol should be used for the connection rather then the PPP protocol. The 192.0.0.1 is the IP address of your BSD/OS machine and 192.0.0.2 is the IP address of the machine dialing into the BSD/OS machine. This is the simplest entry and is recommended for initial PPP or SLIP setup. More complex schemes are available, but it is best to start off with the simplest possible configuration, like the one above.

If you specify `local-addr` and `remote-addr` options for all entries in the `/etc/ppp.sys` file, each possible dialedin host has its own dedicated IP address (which means, of course, that you can have only 254 PPP or SLIP customers on each class C network and each separate PPP or SLIP user can have at most one simultaneous dialin on a given login name). For larger sites, this is undesirable (since you might end up with lots of customers and not enough addresses).

The solution to this situation is a feature called *Dynamic IP Allocation*, which assigns IP addresses to the dialin computers once they dial in. In this case, the total number of addresses used never exceeds the number of connections from calling machines. Dynamic IP Allocation is specified by using a 'link-init' field in the `/etc/ppp.sys` file:

```
  Puser:dialin:link-init=/etc/netscripts/Dialin.login:
```

or for SLIP:

```
  Suser:dialin:slip:link-init=/etc/netscripts/Dialin.login:
```

The "link-init" field specifies a program that assigns an IP address to the machine which has dialed in. The `/etc/netscripts/Dialin.login` script has been provided to assign dynamic IP addresses. This script references the `/etc/netscripts/addr-map` file in order to determine how to assign dynamic IP addresses. The file consists of one line entries with two fields. The first field can be either a ppp or slip interface, or a tty port. The second field is the IP address to be assigned to that interface or tty port. It is strongly recommended to use network interfaces for IP assignment, rather than tty serial ports. You will need to edit the `/etc/netscripts/addr-map` file to make the assignments for your interfaces and IP addresses. Here is a sample file:

```
  ppp0     192.0.0.1
  ppp1     192.0.0.2
  ppp2     192.0.0.3
  ppp3     192.0.0.4
  sl0      192.0.0.5
  sl1      192.0.0.6
  sl2      192.0.0.7
  sl3      192.0.0.8
```

When a user connects he will be assigned a network interface, the `/etc/netscripts/Dialin.login` script will use the IP address associated with the chosen network interface from the table above. If the connection gets assigned ppp0, the `/etc/netscripts.Dialin.login` script will assign an IP address of 192.0.0.1 to the remote client.

It is also possible to dedicate a specific IP address to a PPP or SLIP user. To do this, add a line to `/etc/netscripts/addr-map` that has the PPP or SLIP user's name from the `/etc/ppp.sys` file in the first field and the IP address dedicated to that user in the second field:

```
  Puser      192.0.0.101
```

Each time that Puser dials in, he will be assigned an IP address of 192.0.0.101.

It is also possible to specify the local side IP address via the `/etc/netscripts/addr-map` file. The keyword LOCAL can be used to set the IP address for the local side of the connection:

```
  LOCAL    192.0.0.254
```

The above line in `/etc/netscripts/addr-map` will make the IP address of the local side of the connection be 192.0.0.254 for every `/etc/ppp.sys` entry that uses the `/etc/netscripts/Dialin.login` script to determine IP addresses. It's OK for one local side IP address to be used for all PPP connections. Typically this local side IP address is the same as the local ethernet's IP address.

PPP, IP Forwarding, and Proxy ARP

The next thing you will probably find is that the machine that dialed in can talk to the BSD/OS machine fine, but can not talk to any other host on your network. There are two reasons for this: IP Forwarding and Proxy ARP.

Check that IP Forwarding is enabled with the command:

```
# sysctl net.inet.ip.forwarding
```

If the result is "1", IP Forwarding is enabled. If not, see the *Custom Network Configuration* chapter for more information. Without IP Forwarding enabled, no packets from the ppp connection will ever be forwarded to the Ethernet or token ring interface and vice-versa.

If the machine dialing to the BSD/OS machine is on the same IP network as the local Ethernet (i.e., in the usual case of a class C address, the first three components are the same), then the machines on the local network believe the dialed in machine is also on that network. Therefore, they broadcast an ARP (Address Resolution Protocol) packet on the Ethernet to ask if any machine knows what hardware MAC address (Ethernet, FDDI, or token ring hardware address; not IP address!) should be used to send a packet to the dialed-in machine's IP address. No host responds to this request so the machine cannot send a packet to the dialed-in machine.

It is possible (and desirable!) to get the BSD/OS server to respond to this ARP request. This is referred to as Proxy ARP and is also discussed in the *Custom Network Configuration* chapter. Once the BSD/OS machine is configured to respond to ARP requests from other hosts on the Ethernet or token ring that want to send packets to the dialed-in machine, all machines on the local Ethernet or token ring will be able to communicate with the dialed-in machine.

To automate this Proxy ARP, you can use the "link-up" and "link-down" scripts to manage the ARP tables. The `/etc/netscripts/Dialin.up` and `/etc/netscripts/Dialin.down` scripts have been provided to manage this for you. The fully functional `/etc/ppp.sys` entries look like this:

```
Puser:\
        :dialin:\
        :link-init=/etc/netscripts/Dialin.login:\
        :link-up=/etc/netscripts/Dialin.up:\
        :link-down=/etc/netscripts/Dialin.down:
```

or, for a SLIP user:

```
Suser:\
        :dialin:slip:\
        :link-init=/etc/netscripts/Dialin.login:\
        :link-up=/etc/netscripts/Dialin.up:\
        :link-down=/etc/netscripts/Dialin.down:
```

If you plan on having lots of dialins, you can streamline the `/etc/ppp.sys` file by using the `tc` capability in the entries. Create a standard dialin entry and then use `tc` to refer all user names to that standard entry:

```
Dialin:\
        :dialin:\
        :link-init=/etc/netscripts/Dialin.login:\
        :link-up=/etc/netscripts/Dialin.up:\
        :link-down=/etc/netscripts/Dialin.down:

Puser:tc=Dialin:
Puser2:tc=Dialin:
Puser3:tc=Dialin:
Suser1:slip:tc=Dialin:
Suser2:slip:tc=Dialin:
```

The sample `/etc/ppp.sys` comes with a Dialin entry, as defined above.

Debugging PPP and SLIP Dialin Connections

After the PPP and SLIP users have been configured, you will want to check the connection. A simple first check (for users that are configured to support the standard login authentication method) is to use `telnet` to connect to your own machine (strange but true). This test method will not work for users that are configured to support only PAP and/or CHAP authentication methods and not the standard login method. Identify success for a PPP connection if you get random characters after supplying the PPP user's login and passwd. For example (shown here with some lines folded for presentation purposes):

```
$ telnet localhost
Trying 127.0.0.1...
Connected to localhost.mydomain.com.
Escape character is '^]'.
BSDI BSD/OS 4.0 (myhost.mydomain.com) (ttyp2)

login: Pauto1
Password:
Last login: Tue Oct 2 16:28:05 from myhost6
BSDI BSD/OS 4.0 Kernel #0: Thu Oct  2 13:21:21 MDT 1997

Use of this software is governed by the BSDI End User
Software License.

If you do not accept the terms of this license,
immediately return the distribution to the place
of purchase for a full refund. Further use of the
software will be considered to be acceptance of the
terms of the license.

}#!}!}4} }8}!}$}%}%}&}
```

The strange characters on the last line are the beginning of the protocol's binary data exchange.

For SLIP accounts, you can identify success when the IP address assigned to the remote side of the link is displayed:

```
$ telnet localhost
Trying 127.0.0.1...
Connected to localhost.mydomain.com.
Escape character is '^]'.
BSDI BSD/OS 4.0 (myhost.mydomain.com) (ttyp2)

login: Puser
Password:
Last login: Tue Oct 2 10:05:20 from myhost
Your address is 192.0.0.2
```

The remote side can use the displayed IP address to bring up its network interface. This effectively provides Dynamic IP allocation for SLIP.

Disconnect from `telnet` by entering Control-] (hold the Control key and press the right bracket key). You should receive a `telnet` prompt; enter the `quit` command and press <Enter> to exit `telnet()`.

One common problem that can occur is that the `telnet` session will not connect successfully but rather will disconnect quickly with the following message:

```
Connection closed by foreign host.
```

The most likely problems in this case are either that the user's `/etc/master.passwd` file entry is incorrect or the ownership or permissions of the `/usr/bin/ppp` file is incorrect. For the user, you will want to be sure the primary group is 118 and that the user's shell is `/usr/bin/ppp`. A primary group of 118 is required so that the user can execute the `/usr/bin/ppp` command. The user's passwd entry should look like this (*but on one line*):

```
Puser:lJFwI1G4TyISI:102:118::0:0:PPP for Pauto1:
                    /etc/netscripts/HOME:/usr/bin/ppp
```

The key things to check are that the primary group (the fourth colon separated field) is indeed 118, and that the shell (the last field) is `/usr/bin/ppp`. To verify the permissions on the `/usr/bin/ppp` file itself you can use the following command:

```
# ls -lg /usr/bin/ppp
-r-sr-x---  1 root netdial 24576 Dec 11 16:01 /usr/bin/ppp
```

The output should resemble the output above. If not, you should change the protections, owner, and group to look like the above. To set the `ppp` program to the correct ownership and permissions, type the following two commands:

```
# chown root.netdial /usr/bin/ppp
# chmod 4550 /usr/bin/ppp
```

The other likely reason that the `telnet` test will close the connection at login time is a syntax error in the `/etc/ppp.sys` file. If you have verified the above items you will want to check the `/etc/ppp.sys` file very closely. Common syntax errors are blank spaces after line continuation back slashes ("\") at the end of lines. Another common syntax error is ending an entry with a backslash, rather than with a colon (":").

Once you have the correct output from a `telnet` command, you can attempt to login through the serial ports. If you login with a "dumb terminal" package (like Procom) from the your Client PC, you should get the same results you see from the telnet test above. If you do not, the problem is probably not directly related to PPP or SLIP, but rather logins from serial ports in general. You should refer to the *Configuring Serial and Parallel Devices* chapter for more information.

Once you can login from the serial ports with a dumb terminal emulator, you can try the connection with your PC client PPP or SLIP package. It should login correctly and identify that the connection is established. Once you have passed the login on a standard login method or if you are using PAP and CHAP, the `ppp` program will write any error messages to the `/var/log/daemon.log` file. If you run into any troubles this should be one of the first places to look to try an analyze the problem. If the PPP clients implement PAP or CHAP authentication, you will want to make sure that feature is enabled on the BSD/OS machine. You should verify that the `ppp_direct` entry is uncommented. You should be able to verify that the login was successful by using the `last` command. If you do not get logged in you should check the username and password being used by the PPP client.

If you get through the login but the connection breaks, there is some problem between the two systems in negotiating the PPP or SLIP link. You can turn debugging on for the `ppp` user being tested by adding the following option to the `/etc/ppp.sys` entry for the user:

```
  :debug-all:
```

This tells the `ppp` program to print verbose information about the PPP negotiation. These messages will be stored in the `/var/log/ppp.debug` file. It might be able to give you some insight as to why the two machines are having trouble negotiating the PPP connection.

It is also possible to pass options to the `link-init`, `link-up`, and `link-down` scripts to gather further debugging information from them. This can be done by adding a `link-options` field to the `/etc/ppp.sys` entry. Currently, the standard scripts recognize `:link-options=-debug:`, which will send debugging output from the scripts to the file `/var/log/ppp.debug`. The "link-options" fields are passed as options to the scripts before the standard options. Modify the scripts to recognize any "link-options" field you might find valuable to help you debug problems.

Once the PPP or SLIP connection is successful you should see a ppp process running on your machine:

```
# ps a | grep ppp
  PID  TT  STAT      TIME COMMAND
27633  p9  IWs    0:00.13 -ppp (ppp)
```

You can also use `ifconfig` on the interface to verify that it is up and running. For PPP interfaces:

```
# ifconfig ppp0
ppp0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST>
     link type ppp mtu 1500 speed 57.6kbps
     inet 192.0.0.1 --> 192.0.0.2 netmask 255.255.255.0
```

For SLIP interfaces:

```
# ifconfig sl0
sl0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST>
     link type slip mtu 1500 speed 57.6kbps
     inet 192.0.0.1 --> 192.0.0.2 netmask 255.255.255.0
```

At this point you should be able to `ping` the dialed-in PC client:

```
$ ping 192.0.0.2
PING  192.0.0.2 (192.0.0.2): 56 data bytes
64 bytes from 192.0.0.2: icmp_seq=0 ttl=255 time=230.837 ms
64 bytes from 192.0.0.2: icmp_seq=1 ttl=255 time=192.924 ms
64 bytes from 192.0.0.2: icmp_seq=2 ttl=255 time=192.871 ms
64 bytes from 192.0.0.2: icmp_seq=3 ttl=255 time=183.057 ms
^C
---  192.0.0.2 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 183.057/199.922/230.837 ms
```

If you do not get any ping responses, you will want to identify how far you are getting in the `ping`. The `tcpdump` program is a fine tool to use for testing this.

To use `tcpdump`, set up two windows, either two xterms in X11, or two virtual-consoles. To use virtual consoles, change the "off" on the `ttyc*` entries of the `/etc/ttys` file to "on". The ttyc2 entry should then resemble the following:

```
ttyc2 "/usr/libexec/getty pccons" ibmpc3 on secure
```

Save the `/etc/ttys` file and execute the following command as superuser:

```
# kill -HUP 1
```

Hold down the <Alt> key and type one of the function keys (<F1>-<F9>) to switch from one screen to another.

In one of the windows you should start `tcpdump` (to monitor all of the packets into or out of the interface). For a PPP connection:

```
# tcpdump -i -n ppp0
tcpdump: listening on ppp0
```

For a SLIP connection:

```
# tcpdump -i -n sl0
tcpdump: listening on sl0
```

In the other window, start your `ping` command. If you go back to the tcpdump window you should see *echo request* and *echo reply* packets going into and out of the interface:

```
21:24:30.855049 192.0.0.1 > 192.0.0.2: icmp: echo request
21:24:30.855202 192.0.0.2 > 192.0.0.1: icmp: echo reply
21:24:31.848631 192.0.0.1 > 192.0.0.2: icmp: echo request
21:24:31.848788 192.0.0.2 > 192.0.0.1: icmp: echo reply
21:24:32.848432 192.0.0.1 > 192.0.0.2: icmp: echo request
21:24:32.848593 192.0.0.2 > 192.0.0.1: icmp: echo reply
```

This shows that 192.0.0.1 sent an echo request packet to 192.0.0.2, and that 192.0.0.2 successfully replied. If you only get lines that look like the first line above, then myhost.mydomain.com successfully sent the ping request out the interface, but the PC Client did not respond. In that case, there is a problem on the PC Client side.

If the first line does not even appear, then there is some kind of routing problem on the BSD/OS machine. One possible conclusion is that the routing table did not direct the ping to 192.0.0.2 by routing it through the `ppp0` or `sl0` interface. You will need to investigate the routing table on the BSD/OS machine (try the `netstat -rn` command).

Once you can successfully ping between the PC Client and the BSD/OS machine, you will want to try and have the PC Client ping a machine outside the BSD/OS machine, say, another machine on the local Ethernet. If that is successful, the PPP or SLIP configuration is complete. If not, track down what is happening with the ping request from the PC Client. To do this, use the two windows mentioned above and run one `tcpdump` program that monitors the interface in one of the windows. For a PPP connection:

```
# tcpdump -n -i ppp0
tcpdump: listening on ppp0
```

For a SLIP connection:

```
# tcpdump -n -i sl0
tcpdump: listening on sl0
```

In the other window run a `tcpdump` command to monitor the Ethernet interface. If your Ethernet interface is busy at all, you will probably want to add some filtering to the `tcpdump` command on the Ethernet interface so that you only see packets that are destined for or came from the PC Client's IP address. The command for that would be:

```
# tcpdump -e -n -i ef0 host 192.0.0.2
tcpdump: listening on ef0
```

where ef0 is the 3C509 Ethernet interface card and 192.0.0.2 is the IP address of the client machine. For other interface cards, use the appropriate interface name. You should be able to watch a ping request from the PC Client come in the window monitoring the `ppp0` or `sl0` interface and be forwarded out the

Ethernet, thus showing up on the Ethernet's window. If this does not happen, either IP Forwarding is off (see above) or routing is not correct on the BSD/OS machine.

If the packet does successfully go out the Ethernet port, you should then get an *echo reply* from the Ethernet. If you don't get that, there is probably something wrong on the machine on the Ethernet that you are trying to ping, or the BSD/OS machine is not providing Proxy ARP services for the PC Client. Refer to the *Custom Network Configuration* chapter for more information about IP Forwarding and Proxy ARP. Once the reply comes in the Ethernet port, it should be forwarded on out the `ppp0` or `sl0` port.

By this time, everything should be working correctly.

Configuring PPP and SLIP on BSD/OS 4.0 for Dialout Services

Both PPP and SLIP can be used to dialout to remote networks and establish IP connections with them. The `ppp` program is used to establish dialout PPP or SLIP connections. To establish a PPP or SLIP connection, execute the `ppp` command and provide the `ppp.sys` entry name that corresponds to the host to which you are connecting. The command line would look like this:

```
# ppp Palways
```

Where Palways is the `ppp.sys` entry that corresponds to the host you are trying to contact.

There is no tool that automatically configures these non-PAP/non-CHAP dialout connections. Manually modify the `/etc/ppp.sys` file to configure them. The default `ppp.sys` file, contains some entries to support PPP dialout. They are listed here for easy reference:

```
Dialout:\
        :dialout:\
        :link-init=/etc/netscripts/Dialout.login:\
        :link-up=/etc/netscripts/Dialout.up:\
        :link-down=/etc/netscripts/Dialout.down:

Pdemand:\
        :interface=0:\
        :idle-timeout=600:\
        :local-addr=LOCAL_IP:\
        :remote-addr=REMOTE_IP:\
        :phone-number=REMOTEPHONENUMBER:\
        :s0=\r:e0=ogin:f0=\r:\
        :s1=YOURLOGIN\r:e1=word:\
        :s2=YOURPASSWORD\r:\
        :tc=Demand:

Demand:\
        :dialout:\
        :link-init=/etc/netscripts/Demand.login:\
        :link-up=/etc/netscripts/Demand.up:\
        :link-down=/etc/netscripts/Demand.down:
Palways:\
        :immediate:\
        :idle-timeout=0:\
        :phone-number=REMOTEPHONENUMBER:\
        :s0=\r:e0=ogin:f0=\r:\
        :s1=YOURLOGIN\r:e1=word:\
        :s2=YOURPASSWORD\r:\
        :tc=Dialout:
Pchap:\
        :immediate:\
        :idle-timeout=0:\
        :phone-number=REMOTEPHONENUMBER:\
        :chap-allow:\
        :chap-name=YOURCHAPNAME:\
        :tc=Dialout:
Ppap:\
        :immediate:\
        :idle-timeout=0:\
        :phone-number=REMOTEPHONENUMBER:\
        :pap-peerid=YOURPAPNAME:\
        :pap-passwd=YOURPAPPASSWD:\
        :tc=Dialout:
```

Modify these entries for your specific configuration. Some suggestions as to how you might want to modify the entries for your needs are given below.

The `ppp` program uses the `gettyd` daemon to perform the actual dialing of the modem. The `gettyd` daemon must be configured to provide some serial ports for dialout purposes. The `gettystat(1)` program can be used to verify that some serial ports are available for dialout usage (with folded lines for display purposes):

```
# gettystat
console:status=UNINITIALIZED:secure:manager=init:dialin:
        speed=9600:
tty01:status=IDLE:available:manager=gettyd:dialout:
        mdmctl=DTR,RTS:speed=9600:
```

If there are no serial ports available for dialout, refer to the *Configuring Serial and Parallel Devices* chapter to configure some serial ports for dialout purposes.

Previous versions of `ppp` performed the dialing themselves. The `du` and `at` fields in the `/etc/ppp.sys` file instructed the previous versions to perform the dialing of the modem. Now that `ppp` uses the `gettyd` daemon to perform the dialing, these options are no longer required. The `ppp` program will silently ignore them if they exist.

The ppp program will pass the phone number to the gettyd daemon. The gettyd daemon will select an appropriate dialout serial port and dial the phone number. Once the remote modem answers the phone, control of that serial port will be given to the requesting ppp program by the gettyd daemon. If the use of a specific serial port is desired, a device field can be defined in the /etc/ppp.sys entry and the ppp program will request that specific port from the gettyd daemon. The gettyd daemon will dial the number passed to the gettyd daemon by the ppp program and pass control back to the ppp program once the number has been dialed and is connected.

The Palways and Pdemand entries differ only slightly. The Palways entry never times out due to an idle-timeout line, idle-timeout=0, and establishes a connection as soon as ppp is executed, immediate. The Pdemand entry will hang up if the host does not send any packets for 600 seconds, (idle-timeout=600).

Pdemand will only dial a connection when a packet is destined for the remote IP address. In order for this to operate correctly the remote IP address has to be known before the connection is established. To accomplish this, directly edit the /etc/netscripts/addr-map file. An entry should be added to assign the remote host's IP address to the Pdemand entry. A line like the following will do this:

```
  Pdemand           192.0.0.102
```

This entry assigns an IP address of 192.0.0.102 to the Pdemand entry. When a packet is destined for 192.0.0.102, the ppp program will automatically dial the remote host associated with the Pdemand entry of the /etc/ppp.sys file.

The remainder of both entries is identical. The :phone-number=*REMOTEPHONENUMBER:* identifies the phone number to call for the host you are dialing. You should replace the string *REMOTEPHONENUMBER* with the phone number of the machine you are calling. Since the gettyd daemon is actually dialing the phone, the phone number might actually be a reference to an entry in the /etc/phones file. An entry in /etc/phones can be made that assigns a name to the phone number:

```
  isp_phone     REMOTEPHONENUMBER
```

The phone number field in the /etc/ppp.sys entry can then refer to this /etc/phones entry by using the symbolic name with an "@" in front of it:

```
  :phone-number=@isp_phone:
```

Of course, this line will have a backslash as its last character if additional lines for the entry follow it.

When the string '@isp_phone' gets passed to the gettyd daemon, the gettyd daemon will reference the /etc/phones file to look up the phone number that corresponds to the @isp_phones symbolic name. The advantage of doing this is that all utilities (e.g., tip(1)) that use the gettyd daemon for dialing can use this symbolic name. If the phone number changes, the change only has to be made in the /etc/phones file and all applications that use the gettyd daemon will recognize the change.

The last thing in the entry is known as an "expect-send" string. It is this mechanism that enables you to login to the remote side and initiate the PPP or SLIP protocol, when connecting with the standard login authentication method.

It works like this:

```
... \
:s0=\r:e0=ogin:f0=\r:\
:s1=YOURLOGIN\r:e1=word:\
:s2=YOURPASSWORD\r:\
...
```

direct ppp to send a <CR> (s0=\r) then wait for a string with "ogin" in it. If such a string does not arrive before the timeout, send another <CR>, f0=\r. If the string "ogin" does arrive, then send your login name, s1=*YOURLOGIN*. Supply the correct login name for that machine rather than the string *YOURLOGIN*. Then, expect a string containing "word", e1=word (these are the last letters of "password" and "Password"). When you get that string, send your password, s2=*YOURPASSWORD*. Of course, you should supply the correct password for that host.

If the remote host requires any further commands to initiate its PPP or SLIP protocol, add additional expect-send lines to the entry.

The Ppap and Pchap entries are identical to the Palways entry, with the exception that they use PPP for user authentication, rather than an expect-send string to negotiate the standard login user authentication. Typically, for PAP authentication, the pap-peerid will be the username and the pap-passwd will be its associated password. For CHAP authentication, the chap-name will be the username that the server will use to find the secret for this host in the server's secret file. If the server has an entry in it's secret file for this host set to "Pmyhost", and a secret of "uppervale1", then set chap-name to "Pmyhost". The administrator of the server will provide the server's host identification name returned in the CHAP challenge packet. This name will be used by the CHAP client to lookup the secret in the client's secret file. If the server's host identification name is "servername", then the entry in the client's /etc/chap_md5_secret, file would be:

```
servername:ascii:uppervale1:
```

Some servers do not provide the same host identification name on each connection. The host identification name in the PPP packet can be ignored by setting the chap-secret field in the /etc/ppp.sys file. By setting chap-secret to "servername" the client will always lookup the "servername" entry in /etc/chap_md5_secret to find the secret for this connection.

The Dialout entry shown previously is referred to by all of the other sample entries, via their own tc=Dialout field. The "dialout" field identifies these entries as dialout entries. The "link-init" field identifies a script to be run when the connection is initially being established. It is this script's responsibility to assign IP addresses to the PPP or SLIP interface as well as any other necessary initialization. The default “link-init” script tries to identify the IP address to use for the `ppp.sys` entry being contacted. It refers to the `/etc/netscripts/addr-map` file to try and match the `ppp.sys` entry name to an IP address. The `/etc/netscripts/addr-map` file needs to be edited manually if you wish to assign a specific IP address to a `ppp.sys` entry. To assign *Palways* to IP address 192.0.0.104 you would add a line to `/etc/netscripts/addr-map` that looks like this:

```
Palways             192.0.0.104
```

The “link-up” field identifies a script that is run when the interface has been brought up. It is used to configure the routing tables appropriately. The default script supplied defines the remote host’s IP to be the default route. If this is inappropriate, you should create your own “link-up” script. The “link-down” field identifies a script that is run when the connection is dropped. It can be used to modify the routing table as needed once the link goes down. The default “link-down” script removes the default route if the default route corresponds to the link that is going down.

To implement dialout SLIP, all that needs to be done is select the SLIP protocol. This is done by adding an “slip” field to an entry. So, if you wanted to make Palways a SLIP connection rather than PPP, its entry would resemble this:

```
Palways:tc=Dialout:\
        :immediate:\
        :idletime-out=0:slip:\
        :pn=REMOTEPHONENUMBER:\
        :s0=\r:e0=ogin:f0=\r:\
        :s1=YOURLOGIN\r:e1=word:\
        :s2=YOURPASSWORD\r:
```

For dialout SLIP connections, it is not possible to implement dynamic IP assignment. The `ppp` program needs to know ahead of time both the remote and local IP address. For SLIP connections it is necessary to add an entry in the `/etc/netscripts/addr-map` entry for the SLIP entry:

```
Palways             192.0.0.105
```

To test a dialout PPP or SLIP connection, you can execute the `ppp` command with the `-x` option so that you can monitor the login procedure:

```
# ppp -x Palways
```

You should monitor the output to verify that the expect-send strings correctly log you into the remote system, if you are not using PAP or CHAP authentication. Once you have connected successfully you should see the following:

```
Waiting for End of session
```

If the expect-send string does not enable you to get logged into the remote machine, modify the `/etc/ppp.sys` file appropriately. You might wish to check the exact syntax of logging into the remote machine by trying to log into the remote machine using a dumb terminal emulator, like the `tip` command.

Edit the `/etc/ppp.sys` file accordingly.

Once you have the connection working, you might wish to automate it so that it establishes a connection automatically at boot time. To do this, add the following line to the `/etc/netstart` file:

```
daemon ppp -bd Palways
```

The `daemon` command will execute `ppp` and properly disassociate it from the console device. The `-bd` option tells `ppp` to run as a background process and in daemon mode. The daemon mode instructs `ppp` to redial the connection if the connection drops.

Multilink PPP

Support for multilink PPP is available in BSD/OS 4.0. The multilink PPP feature allows IP packets to be sent over multiple phone lines simultaneously to the same machines, thereby increasing overall data throughput. The kernel automatically shares the load of IP traffic across the multiple phone lines that are connecting the two machines via PPP.

There are two configuration settings in the `/etc/ppp.sys` file to enable multilink PPP for a given connection. The first is the `multilink` option. This enables the multilink PPP feature for the entry. The second setting in the `/etc/ppp.sys` entry defines an Endpoint Discriminator, which is used to label the PPP connections that should be treated as part of the same "bundle" of connections. When a new connection is made, the Endpoint Discriminator is used to identify the new connection as an addition to the previous connections rather than a separate network connection. The Endpoint Discriminator must uniquely identify the system that sends it. If it does not, PPP will incorrectly associate the wrong connection with the wrong host. Three different types of Endpoint Discriminators are supported: `ed-802.1`, `ed-ip`, and `ed-local`. The goal of each is to create a unique Endpoint Discriminator for the set of connections.

The `ed-802.1` field uses a local Ethernet address as the unique Endpoint Discriminator. This can be used on any host with an Ethernet card. The `ed-ip` uses an IP address as the unique Endpoint Discriminator. This is useful for a host that has a static IP address associated with it. A host that is assigned a dynamic IP address cannot use this. The `ed-local` option requests that the system create a unique Endpoint Discriminator. This option will be adequate for most sites.

The multilink option and the appropriate Endpoint Discriminator option are required on both the server and the client. For the server, the entry in `/etc/ppp.sys` for a client that is going to be allowed to use multilink PPP will need the multilink option and appropriate `ed` option. For example:

```
Pmulti:multilink:ed-local:tc=Dialin:
```

This would allow a PPP user called Pmulti to establish two dialup PPP connections, and the kernel would automatically share the load between those two machines across the two phone lines. The server would provide an Endpoint Discriminator to the client determined by the system, ed-local.

This is an example of an entry that uses an Ethernet address as its Endpoint Discriminator:

```
Pmulti:multilink:ed-802.1=0-0-c0-f4-36-:dc:tc=Dialin:
```

Note that a local Ethernet address is entered in hexadecimal, with dashes replacing the usual colon characters.

Finally, here is an entry using a permanent local IP address as Endpoint Discriminator:

```
Pmulti:multilink:ed-ip=10.1.2.3:dc:tc=Dialin:
```

To configure a dialout entry, the multilink option needs to be set and the method that the dialout computer will use to provide the server with an appropriate Endpoint Discriminator. Here is a Ppap entry that has been modified to allow it to establish a multilink connection with the PPP server:

```
Ppap:\
        :immediate:\
        :idle-timeout=0:\
        :multilink:\
        :ed-local:\
        :phone-number=REMOTEPHONENUMBER:\
        :pap-peerid=YOURPAPNAME:\
        :pap-passwd=YOURPAPPASSWD:\
        :tc=Dialout:
```

In order to establish two phone line connections to the remote PPP server, a separate ppp command should be executed on the client to establish each phone line connection:

```
# ppp Ppap
# ppp Ppap
```

It was possible to use the Ppap entry for both connections since there was no difference required in the entry to establish each connection. If the remote site required you to dial a different phone number to establish each connection, you would need two different entries in /etc/ppp.sys, one for each phone number. You could separate the entries like this:

```
Ppap1:\
        :phone-number=REMOTENUMBER1:\
        :tc=Ppap:
Ppap2:\
        :phone-number=REMOTENUMBER2:\
        :tc=Ppap:
Ppap:\
        :immediate:\
        :idle-timeout=0:\
        :multilink:\
        :ed-local:\
        :pap-peerid=YOURPAPNAME:\
        :pap-passwd=YOURPAPPASSWD:\
        :tc=Dialout:
```

Then, to actually establish the connection:

```
# ppp Ppap1
# ppp Ppap2
```

The key point is that each entry provides the same Endpoint Discriminator.

Each host passes its Endpoint Discriminator to the other. It is not necessary for each host to use the same method for determining the Endpoint Discriminator. The server can use `ed-802.1` while the client uses `ed-local`.

Once a multilink connection has been established, the `netstat` command can be used to see the association of ppp devices and `pif` devices:

```
# netstat -in
Name Index   MTU Speed Mtrc Address              Network
we0      1  1500    10M   0 00:20:af:e9:67:99
we0      1  1500          0 192.0.0.1            192.0.0
lo0      2  4352          0
lo0      2                0 127.0.0.1            127
sl0*     3  308           0
ppp0     4 1500   9.6k    0                      ->pif1
ppp1     5 1500   9.6k    0                      ->pif0
ppp2     6 1500   9.6k    0                      ->pif1
ppp3     7 1500   9.6k    0                      ->pif0
pif0     8 1500           0
pif0     8                0 10.0.0.1             10.0.0.3
pif1     9 1500           0
pif1     9                0 10.0.0.2             10.0.0.4
```

From the output above you can see that the IP address for a given multilink connection is associated with a `pif` interface and not a ppp interface. If you want to implement dynamic IP assignment on a multilink connection, you will need to assign an IP address for each `pif` interface in `/etc/netscripts/addr-map` file:

```
pif0    192.0.0.33
pif1    192.0.0.34
```

When a multilink connection is made, a `pif` interface will be assigned to that connection. The `/etc/netscripts/Dialin.login` script will reference the `/etc/netscripts/addr-map` file and assign the corresponding IP address. If `pif0` gets assigned to the connection, then the 192.0.0.33 IP address will be assigned to the connection. The ppp interfaces have indicators as to which `pif` interface they are bundled in. In the example above, both ppp0 and ppp2 are associated with the pif1 interface. While the ppp1 and ppp3 interfaces are associated with the pif0 interface.

Each ppp interface is managed separately. A single ppp interface can be taken down manually or might disconnect automatically due to a timeout. As long as there is one ppp device connected, the pif interface will remain up and operational. For example:

```
# ifconfig ppp0 down
# netstat -in
Name Index   MTU Speed Mtrc Address            Network
we0      1  1500    10M   0 00:20:af:e9:67:99
we0      1  1500          0 192.0.0.1          192.0.0
lo0      2  4352          0
lo0      2                0 127.0.0.1          127
sl0*     3  308           0
ppp0*    4 1500           0
ppp1     5 1500    9.6k   0                    ->pif0
ppp2     6 1500    9.6k   0                    ->pif1
ppp3     7 1500    9.6k   0                    ->pif0
pif0     8 1500           0
pif0     8                0 10.0.0.1           10
pif1     9 1500           0
pif1     9                0 10.0.0.2           10
```

The ppp0 interface was taken down, but the pif1 interface remained active because the ppp2 interface was still up. When the ppp0 interface is brought back up, it will automatically join into the pif1 multilink bundle.

It is also worth noting how traffic on multilink PPP connections is monitored. The pif interface only counts outgoing packets, never incoming packets. So, you can always expect the incoming packet count for pif connections to be zero. Each ppp interface counts both the incoming and outgoing packets that went through that interface. The pif interface counts all outgoing packets through the pif interface. So, if 7 packets were sent out the pif0 interface, you could expect the sum of the increase of outgoing packets for ppp1 and ppp3 interfaces to increase by 7. It might be that 4 packets went out ppp1 and 3 packets out ppp3.

PPP and SLIP Hints

1. In order to support PAP and CHAP a ppp_direct entry was added to the /etc/ppp.sys file. As a result there should **NEVER** be a username of ppp_direct in the /etc/master.passwd file.
2. For Trumpet WINSOCK PPP and Windows 3.1 PPP connections, set the MTU size to 512 on the Trumpet WINSOCK configuration. The symptom of this problem is that small packet type connections, like ping and telnet, work fine. However, when large packet transfers like ftp and WWW images take place, there are extremely long delays. The problem is that although WINSOCK says it can handle MTU's of 1500, it only notices the first 1000 or so bytes in a packet. So, after some timeout, it requests the unrecognized

bytes over again. Force the BSD/OS machine to set the MTU for a given route to be 512 by the following command:

```
# route change dest -mtu 512
```

Where *dest* is the IP address assigned to the PPP machine dialing in. Alternatively, set `mr#512` in the `/etc/ppp.sys` file entry for the particular host that is using a PPP client package that has this problem.

3. Some of the newer terminal servers implement PPP on the terminal server. For these terminal servers, it can be more efficient to have the terminal server implement the PPP protocol than having the terminal server telnet to the BSD/OS machine and have the BSD/OS machine implement the PPP protocol. If the BSD/OS machine implements the PPP protocol, then a packet from a PPP dialin machine destined for the internet will generate an Ethernet packet through the telnet connection and then another one from the BSD/OS machine to the router. If the PPP protocol is implemented on the terminal server, only one packet is generated directly from the terminal server to the router.

   These terminal servers usually come with software that enables the BSD/OS machine to log the PPP connect times for accounting purposes. If the terminal server is implementing the PPP protocol, the BSD/OS machine requires **no** PPP configuration.
4. If you are making PPP connections through a terminal server via `telnet` connections, be sure the terminal server is establishing an "8-bit clean" connection with the BSD/OS machine. Also, make sure the terminal server is in transparent mode. Do not use the `rlogin` protocol/command. The byte sequence:

   ```
   <377><377><s><s>
   ```

   sent as data will cause the line to freeze if you use `rlogin`!
5. If you get "ppp0: output stalled" messages on the console, it means that the PPP driver has been unable to send packets. You are probably experiencing some kind of flow control problem between the serial ports and the modem or in the `telnet` connection between the terminal server and the BSD/OS machine.
6. Due to a re-implementation of the `cmap` function it may be necessary to add a `:cmap= QS:` field to entries that previously did not require it.

Configuring a Custom Network

The BSD/OS Network configuration is performed during system installation (or by running `maxim`(8)). The `maxim`(8) configuration program performs basic network configuration for systems with Ethernet, FDDI, or Token Ring network interfaces (although FDDI and Token Ring support is not be available on the installation floppy). The section on *Basic Network Configuration* explains the information you will need to provide to configure your network interface properly for simple configurations. This chapter details configuration for the wide variety of more complicated networking configurations.

These other network interface configurations are supported, but need to be configured manually. If these network interfaces are physically installed in your system and recognized by the BSD/OS kernel (indicated by a brief message during system boot), you can learn about configuring these network interfaces in the section on *Configuring Interfaces*. If these network interfaces are not yet physically installed in your system or are not recognized by the BSD/OS kernel, start with the section on *Installing Network Interfaces*.

For information on configuring and using dial-up PPP and SLIP network interfaces, see the chapter entitled *Configuring Dialup IP Services*.

Networking Startup

Once configured, BSD/OS network operation is initiated from the `/etc/netstart` file. This file is a shell script that is run from the `/etc/rc` file when the system boots.

Four major things happen in the `/etc/netstart` file:

- The system hostname is set.
- The NIS/YP domainname (if any) is set.
- The network interfaces are configured.
- Routing is started.

Routing can be the most complicated aspect of networking startup. Few network hosts have to deal with complex routing, and systems typically point their "default" route to a local router or their Internet bandwidth provider's router, relying on one of them to handle all of the more complex routing issues. See the section on Routing below for more information.

Basic Network Configuration

If you are only configuring one Ethernet, FDDI, or Token Ring network interface and it is installed in your system and recognized by the kernel, you can respond to the questions during BSD/OS Installation, or run `maxim`(8) explicitly.

Before you do either of these, you will need some information about your local network; this is usually available from your local site administrator or Internet bandwidth provider. If you are installing BSD/OS from a CD mounted on a remote machine you will need this information during system installation. Otherwise you can defer network configuration and run `maxim`(8) after your system is installed.

Information you will need to know before you get started includes:

1. Your machine's hostname. This should be the fully qualified domain name (e.g., austin.bsdi.com). There have actually been whole documents written on choosing this name! If you are at a large site you will be constrained by local policy. This name should be a name within the domain, not the domain name itself. Hostnames may only include letters, digits, and the minus sign and must start with a letter.
2. Your IP Address (it will look something like 137.39.95.2).
3. Your network's netmask. This is often `255.255.255.0`.
4. The IP Address of your local router or your Internet bandwidth provider's gateway. This is also referred to as the "default gateway". If you are not connected to the Internet or a large network through another machine or router, then you will probably not have one of these.
5. The BSD/OS name of the network interface you are using. (e.g., we0 for WD8003/8013 & 3Com 3C503). You can get this information at system boot time or by reviewing `/var/db/dmesg.boot`.
6. The NIS/YP domainname, if you intend to run your system as an NIS client.

If you are connecting to a network via some kind of point-to-point protocol (Frame Relay, HDLC, PPP, SLIP), then you will also need to know the IP address of your bandwidth provider (which will almost certainly be the answer to item #4).

Configuring Network Interfaces

Before you can configure and enable a network interface, it must be physically installed in your system and recognized by the BSD/OS kernel. For more information on installing and locating network interfaces, see the section below on *Installing Network Interfaces*.

Network interfaces are configured and enabled by commands in the `/etc/netstart` file (see `/etc/netstart.proto` for examples). LAN interfaces (Ethernet, FDDI or Token Ring) are configured by editing variable definitions in this file. The `maxim`(8) configuration program performs all these tasks for you. Other interfaces are configured by adding `ifconfig`(8) commands to this file. Comments in this file indicate where these changes should be made. In general, it is best to add them after the configuration commands for the first Ethernet, FDDI, or Token Ring network interface and before the loopback interface (lo0) is configured.

There are several pieces of information required to construct an `ifconfig` command to configure and enable a network interface. The first piece of information you will need is the BSD/OS name of the network interface. This name is displayed at boot time; if you missed it, you can use `dmesg`(8) to display the boot messages. Some examples are `ef0` for 3Com Etherlink III Ethernet interface, `fpa0` for a DEC FDDI controller, and `ntwo0` for an SDL Communications RISCom/N2 high-speed synchronous serial network interface.

Configuring Ethernet, FDDI, or Token Ring Network Interfaces

For Ethernet, FDDI or Token Ring network interfaces you will also need:

- The IP address of your network interface.
- The IP network mask (netmask) for your network. If you are using the default netmask for your network type (class A, B, or C), you will not need to specify it explicitly.
- Network interface specific media flags.
  Some network interfaces support more than one media type. On these, media flags are used to select the desired one. For example, some 3Com Etherlink III network interfaces can support BNC, TP, and AUI media (i.e., cables). The media flags are documented in the `ifconfig`(8) manual page. `Ifconfig`(8) can also be used with the `-m` flag to display a list of available media options on a given network interface. The `maxim`(8) configuration program figures these out automatically.

Once you have obtained all this information, the easiest path is to use the `maxim`(8) network configuration screens to configure this network interface. Alternatively, you can build the interface definitions to be inserted into the `/etc/netstart` file. The format of an interface definition is:

```
ipaddr_if_name="ipaddr"
netmask_if_name="netmask"
linkarg_if_name="mediaflags"
additional_if_name=
```

For example, given a 3Com Etherlink III (`ef0`) with an IP address of 10.1.1.1, and netmask of 255.255.255.0 using the BNC (10base2) connector:

```
# ef0::
ipaddr_ef0=10.1.1.1
netmask_ef0=255.255.255.0
linkarg_ef0="media bnc"
additional_ef0=
```

The `maxim`(8) program uses the comment line when it parses the file.

Another example is the same IP address and netmask configured on a Megahertz PC Card Ethernet interface (`mz0`), which does not have any software configurable media selection:

```
# mz0::
ipaddr_mz0=10.1.1.1
netmask_mz0=255.255.255.0
linkarg_mz0=
additional_mz0=
```

In both cases, it is necessary to update the `interfaces=` line in the `/etc/netstart` file to add the interface name. Also the primary interface should be listed on the `primary=` line. Here is an example entry (for a system that uses an Etherlink III):

```
primary="ef0"
interfaces="ef0"
```

Here is an example entry for a system that has an `ef0` and `mz0` interface with `ef0` as the primary:

```
primary="ef0"
interfaces="ef0 mz0"
```

Configuring Synchronous Serial Network Interfaces

For serial network interfaces, such as the SDL Communications RISCom family of interfaces, you will need to know

- The local IP address of your serial network interface.
  This address can frequently be the same as the address of one of the Ethernet, FDDI or Token Ring network interfaces on your system.
- The IP address of the remote system to which your serial interface is connected.
- The link type.
  The high-speed serial synchronous network interfaces can be configured to run the PPP link-layer protocol, the Frame Relay link-layer protocol, or the cisco HDLC link-layer protocol. See the section below on *Link Type* for more information.
- Network interface specific link flags.
  See the manual pages for the particular network interface you are using.

Once you have obtained all this information you either build an `ifconfig` command (for Cisco style HDLC and Frame Relay connections) or use the `pppattach`(8) command (for PPP connections). Frame Relay connections might also require the use of the `frconfig`(8) command. The format of the `ifconfig` command (shown here folded) is:

```
ifconfig interface inet ipaddr destination
     remote_ipaddr linktype linktype linkflags
```

For example, given an SDL Communications RISCom N2 with the local IP address of 10.1.1.1 and a remote IP address of 10.2.3.1 running cicso HDLC, you would use something like this (folded):

```
ifconfig ntwo0 inet 10.1.1.1 10.2.3.1
                           linktype chdlc
```

To configure the same connection using Frame Relay, the command would simply become:

```
ifconfig ntwo0 inet 10.1.1.1 10.2.3.1 linktype fr
```

It is also possible to configure a Frame Relay link to act more like an Ethernet without broadcast (Multiple Destination No Broadcast, or MDNB). In this mode you assign a single address to the Frame Relay link, just as you would for an Ethernet link. You must first, however, set the Frame Relay mode to MDNB (it defaults to Point-to-Point (P2P)). This is done with the `frconfig`(8) command. Since the `frconfig`(8) command only applies to interfaces with the linktype of “frelay”, you will need to first set the link type, the the frame relay mode, and finally the address. Additionally, if your default route should be directed at a specific DLCI (Data Link Connection Identifier) (say, 16 for instance) you may also specify that on the `frconfig` command:

```
ifconfig ntwo0 linktype frelay
frconfig ntwo0 mode MDNB default 16
ifconfig ntwo0 inet 10.1.1.1 netmask 255.255.255.0
```

See the `frconfig`(8) manual page for other options that may be set with the `frconfig`(8) command.

If, instead, you are using the PPP protocol, the following `pppattach`(8) command would be used:

```
pppattach ntwo0 10.1.1.1 10.2.3.1
```

You would then add this command (either the ifconfig command or the pppattach command) to the `/etc/netstart` file as directed by comments in that file.

If your system's only network interface is a serial network interface, you might experience delays in packets addressed to yourself, especially when using X Windows applications that use TCP/IP to communicate with the X server running on your console. This is because the system does not recognize the local address on serial interfaces on outbound packets. The best fix for this is to add this address as an alias on your loopback interface. Using the above example, you would also add another `ifconfig` command to the end of the `/etc/netstart` file:

```
ifconfig lo0 add 10.1.1.1
```

Editing the `/etc/netstart` *file*

Once you have determined the proper parameters to the `ifconfig` command, edit the `/etc/netstart` file. See `/etc/netstart.proto` for a good starting point for the `netstart` file. Add the LAN interface definitions at the beginning of the file and add commands to start additional synchronous serial network interfaces after the commands that configure the primary Ethernet, FDDI, or Token Ring network interface and before the loopback interface is configured. Comments in the file direct you to the appropriate places. Changes to this file will take effect when you reboot. To have them take effect before rebooting, issue the commands as root.

Link Type

All network interfaces are configured with a link type that describes the underlying link-layer network protocol being used. The link type in use on a particular network interface is displayed as part of the output of the `ifconfig` command. Most network interfaces only support one link type, such as the Ethernet, FDDI, Token Ring, and asynchronous SLIP and asynchronous PPP interfaces in this example (upright bold is used for emphasis):

```
% ifconfig ef0
ef0: ether flags=9863<UP,BROADCAST,NOTRAILERS,RUNNING, \
                              SIMPLEX,LINK0,MULTICAST>
```

```
        inet 10.40.196.10 netmask 255.255.255.0 \
                              broadcast 10.40.196.255
% ifconfig fpa0
fpa0: fddi flags=8863<UP,BROADCAST,NOTRAILERS,RUNNING, \
                              SIMPLEX,MULTICAST>
        inet 205.230.237.2 netmask 255.255.255.240 \
                              broadcast 205.230.237.15
% ifconfig te0
te0: token_ring flags=2043<UP,BROADCAST,RUNNING,LINK1>
        inet 1.1.1.1 netmask 0xff000000 \
                              broadcast 1.255.255.255
% ifconfig sl0
sl0: slip flags=9011<UP,POINTOPOINT,LINK0,MULTICAST>
        inet 10.40.200.19 --> 10.40.200.24 \
                              netmask 255.255.255.255
% ifconfig ppp0
ppp0: ppp flags=8010<POINTOPOINT,MULTICAST>
```

The output in the above example was folded for display purposes.

On some classes of network interfaces it is possible to chose between more than one link-layer protocol. This is most common with synchronous serial network interfaces, such as the RISCom H2, N1, and N2 interfaces. In fact, setting the link type is a requirement since the default is **none**; one must be chosen before the link can be used. For example:

```
% ifconfig ntwo0
ntwo0: flags=8010<POINTOPOINT,MULTICAST>
        link type none mtu 1500

# ifconfig ntwo0 linktype ppp
# ifconfig ntwo0
ntwo0: flags=8051<POINTOPOINT,RUNNING,MULTICAST>
        link type ppp mtu 1500

# ifconfig ntwo0 linktype chdlc
# ifconfig ntwo0
ntwo0: flags=8051<POINTOPOINT,RUNNING,MULTICAST>
        link type chdlc mtu 1500

# ifconfig ntwo0 linktype fr
# ifconfig ntwo0
ntwo0: flags=8051<POINTOPOINT,RUNNING,MULTICAST>
        link type frelay mtu 1500
```

For PPP you typically use the `pppattach`(8) command (which in turn uses the `ppp`(8) command) to start a PPP session. The `ppp`(8) command will automatically set the linktype to ppp.

On these classes of network interfaces, the link type can be changed by specifying the `linktype` argument to the `ifconfig` command. The link types chdlc and none can be specified at any time; if the link is in use, the new link layer will be enabled immediately. If a synchronous serial network interface was up and operating when the link layer was changed, connectivity will not be re-established until the remote end is also changed.

The Parallel Interface (PIF)

The Parallel Interface (pif) driver is a logical network interface that is used to bundle several physical point-to-point interfaces as a single logical interface.

If you are using the PPP link-layer protocol, the PPP Multilink Protocol will automatically take care of allocating a PIF interface. See the section on Multilink PPP in the *Configuring Dialup IP Services* chapter for more information.

If you are using Cisco HDLC link-layer protocol, then you can use `ifconfig`(8) to link together the interfaces. Configuring the interfaces is similar to configuring them individually, except that you assign the IP address to the PIF interface, and link the physical interface to the PIF interface. For example:

```
# ifconfig ntwo0 linktype chdlc pif pif0 up
# ifconfig ntwo1 linktype chdlc pif pif0 up
# ifconfig pif0 192.107.121.10 192.35.65.29
```

The first two lines link the `ntwo0` and `ntwo1` interfaces to `pif0`, and mark them as up. The third line assigns the IP addresses to the `pif0` interface.

If you are using Cisco HDLC or Frame Relay link-layer protocol, you can also change how the PIF driver chooses to distribute outgoing packets among the connected interfaces. The default value is "next-idle", which will use a round-robin technique to search the list of interfaces for one that is idle. This spreads the traffic most evenly across the interfaces, even if they are of different speeds and, thus, is most desirable. The "next-up" value will just round-robin the packets, ignoring whether or not the interface is already doing output.

```
# ifconfig pif0
pif0: flags=8000<MULTICAST>
    link type propmux mtu 576
    pifflags next-idle
# ifconfig pif0 pifflags next-up
# ifconfig pif0
pif0: flags=8000<MULTICAST>
    link type propmux mtu 576
    pifflags next-up
```

A PIF interface inherits the characteristics of the interfaces that are linked to it. For example:

```
# ifconfig ntwo0
ntwo0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST>
    link type chdlc mtu 1500
# ifconfig pif0
pif0: flags=8000<MULTICAST>
    link type propmux mtu 576
    pifflags next-idle
```

If we then attach `ntwo0` to `pif0`, and bring up `pif0`, we have:

```
# ifconfig ntwo0 pif pif0 up
# ifconfig pif0 10.0.0.1 10.0.0.2
```

```
pif0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST>
    link type propmux mtu 1500
    pifflags next-idle
    inet 10.0.0.1 --> 10.0.0.2 netmask 255.0.0.0
# ifconfig ntwo0
ntwo0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST>
    link type none mtu 1500
    attached to pif0
```

The 10.*.*.* network is a Class A network used for examples.

Routers

A router (sometimes referred to as a gateway) forwards packets from one network to another. For sites that use BSD/OS as their Internet Router, one purpose of the machine is to ensure the packets get forwarded from the Internet to the internal clients and vice-versa. Some sites, however, are concerned about the potential security problems inherent in letting Internet users send packets to any machine on their network. They do not pass packets directly from the Internet to their network but instead use some other scheme that either stops or filters packets between the Internet and their internal network (e.g., disabling internal routing and connecting to the server for Internet access, implementing proxy servers, or building/acquiring a firewall router). You must choose an appropriate method of Internet communication for your site.

BSD/OS has the capability to act as a router once you choose to enable it like this:

```
# sysctl -w net.inet.ip.forwarding=1
```

Disable it like this:

```
# sysctl -w net.inet.ip.forwarding=0
```

or query its value like this:

```
# sysctl net.inet.ip.forwarding
```

To enable this change permanently, uncomment the appropriate line in the `/etc/rc.local` file. That file also contains other `sysctl`(8) commands to configure other networking related configuration variables.

Note: network configuration commands in `/etc/rc.local` are also executed when running the `/etc/netstart` script. In the future, these commands will be moved to the `/etc/netstart` file itself.

You can also recompile the kernel with

```
options GATEWAY
```

and IP forwarding will automatically be enabled (after reboot of the new kernel). Additionally, certain kernel data structures will be resized more appropriately for router machines.

Routers with many thousands of routing table entries (such as an ISP's router running `gated`(8) to speak the BGP routing protocol with an up-stream network) would need to enable support for large routing tables as directed by comments in the kernel configuration file.

BSD/OS has several networking security features that can help defeat and detect some types of denial of service attacks.

The first feature is a limit on the number of fragmented IP packets in the IP reassembly queue. The default limit is 200 and can be changed with the `sysctl(8)` variable `net.inet.ip.maxfragpackets`. To change the limit of the number of packets on the IP reassembly queue add a command like the following to the end of the `/etc/netstart` file. This example would reduce the limit on outstanding fragments to 100:

```
 sysctl -w net.inet.ip.maxfragpackets=100
```

The second feature is an optional test to insure that packets are received on the expected interface. This feature looks up the route back to the source of received IP packets. If there is no route to the source available, or the packet did not arrive on the expected interface the packet is discarded. The expected interface is the one that would be used to send a packet back to the reported source of the packet.

IP source address verification should not be used when concurrent alternate paths exist from the BSD/OS system where this feature is enabled, as this might cause valid packets to be discarded. For example, a small ISP that has one connection to a backbone network and one connection to each of its clients could enable this feature. If the same ISP has two connections to a backbone network, or one connection to each of two backbone networks they should not enable this feature. In this situation the router can use BSD/OS IP Filters (see *Configuring IP Filters*).

This feature is enabled via the `net.inet.ip.sourcecheck sysctl`(8) variable or by adding the `IPSOURCECHECK` option when building a kernel. For example, to enable IP source address verification, remove the comment in front of the appropriate command in the `/etc/rc.local` file.

IP source address verification is a valuable tool for protecting against some forms of IP-spoofing as described in CERT advisory CA 96.21, "TCP SYN Flooding and IP Spoofing Attacks". The full text of this advisory is available as `ftp://info.cert.org/pub/cert_advisories/`
`CA-96.21.tcp_syn_flooding`. If you are a service provider, using IP source verification will protect your customers against attacks from the Internet that appear to be coming from your customers' networks, and it will ensure that packets sent from your customers' networks have a source address on your customers' networks (preventing them from spoofing source addresses and/or attacking others).

The IP source address verification code will log a message when discarding a packet. To prevent a large number of these packets from using an excessive amount of disk space log messages are limited to one per IP address per time interval. The time interval defaults to five seconds and can be configured with the `net.inet.ip.sourcecheck_logint` `sysctl`(8) variable. A value of zero disables the time interval.

A third feature can disable the forwarding of source routed IP packets. Although useful as a debugging tool (see `traceroute`(8)), source routed IP packets can present a security threat to some IP firewalls. To disable the forwarding of these packets, uncomment the line that sets the value of the `net.inet.ip.sourcecheck` `sysctl`(8) variable in the `/etc/rc.local` file and reboot – or, uncomment the assignment of the `IPFORWARDSCRT` variable in the configuration file when building a kernel.

The fourth feature prevents forwarding of any fragment with an offset of 1. Without this it is possible to spoof many firewalls by sending an initial TCP SYN packet fragment that can make it through the firewall and then overwrite all but the first 4 bytes with new data with a new fragment. BSD/OS clients are not subject to this attack, but other OS's may be. This feature is enabled via the net.inet.ip.do_rfc1858 flag. This is turned on by default when the kernel is built with BSD/OS IP Filters.

The fifth feature is BSD/OS IP Filters. This allows filtering on any aspect of the packet and is described in more detail in the section *Configuring IP Filters*).

Routing

When communicating with hosts that are not on a local network, a BSD/OS system requires routing information to tell it where to send IP packets. Routing configuration can be a very complex; this section will only describe a few common situations.

Configuring a Default Route

If the BSD/OS system is on a Local Area Network (a LAN is usually an Ethernet, FDDI, or Token Ring network) that has only one router to the outside world (i.e., the rest of a company/campus and/or the Internet), a default route can be installed pointing to this router, making it the "default router". The IP address of this default router might be specified via `maxim`(8), or as the value of the `defroute=` variable in the `/etc/netstart` file.

When a BSD/OS system is configured to use PPP to dial out to connect to the outside world, the default PPP "Dialout" scripts will automatically install a default route if one does not already exist. This route will be removed when the PPP connection is terminated. See the *Configuring Dialup IP Services* chapter for more information on PPP.

If the BSD/OS system is on a LAN with more than one router, using a default route might be sufficient. BSD/OS supports the use of ICMP redirect messages that allow one router to request that the BSD/OS system use another router to communicate with a specific host. In addition, BSD/OS now will automatically remove these entries after a time-out period so it is no longer necessary to run `routed`(8) or `gated`(8) to manage the routes created by ICMP redirects.

Dynamic Routing

If the BSD/OS system is on a LAN with more than one router and the default route is not adequate, it might be necessary to learn routing information by using `routed`(8) in quiet mode. This is done by setting `routedflags=-q` in the `/etc/netstart` file and rebooting. Routed will monitor the LAN for RIP (v1 and v2) routing information and participate in the Router Discovery protocol as a client (which is a strange vocabulary; most people would think of it as a host).

Note that in 3.0, the `routed` server is implemented using `gated`(8) with a default configuration. Do not be concerned if you see `gated` running rather than `routed`.

When a BSD/OS system has more than one network interface (two Ethernet interfaces, for example) it can function in one of two ways. If the BSD/OS system is not supposed to forward packets between the two networks, it is considered a multi-homed host. If there are routers on only one of the networks, then a default router can be configured. If there are routers on more than one network, then `routed`(8) can be used in quiet mode as described in the previous paragraph.

If you want a BSD/OS system to forward packets between multiple LANs it needs to be configured as a router. Besides enabling the forwarding of IP packets (as described in the Routers section above), it might be necessary to provide routing information. To enable routed, set `routedflags=-s` in the `/etc/netstart` file and reboot.

It is also possible to start `routed`(8) with no flags. In this case, `routed` will supply routing information if IP forwarding is enabled (as described in the Routers section above) and there are two or more LAN interfaces, otherwise it will just monitor the LAN for routing information. Refer to the `routed`(8) documentation for more details on the protocols it supports.

Routing configuration can quickly become very complicated. Situations requiring routing over serial lines (CHDLC, Frame Relay, PPP and SLIP) and routing protocols other than RIP are beyond the scope of this document. Your network administrator or network bandwidth provider might be able to help with more complex configurations. You might also want to refer to the `gated`(8) documentation.

Proxy ARP

If configuring a SLIP or PPP client using an IP network address inside your Ethernet, FDDI, or Token Ring's IP network, you will need to set up Proxy ARP (or be really careful with your `gated`(8) configuration). This is normally performed by the `config_dialin`(8) program, which configures the client to use the `/etc/netscripts/Dialin.up` and `/etc/netscripts/Dialin.down` scripts.

This can also be done with the command:

```
# arp -s 12.34.56.70 88:88:88:88:88:88 pub
```

Where 12.34.56.70 is the IP address of the connected machine and 88:88:88:88:88:88 is the Ethernet, FDDI, or Token Ring MAC Address of the server (and, when you have things working, put this `arp` command in the `/etc/netstart` file).

You can obtain the MAC address of the Ethernet, FDDI, or Token Ring network interface on your server by giving your IP address to the `/etc/netscripts/myether` program, for example:

```
# /etc/netscripts/myether 240.124.135.1
00:c0:f2:00:1e:c0
```

Alternatively, you can obtain the MAC address of the server with the command `netstat -in`. For example, given:

```
% netstat -in
Name Index   MTU Speed Mtrc Address                 Network
tn0      1  1500    10M    0 00:c0:f2:00:1e:c0
tn0      1                 0 240.124.135.1          240.124.135/24
lo0      2  4352           0
lo0      2                 0 127.0.0.1              127
ppp0     3  1500           0
ppp0     3  1500           0 240.124.135.2          240.124.135.3
% ifconfig ppp0
ppp0: flags=9051<UP,POINTOPOINT,RUNNING,LINK0,MULTICAST>
 link type ppp mtu 1500 speed 115.2kbps
 inet 240.124.135.2->240.124.135.3 netmask 255.255.255.0
```

run the command:

```
% arp -s 240.124.135.3 0:c0:f2:0:1e:c0 pub
```

to establish the correct ARP behavior.

Debugging Network Problems

If you suspect network problems, the simplest test is to use `ping`(1) to try to send a packet to another host on the network. Start with a nearby machine (one on the same network) and work your way further out.

Until you get the network running and the `named`(8) name daemon set up, you will probably need to use the `-n` flag with networking utilities that print hostnames so they will not try to resolve IP addresses to symbolic hostnames. Otherwise the programs might hang trying to resolve the name.

Use the command `netstat -rn` to view your routing table. Here is the routing table from BSDI's austin.bsdi.com machine:

```
default             137.39.95.1         UGS      1500 ntwo0
127                 127.0.0.1           URS      4352 lo0
127.0.0.1           127.0.0.1           UH       4352 lo0
```

```
137.39.95.1        137.39.95.2        UH       1500 ntwo0
137.39.95.2        127.0.0.1          UGHS     4352 lo0
```

The "default" entry points to the bandwidth provider to whom all non-local packets are forwarded. If you are on an Ethernet, there are probably multiple hosts on your network and your machine will send packets to the local Ethernet directly, forwarding packets to other networks only when they are not destined for the local network. 127.0.0.1 is the loopback network. Note that 137.39.95.2 (the local machine in this example) is routed to it, so that packets generated on the local machine that are for the local machine are quickly forwarded without using the network.

The austin.bsdi.com site is on a point-to-point link so there is only one other host on the "network" to `ping`.

The `ifconfig` command reports (and changes) the configuration of a network interface:

```
# ifconfig ntwo0
ntwo0: ppp flags=51<UP,POINTOPOINT,RUNNING>
    link type ppp mtu 1500 speed 115.2kbps
    inet 137.39.95.2 --> 137.39.95.1 netmask 255.255.255.252
# ping -c 1 137.39.95.1
PING 137.39.95.1 (137.39.95.1): 56 data bytes
64 bytes from 137.39.95.1: icmp_seq=0 ttl=255 time=37 ms
--- 137.39.95.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 37.061/37.060/37.061 ms
```

If you do not use the `-c 1` option, `ping` will keep trying forever. In that case, kill it by typing your interrupt character (usually <Control-C>).

You can also use the `traceroute`(8) command to print out the path a packet takes on its journey from here to there (over the Internet it can be a long journey, indeed).

Another tool that can be helpful for network debugging is `tcpdump`(1). The `tcpdump` command enables you to see all the traffic passing through your machine or even all traffic on the network to which it is connected. See the manual page for more information. Note, `tcpdump` can consume a large amount of resources on your machine when running.

Network Configuration Problems

In a severe case, configuration problems will manifest themselves as the appearance of total network failure (i.e., nothing works). Typically, these only occur when you are initially setting up the system. After that, you might find that many services work while some others don't.

It is difficult to debug configuration problems via documentation. You can examine manual pages that seem related and usually glean some information there.

Watch out for having TCP header compression enabled on a SLIP or PPP line that doesn't support it. Also watch out for "autocompression", since it can sometimes be enabled by line noise.

Check your DNS server, sometimes a dead name server can cause the appearance of a dead network; always keep a numerical IP address handy for using with `ping`.

Helpful commands are `ifconfig`, `netstat -i`, and `netstat -s`.

Routing Problems

These are usually seen as the ability to talk to some hosts but not others. Try using `ping` to contact your system, then a nearby host, then your local router. Use `traceroute` to find out how far the packet is going.

Helpful commands are `ping`, `traceroute`, and `netstat -rn`.

Probably the most useful thing to understand when debugging routing problems is that your routing information only tells your system how to route packets to a remote destination; the remote destination needs routing information to send packets back to your system.

This is especially important when using PPP on a BSD/OS system to connect your LAN to the Internet. Not only do you need a default route via the PPP link to direct your packets to the Internet, but the Internet also needs to be informed of how to reach your LAN. Usually, your ISP should solve this problem for you.

Connectivity Problems

Connectivity problems are caused by packet filters, broken cables, noisy transmission lines, and crashed servers (among other things). Any server your system relies on heavily can cause problems. NFS and DNS server problems are the most painful. If a remote DNS server isn't reliable, then you should consider running `named` locally.

Installing Network Interfaces

There are two main classes of network interfaces. The most common are those supporting shared media such as Ethernet, FDDI, and Token Ring. Network interfaces for high-speed synchronous serial lines (usually HDLC) are also widely used.

For either type it is first necessary to insure that your particular network interface is supported by this release of BSD/OS (see the chapter on *Supported Hardware and Configuration* for a list of supported network interfaces). Also, look for a manual page specific to your network interface. For example, if you have a *RISCom/N2* high-speed synchronous serial network interface you can use `man -k` to search for manual pages related to *RISCom* network interfaces (shown here folded):

```
% man -k RISCom
n2setup (8) - configure SDL Communications RISCom/N2
```

```
                        family of interfaces
ntwo (4) - SDL Communications RISCom/N2 high-speed
                        synchronous interface
rc (4) - SDL Communications RISCom/8 8-port serial
                        multiplexor
rh (4) - SDL Communications RISCom/H2 dual high-speed
                        synchronous interface
rn (4) - SDL Communications RISCom/N1 high-speed
                        synchronous interface
% man -s 4 ntwo
[...]
% man -s 8 n2setup
[...]
```

The BSD/OS kernel comes configured with support for quite a few network interfaces, but it might be necessary to change the kernel configuration and build a new kernel with support for a particular network interface (or set of interfaces). See the *Rebuilding the Kernel* chapter for details of editing a kernel configuration file and building a new kernel.

BSD/OS network interface names end with a digit that identifies a particular interface from multiple instances of that interface. The first network interface of a given type will have an identifier ending with 0 (zero); the second will end with 1 (one); the third with 2, and so on. For PCI, EISA, these numbers are determined by the slot in the motherboard the network interfaces occupies. For most ISA network interfaces these numbers are associated with the boards based on the configured port address of the network interface. Some networking interfaces (usually ISA-based interfaces) even assign the lowest unit number to the interfaces with the lowest MAC (physical) address.

Sometimes there is more than one network interface on a single card. For example, the RISCom/N2 can have two high-speed synchronous serial interfaces. In this case, the kernel configuration file entry for the first card, ntwo0, will be associated with both the ntwo0 and ntwo1 network interfaces.

When installing ISA network interfaces, it is important to avoid address and IRQ conflicts. See the *Hardware Configuration* chapter for more information.

Next, the network interface should be physically installed into your system. It might be necessary to configure certain switches and jumpers on the network interface; refer to the hardware documentation, the related manual pages, and the kernel configuration file for details such as port numbers and IRQs.

Note that when installing and/or removing EISA network interfaces it is necessary to boot DOS and run the EISA configuration utility distributed with your computer. When installing new EISA network interfaces, you will also need the floppy distributed with the network interface, which contains the necessary EISA configuration information. See the hardware documentation for your computer/mother board for more information.

When the system is rebooted, you should see a one-line message indicating the presence of the network interface. If you miss the boot messages, you can view the `/var/db/dmesg.boot` file to see them again.

```
% grep ntwo0 /var/db/dmesg.boot
ntwo0 at isa0 iobase 0x320 irq 5 maddr
                     0xe0000-0xe7fff board type = N2
```

(shown here folded).

Once the network network interface is installed, it can be configured as described above in *Configuring Network Interfaces*. Some interfaces have programs that need to be run before the network interface can be configured and loaded. These programs should be referenced in the manual pages for the network interface. For example, the *RISCom/N2* network interface has an optional on-board CSU that might need to be configured with the `n2setup`(8) program. You should create a script to perform configuration of such network interfaces in the directory `/etc/rc.hardware`. See the README file in that directory for general instructions; see `/etc/rc.hardware/README.`*`device-type`* for information about a specific device type (e.g., *`ntwo`*). The `/etc/rc.hardware` file includes sample scripts for common configurations in files named `/etc/rc.hardware/proto.`*`device-type.`* Often, you can copy a prototype script to a name like `5.ntwo.0` to configure the ntwo0 device. The README files explain the naming conventions.

Installing Multiple Network Interfaces

The BSD/OS kernel is configured with support for quite a variety of network interfaces at various bus addresses. The exact network interfaces and bus addresses are described in the file `/sys/i386/conf/GENERIC`. This file and a description of the procedure necessary to build a new kernel are described in the *Rebuilding the Kernel* chapter as well as the *Building Kernels on BSD/OS* document (`man -M bsdi config`) and the `config`(8) manual page.

Installing Multiple Network Interfaces of Different Types

If you are using network interfaces of different types, chances are good that you only have to install the second network interface physically and reboot for it to be recognized by BSD/OS.

Installing Multiple Network Interfaces of the Same Type

If you are installing more than one network interface of the same type, it might be necessary to build a new BSD/OS kernel. This might also be necessary if you wish to use a supported network interface that is not configured into the distributed BSD/OS kernel. In general, support for up to three EISA and PCI network interfaces of each type is configured into the kernel because their addresses are automatically configured.

To configure support for two ISA network interfaces of the same type you will need to find the entry or entries for the first network interface in `/sys/i386/conf/GENERIC`. If only one entry for the network interface exists, such as for the 3Com 3C509 Etherlink III, you will see a single line similar to this:

```
ef0 at isa? port 0x250
```

Duplicate this line, changing the unit number to 1 and the port value to the unused bus address you wish to use for the second card. Be sure to read the manual page and hardware documentation for the network interface you are using so you can avoid conflicts with other system interfaces. For a second 3C509, you might specify:

```
ef0 at isa? port 0x250
ef1 at isa? port 0x260
```

In some cases, there are entries in `/sys/i386/conf/GENERIC` for all of the possible addresses of a type of network interface. One good example is the Western Digital/SMC WD8003 and WD8013 family of network interfaces:

```
we0 at isa? port 0x280 iomem 0xd0000 iosiz 16384
we0 at isa? port 0x2A0 iomem 0xd0000 iosiz 16384
we0 at isa? port 0x2C0 iomem 0xd0000 iosiz 16384
we0 at isa? port 0x2E0 iomem 0xd0000 iosiz 16384
we0 at isa? port 0x300 iomem 0xd0000 iosiz 16384
we0 at isa? port 0x320 iomem 0xd0000 iosiz 16384
we0 at isa? port 0x340 iomem 0xd0000 iosiz 16384
we0 at isa? port 0x360 iomem 0xd0000 iosiz 16384
we0 at isa? port 0x380 iomem 0xd0000 iosiz 16384
we0 at isa? port 0x3A0 iomem 0xd0000 iosiz 16384
# 0x3C0 is not usable with color pccons
# we0 at isa? port 0x3C0 iomem 0xd0000 iosiz 16384
we0 at isa? port 0x3E0 iomem 0xd0000 iosiz 16384
```

In this case, find the entry for the address you configured for the second interface's card, change we0 to we1, and choose another memory address. For example, if the second network interface is configured for the port address 0x2E0, change the above to read:

```
we0 at isa? port 0x280 iomem 0xd0000 iosiz 16384
we0 at isa? port 0x2A0 iomem 0xd0000 iosiz 16384
we0 at isa? port 0x2C0 iomem 0xd0000 iosiz 16384
we1 at isa? port 0x2E0 iomem 0xd8000 iosiz 16384
we0 at isa? port 0x300 iomem 0xd0000 iosiz 16384
we0 at isa? port 0x320 iomem 0xd0000 iosiz 16384
we0 at isa? port 0x340 iomem 0xd0000 iosiz 16384
we0 at isa? port 0x360 iomem 0xd0000 iosiz 16384
we0 at isa? port 0x380 iomem 0xd0000 iosiz 16384
we0 at isa? port 0x3A0 iomem 0xd0000 iosiz 16384
# 0x3C0 is not usable with color pccons
# we0 at isa? port 0x3C0 iomem 0xd0000 iosiz 16384
we0 at isa? port 0x3E0 iomem 0xd0000 iosiz 16384
```

Installing Network Interfaces That Are Not Pre-configured

Some supported network interfaces are not configured into the distributed generic BSD/OS kernel. Frequently, the reason for this is to save space in the distributed kernel (which must fit on a floppy disk). Entries in `/sys/i386/conf/GENERIC` for these interfaces will start with a # (pound sign), denoting that they are comments and are to be ignored. If you are using only one of these, it is necessary to remove the # from the beginning of the line. For example, to add support for a DEC DEFPA PCI FDDI network interface, find the configuration line for it:

```
# DEC PCI FDDI (DEFPA-XX)
#fpa0 at pci?
```

and change it to:

```
# DEC PCI FDDI (DEFPA-XX)
fpa0 at pci?
```

To add a second one, add another line:

```
# DEC PCI FDDI (DEFPA-XX)
fpa0 at pci?
fpa1 at pci?
```

Another example is the SDL Communications RISCom/N2, which needs to be jumpered for the correct address. All examples here are shown folded. To add support for one at the address listed in `/sys/i386/conf/GENERIC`:

```
#ntwo0 at isa? port 0x300 iomem 0xe0000
                        iosiz 32768 flags 0
```

remove the leading #:

```
ntwo0 at isa? port 0x300 iomem 0xe0000
                        iosiz 32768 flags 0
```

To add support for a second RISCom/N2 at 0x320, add another line with a new unit name and port (it’s OK to use the same address):

```
ntwo1 at isa? port 0x320 iomem 0xe0000
                        iosiz 32768 flags 0
```

Configuring IP Version 6

Kernel Configuration

BSD/OS 4.0 now includes support for IP Version 6 (IPv6) and IPSEC. There are several kernel config options related to this:

- **INET6** This is the IPv6 code, and may be defined without any other options.
- **KEY** This is the key management code.
- **IPSEC** This enables the IP security code, and allows for authentication. KEY must also be defined.
- **IPSEC_ESP** This enables the framework for encryption with IPSEC, and requires that IPSEC also be defined.
- **IPSEC_DES** This enables the DES method of encryption, and requires that IPSEC_ESP also be defined.
- **INET6_DEBUG** Debugging code for INET6.
- **KEY_DEBUG** Debugging code for KEY.
- **IPSEC_DEBUG** Debugging code for IPSEC.

Note: Only the domestic distribution contains the encryption code, and the debug options are only useful on sites with a source license.

Interface Support

Not all networking interface drivers have been modified to support native IPv6. It is currently supported over all Ethernet and FDDI interfaces, point-to-point interface using Cisco HDLC encapsulation, and the loopback interface. Most notably missing at this time is PPP support, which will be available in a future release. IPv6 traffic can be sent over other interfaces by setting up an IPv6 in IPv4 tunnel. For information on setting up an IPv6 in IPv4 tunnel, see the `route`(8) manual page.

Configuring an Interface for IPv6

Configuring IPv6 addresses is not currently supported with `maxim`(8). Instead, extended interface definitions must be inserted into the `/etc/netstart` file. (See the comments in the `/etc/netstart` file for up-to-date information about configuring IPv6 address.) The basic definition is:

```
ip6addr_if_name="ipv6addr/prefix"
```

This will cause the specified *ipv6addr* address to be configured on the interface. In addition, a link-local address will be automatically assigned. If only the link-local address is desired, that can be achieved via:

```
ip6addr_if_name=auto
```

Assuming that an IPv4 address has been specified for the interface, an IPv4 compatabile address can be configured by specifying:

```
v4compat_if_name=YES
```

A default IPv6 route is specified as:

```
def6route="ipv6addr"
```

The default route can be a tunnel, specifed as:

```
        def6route="-tunnel -inet ipaddr"
```

IPv6 in IPv4 automatic host-to-host tunneling can be enabled by:

```
        htunnel=YES
```

If any interfaces are configured for IPv6 addresses, the loopback interface will also be configured. If no other interfaces are configured, the loopback interface can still be configured by specifying:

```
        have_ipv6=YES
```

Adding IPv6 address to the DNS

IPv6 addresses are stored in the DNS as AAAA (pronounced "quad A") records. They are stored just as A records, but with IPv6 address. For example:

```
localhost       IN      A       127.0.0.1
                IN      AAAA    ::1
```

Reverse mappings are stored in the ip6.int domain, with the whole 128 bit address in reverse order 4 bits at a time. For example, in the `/etc/named.boot` file you might have:

```
primary     0.0.0.0.0.0.0.0.ip6.int     0.0.0.0.0.0.0.0
```

And then the file "0.0.0.0.0.0.0.0" would contain the line:

```
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0 IN PTR localhost.BSDI.COM.
```

Application support

Not all networking applications are IPv6 savvy. For those applications and servers that have been modified to support IPv6, the binaries will run with kernels built both with and without IPv6 support.

Servers that have been modified to support IPv6 include `inetd`, `tcpd`, `ftpd`, `telnetd`, `fingerd` and `tftpd`. The `/etc/inetd.conf` file has been modified to accept IPv6 connections with these servers, along with most of the internal services in inetd. Just make sure that when using IPv6, that one or more interfaces are configured with an IPv6 address (in `/etc/netstart`) before inetd is started.

Applications that support IPv6 include `ftp`, `telnet`, `finger`, `tftp`, and `ping`.

Administrative commands that have been modified for IPv6 include `traceroute`, `netstat`, `ifconfig`, `route`, `sysctl` and `tcpdump`.

New commands include `ipkey` (called key in the NRL release) and `ikmpd`.

Configuring the Domain Name System

The Easy Path to DNS Configuration

The MaxIM administration interface can perform just about every operation required to configure DNS for your system. Invoke it by:

```
% maxim
```

Then click on the DNS sections in order.

Then skip the rest of this chapter unless you want to know the detailed operation rules and file formats that DNS uses.

How DNS Works

If you use BSD/OS on a network, the system must translate host names into addresses. Host names are specified as a hierarchical name with dots separating the components of the name, for example FOO.EXAMPLE.COM. A host is considered to be a part of a domain, which is the part of its name after the first dot. Within a domain, the first component of a name can generally be used as a shorthand version of the name.

There are three mechanisms for translating these names into addresses: the Domain Name System (DNS), NIS (yellow pages), and a simple lookup table in `/etc/hosts` (see `hosts`(5)). The choice of lookup mechanism(s) is specified in the `/etc/irs.conf` file; see `irs.conf`(5).

BSDI strongly discourages the use of NIS for host lookups.

DNS is a hierarchical, distributed database for resolving hostnames to IP addresses (and vice versa). The daemon that performs this service is `named`(8). The `named` server can act as a *resolver* to do lookups, including a caching function, and it can also serve as an authoritative source of DNS information for one or more zones (or domains). A system can use a local `named` server, a remote server, or a lookup table in `/etc/hosts`. The manual page `hostname`(7) describes the usage of host names, the lookup procedures and search rules.

The `named` daemon is automatically started at system boot time if the file `/etc/named.boot` exists. This file contains configuration information for the server, and determines whether the server provides authoritative information for any domain. An authoritative server can be a primary server, loading the zone information from a file, or a secondary server that loads information from another server over the network. A server that is not a primary or secondary server for any zone acts as a local caching resolver. See the `named`(8) manual page for more information.

The file `/etc/resolv.conf` can be set up to specify the use of local and/or remote name servers; see `resolver`(5). If that file does not exist or does not specify the server location, a local server is used if present. If you are not going to configure DNS, then you will need to explicitly list every hostname that you intend to use over the network in your machine's `/etc/hosts` file. When using DNS, either by running `named` or configuring `/etc/resolv.conf`, the contents of `/etc/hosts` are totally ignored.

DNS is automatically configured during system installation using MaxIM. This chapter goes into much more detail.

For additional information on DNS and BIND configuration, we recommend *DNS and BIND*, referenced in the chapter entitled *Further Documentation*.

Local Cache

BSDI recommends configuring DNS for non-primary systems as a simple caching server (obviously the primary name server for your domain will need a more complex configuration). This will yield a local cache of commonly used IP addresses, so that if other nameservers go down, you can still resolve those hostnames to IP addresses. A simple version of `/etc/named.boot` configures a caching server:

```
directory  /etc/namedb
cache      .                    root.cache
primary    0.0.127.IN-ADDR.ARPA  localhost.rev
; forwarders 12.34.56.70
; slave
```

Edit `/etc/namedb/localhost.rev` to change the domain names to those of your domain.

If you have a local, reliable server with a richer cache (mail server, etc.), you might want to forward requests to it but still maintain your own local cache. To do so, uncomment the following lines from the above sample:

```
forwarders 12.34.56.70      Ask them...
slave                       NEVER ask anyone else
```

then replace 12.34.56.70 with the IP address of your server. In practice, you should never have more than one forwarder listed (otherwise you can generate a lot of network traffic and make name lookups slower).

The forwarding facility is useful to generate a large site-wide cache on a master server and thus reduce traffic over links to outside servers. It can also enable servers to run even though they do not have access directly to the Internet but wish to act as though they do (in which case you must use "slave").

Secondary Server

Rather than running a caching-only server, you might want to run a name server that also retains complete information for one or more local domains (a "secondary server"). Create one by adding lines like the following to `/etc/named.boot`:

```
secondary EXAMPLE.COM 12.34.56.70 secondaries/EXAMPLE.COM
```

where EXAMPLE.COM is your domain name, EXAMPLE.COM), 12.34.56.70 is the IP Address of the primary server for that domain, and `secondaries/EXAMPLE.COM` is the name of a file in the directory `/etc/namedb/` where you wish the DNS information for that domain to be cached. The standard convention used by MaxIM is to store secondary domain information in the `secondaries/` subdirectory of `/etc/namedb/` and BSDI recommends following that convention when configuring secondaries by hand. Add secondary lines for domains you frequently use as it further protects you from nameserver outages.

Add a secondary line for the relevant IN-ADDR.ARPA. domains (for `gethostbyaddr`(3)), all on one line:

```
secondary 56.34.12.IN-ADDR.ARPA
              12.34.56.70 secondaries/EXAMPLE.COM.bak
```

This example would create a secondary for the 12.34.56 network (note that the network domain address is backwards in the second field).

NOTE: Do not confuse these secondary servers created for convenience with the NIC's "official" secondary server for your site. You can create all the unofficial servers you wish.

You do not need to create or configure the `/etc/namedb/secondaries/*` files. These will automatically be downloaded from the primary nameserver and saved in the `/etc/namedb/secondaries/` directory when `named`(8) is started.

Remote Cache

Another way to configure DNS is to use the `/etc/resolv.conf` file to specify a remote nameserver to resolve hostnames for you.

Typically, you specify your fileserver, network gateway machine (a machine on which you already rely for basic system operation), or your Internet bandwidth provider's recommended name server.

Setting up DNS services using a remote server requires only the creation of a simple `/etc/resolv.conf` file (see `resolver`(5)). You need only know the IP address of your local server. For example:

```
nameserver 12.34.56.70
```

This will cause all hostname lookup requests to be forwarded to the server at 12.34.56.70 for resolution. If the hostname is not set to a full domain name for some reason, `/etc/resolv.conf` can also set the default domain name, for example:

```
domain EXAMPLE.COM
```

References to "foo" will be promoted to references to foo.EXAMPLE.COM.

Primary Name Server

Use MaxIM's many configuration options for DNS to configuration your primary name server.

Changes From Older Versions

If your system is a secondary name server, you should delete all your secondary (backup) zone files before upgrading from BSD/OS version 1.1 to any later version. Search the `/etc/namedb/` directory for the backup files (named `*.bak` or `secondaries/*` in the nomenclature used throughout this chapter).

When attempting to resolve a name, the DNS resolver in BSD/OS no longer tries every partial subdomain of your default domain as did pre-2.0 resolvers (this is for security reasons and to conform with RFC 1535). You may specify an explict list of domains to try with the search line in `/etc/resolv.conf`, see `resolver`(5) for details.

Configuring IPX/SPX Services

Overview

IPX/SPX services allow your BSD/OS system to act as a Novell file and print server, a client of other Novell file and print servers as well as an IPX to IP gateway system for Novell desktop client systems.

Connections between Novell clients and IP based services may be filtered and restricted on a number of criteria including destination address, Novell user or group, and protocol.

Branding

Before you can use any IPX services your IPX subsystem must be *branded* with a license key. To obtain a license key for your system you must have the ethernet address (MAC address) of the network card that will be used for IPX services. You can obtain this information with the ifconfig command.

```
% ifconfig ne0
ne0: flags=8963<UP,BROADCAST,NOTRAILERS,RUNNING,PROMISC,SIMPLEX,MULTICAST>
        link type ether 0:0:e8:e:46:37 mtu 1500 speed 10Mbps
        media manual
        inet 12.34.56.70 netmask 255.255.255.0 broadcast 12.34.56.255

%
```

In the case above, the string of hex information after the word *ether* is the required ethernet address for the card. You will need to make a note of the ethernet address to obtain a license key. In order to obtain your license key visit the BSDI web page:

```
http://www.bsdi.com/products/license/ipx
```

Once you have obtained the license key you may brand the system using *maxim*. Select the **Configure** choice in the **IPX/SPX** configuration section of maxim. Fill in the forms with the proper information for your network and submit them. Upon completion of the configuration forms use your browser's *back button* to return to the main maxim screen. Again in the **IPX/SPX** section of the form select **brand**. A form will be presented that will request the license key that was obtained from the license key web page. Fill in the form and submit it. Your system will then be able to use IPX services after it has been rebooted.

Configuration File Locations

Most configuration files that affect IPX services are located in `/etc/ipx.` There are also some configuration parameters for IPX/SPX services in the file `/etc/netstart`.

Configuring IPX/SPX File Services

After IPX/SPX services have been configured with maxim and the system has been rebooted, the default Netware volume (sys) will be exported. This default sys volume is located at `/var/ipx`. Other directories may also be exported by adding them to `/etc/ipx/netcexports`. The format of `/etc/ipx/netcexports` is:

```
NETCON_VOLUME <vol-name> <bsd-os-path> [read-only]
```

For example, the following file would export three Netware filesystems. The BSD/OS `/sys` directory would be available via the Netware volume named *SYS*, the BSD/OS directory `/usr` would be available as the volume named *USR* and the directory `/home/user1` would be available as the volume *USER1*.

```
NETCON_VOLUME   SYS     /sys
NETCON_VOLUME   USR     /usr
NETCON_VOLUME   USER1   /home/user1
```

For changes to be effective the system must be rebooted! Separators between fields must be tabs!

Configuring IPX/SPX Print Services

When a BSD/OS system is configured for IPX/SPX services all printers defined in `/etc/printcap` are automatically available to any Netware client. No further configuration is needed.

Configuring IPX/SPX Client Services

Mounting Remote Netware Volumes

Remote Netware volumes can be mounted automatically at boot time, or manually on the command line by the *root* user. In either case there must be a valid entry in the file `/etc/ipx/netcpasswd` for the Netware server that will be mounted. `/etc/ipx/netcpasswd` is a plain ASCII file that contains unencrypted passwords for users of remote Netware servers. For this reason it is very important that `/etc/ipx/netcpasswd` maintain an ownership of *root* and *permissions of 0600 (rw for owner only)*. Otherwise security of the system and remote Netware servers can be compromised by ordinary users. This is an example of a typical `/etc/ipx/netcpasswd` file:

```
# netcpasswd file used by /usr/sbin/netcpass daemon
# NW-srv is the name of the Novell Netware Server.
# NW-Login-name => is the Netware Login Name on the Netware Server
# NW-Passwd=> is the password on the Server (a '-' says no passwd)
# unix-username=> is the unix login name of the user
# type = either 'user', 'group' or 'world'
# The flags field contains:
#       'default' for the default machine name login
# An Entry with flags of default is required.
#
#
#[<-------UPPER CASE ONLY--------->]
# NW-srv    NW-Login-name   NW-passwd    unix-name   type    flags
#-------------------------------------------------------------------
MYSERVER    NOBODY          keep.out     nobody      user    default
MYSERVER    WHEEL           -            wheel       group   default
MYSERVER    NOGROUP         -            nogroup     world
MYSERVER    JOEUSER         joe.passwd   joeuser     user
```

For any volume mounted from the Netware server MYSERVER the default owner for all newly created files will be **nobody.** The default group for all file creations will be **wheel.** If BSD/OS user *joeuser* creates a file in the mounted Netware volume it will be owned by Netware user *joeuser.* For a user to own their own files on the remote Netware server volume, their information must be in `/etc/ipx/netcpasswd`.

If the Netware volume is to be mounted on the BSD/OS system at boot time an entry in `/etc/fstab` similar to the one below is needed:

```
#device         mount_point  type  flags  dump  fsck_pass
VOYAGER:SYS:/   /nw          netc  rw     0 0
```

Netware volumes may also be mounted by root from the command line using `mount_netc`.

```
root# mount_netc MYSERVER:SYS:/ /netware
```

The above command would mount the Netware volume *SYS* of Netware server *MYSERVER* on the BSD/OS mount point of `/netware`. For more details on `mount_netc` please see the `mount_netc`(8) man page.

Printing To A Netware Print Queue

Any BSD/OS system that is configured for IPX/SPX services may print to any Netware print queue without special configuration of the BSD/OS system using the `netcprint` command. The following command will print the file `/etc/hosts` to the Netware printer associated with print queue *BIGPRINT* on server *MYSERVER*:

```
user% netcprint -s MYSERVER -q BIGPRINT /etc/hosts
```

It is also possible to configure `/etc/printcap` to allow printing to a Netware printer using `lpr`. This makes printing to Netware printers transparent to BSD/OS users. This configuration is done by the System Administrator (root) using the `config_ipxprint`(8) utility. `config_ipxprint` will interactively prompt the system administrator for the information needed to create a new `/etc/printcap` entry and build an output filter to allow the print requests to be re-directed to the Netware print queue.

Configuring IPX to TCP/IP Gateway Services

Server Setup

No special server configuration is needed to gateway IPX network users to TCP/IP network resources. The `netchost`(8) IPX tunnel proxy starts at boot time on all systems configured for IPX/SPX services.

Setting Up Internet Access Control

The IPX to TCP/IP Gateway provides a rules based Internet access control system to ALLOW or DENY access to the Internet for Netware Users and Groups. When a workstation connects to the BSD/OS server the Netware User information is transfered to the IPX tunnel proxy server program `netchost`. The server program then obtains other User information from the Netware Bindery/NDS, such as Group membership. Every time the workstation attempts to Connect to an internet site the Access control system searches two databases. The first is the ALLOW database `/etc/ipx/netchosts.allow`, then the DENY database is searched `/etc/ipx/netchosts.deny`. The databases are searched by PROTOCOL/SERVICE (as in `/etc/services`) if the Netware User Name, Group Name, Internet Address or Host Name match an entry in the databases then access will be allowed or denied depending on which database the entry is found in.

Rules

- If no databases exist access is granted.
- ALLOW database is searched first, if a match is found access is granted.
- DENY database is searched after ALLOW, if a match is found access is denied.
- Databases are searched by PROTOCOL/SERVICE. The PROTOCOL/SERVICE name must be the same as found in `/etc/services`. The PROTOCOL/SERVICE name must match the name of the service, which is the first name on each line in the `/etc/services` file.
- PROTOCOL/SERVICE name must be the first item on each new line, followed by a ":" (i.e. "ftp:" or "http:").
- Netware Users and Group names should be UPPERCASE.
- Internet Address should be in IP address dotted format ie: "12.34.56.70".
- Host names should be lower case (i.e.: "www.example.com").

Configuration

To configure Gateway Access Control:

1. Verify that a valid entry exists in the password mapping file for the Netware server or servers that the workstation(s) are logged into.

   Example `/etc/ipx/netcpasswd` file:

   ```
   NETWARE SUPERVISOR PASSWORD root user default
   NETWARE EVERYONE      -     wheel group default
   ```

   In the example, server NETWARE user SUPERVISOR is mapped to UNIX user *root* and is the default. Netware group EVERYONE is mapped to UNIX group *wheel* and is the default. Each Netware server that workstations log into must have at least a default user and group mapped before group lookups can occur.

2. If you want to check Netware Groups the `netchost` daemon must be started with the "-g" groups option. Edit the `/etc/ipx/ipxstart` file and add the "-g" option to the `netchost` startup line. You may also kill and restart `netchost -g` from the command line as root.

   ```
   (sleep 3;/usr/sbin/netchost -g > /dev/null 2>&1) &
   ```

3. Create the database file and add the correct entries.

DENY database

`/etc/ipx/netchosts.deny`

```
#This denies all protocols/services to all users
#and groups unless permitted access by entries
#in the allow file.
ALL:ALL

#This denies Netware group EVERYONE access to ALL
#http sites.
http:EVERYONE
```

ALLOW database

`/etc/ipx/netchosts.allow`

```
#This gives Netware user TONY and group DEVELOPMENT
#access to everything.

ALL:TONY DEVELOPMENT

#This allows access to www.example.com and 12.34.56.70
#to all users.
#
http:www.example.com 12.34.56.70
```

MS-Windows & MS-Windows95 winsock setup

The setup program needed to configure the special `winsock.dll` that tunnels IP via IPX to the gateway proxy is located in the public directory of the default exported volume on the BSD/OS server. You will need to map the server's volume to install the software on your MS-Windows client.

At a DOS prompt:

```
 DOS> map j:=SERVER/sys:\
```

You can also use Windows95 Explorer to map the drive. After the drive is mapped you will need to run the setup program. The setup program for MS-Windows 3.1x is `ggate31.exe` in the public directory. In the example above that would be `j:\public\ggate31.exe`. The MS-Windows95 setup program would be `j:\public\ggate95.exe`. Select the proper setup program and execute it using **File -> Run** and follow the prompts to configure the client system for TCP/IP access via the server gateway.

The setup will also prompt you for the gateway server name. This is the name of the BSD/OS Proxy server to which you want to connect. Additionally you will be asked if you want to Auto-start Spinsock for Windows 3.1. You should answer [Y] Yes. This will add `spinsock.exe` to the Windows 3.1 *Startup.grp* startup group. Every time Windows 3.1 starts, `spinsock.exe` will connect to the gateway server. You can add or remove `spinsock.exe` from the startup program group at any time to turn the Auto-start on or off. Windows95 will Auto-start and Connect to the gateway server the first time an application loads the supplied `wsock32.dll`. The *Startup.grp* entry is not necessary on Windows95.

Any truly winsock compliant application should work with the IPX to IP gateway. It is important to make sure that Spinsock is running and connected for TCP/IP using applications to operate properly on MS-Windows client machines. The network application should be in the same directory as spinsock.exe, or spinsock.exe must at the start of the user's search path.

Security Note

When initially configured IPX/SPX services create Novell user accounts in the bindery for all users listed in `/etc/passwd` on the BSD/OS server. These bindery entries have **no password** assigned until the first time a user logs for file or print service via IPX/SPX services. This may not be the desired behavior at some sites. The system administrator (root) can set bindery passwords for all users using `netcpasswd`(8).

Configuring IP Filtering

BSD/OS can be configured to filter IP packets based on any aspect of the packet, including the full contents, the source, destination and/or return interfaces, as well as the mbuf flags. It allows you control over packets being sent from your machine, to your machine or that are being forwarded by your machine.

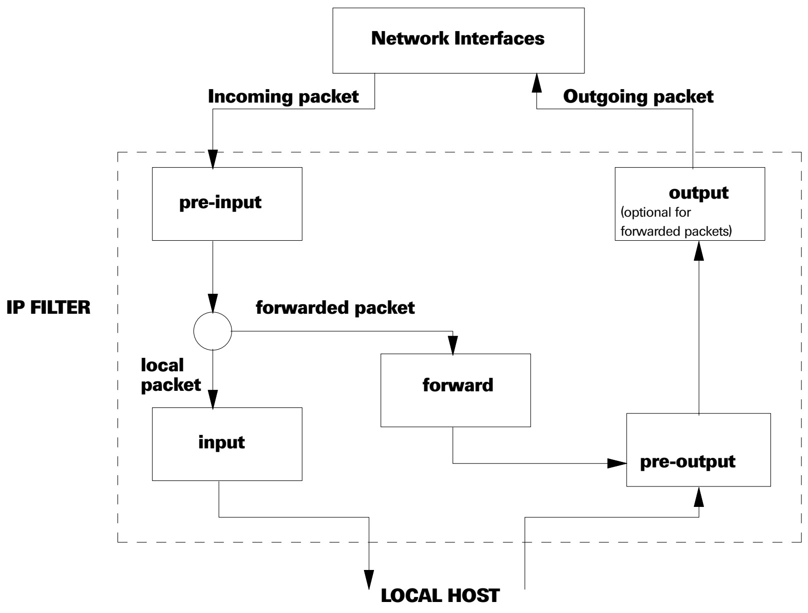
There are 5 distinct locations in the IP packet processing where filters can be applied. Any one packet will typically only be processed by a single packet filter point:

- **Pre-input**
  - *Applied to:*
    Called on every packet as it enters IP input processing,(a packet that is received from a network interface). Packets may be fragmented. Packets may be altered by the filter, but care must be taken to keep the checksum correct.
  - *Typical Purpose:*
    This filter point is typically not used. You will typically want to use the output and forward filters mentioned below. However, due to the ability to alter a packet, this filter point can be used to create a NAT (Network Address Translation) box.
- **Input**
  - *Applied to:*
    Called only on full (de-fragmented) packets destined for the local machine.
  - *Typical Purpose:*
    These are used to protect the local machine. Gateways typically should also have very strict input filters. It is not uncommon to nearly eliminate all traffic to or from the outside network from the gateway. Even packets from inside the local network destined for the gateway should be closely scrutinized.
- **Forward**
  - *Applied to:*
    Called only on packets being forwarded through the local machine. Packets may be fragmented.
  - *Typical Purpose:*
    This is probably the most traditional filter. This filter point is used by a gateway to protect machines inside the local network from attacks on the outside, or to prevent machines on the local network from contacting the outside.
- **pre-output**
  - *Applied to:*
    Called on every packet as it enters IP output processing,(a packet that is generated from the local machine on it’s way to a network interface or one that is being forwarded through the machine). Forwarded packets

may be fragmented. Packets may be altered by the filter.

- ○ *Typical Purpose:*
  This filter point is typically not used. You will typically want to use the input and forward filters mentioned below. However, due to the ability to alter a packet, this filter point can be used to create a NAT (Network Address Translation) box.

- **output**
  - ○ *Applied to:*
    Called only on packets generated locally, optionally it can be applied to packets being forwarded through this machine.
  - ○ *Typical Purpose:*
    These are used to protect the local machine. While the need for filtering incoming packets may be obvious, the need to filter outgoing packets is often missed. Output filters are needed to prevent an intruder from opening up a session back out of the machine.



At each of the filtering locations a “stack” of filter files is applied to the packet. Each filter file contains a particular set of filtering rules to be applied to the packet, see the `ipfwcmp`(8) man page for the syntax of filter files. The filter files are applied in the order they appear on the filter stack. Once the last filter file in the stack has been applied the packet is passed to the next phase of IP processing. Controlling the filter stack is done via the `ipfw`(8) command. It allows you to add, remove and change the order that filter files are applied.

To illustrate a BSD/OS IP Filter consider a site that uses a single Class C network (192.168.42.0) attached to an ISP via a T-1 connection. The ISP has assigned the IP address 10.2.62.17 as the site's gateway machine and 10.2.62.1 to its end of that connection. The ethernet port on the local machine has been assigned the address of 192.168.42.254 with a netmask of 255.255.255.0. Further, the site has a series of non-BSDI machines that are vulnerable to attack. The site has numbered all of these machines between 192.168.42.64 and 192.168.42.95 (this is the same as network 192.168.42.64 with a netmask of 255.255.255.224). Because of the O/S running on these machines, all access to the external world must be denied. The local ethernet is interface exp0 and the T-1 connection is ntwo0. To accomplish this the following forward filter could be used:

```
//
// Filter packets coming from the outside world
//
input interface(ntwo0) {
    //
    // All forwarded packets must be destined for one of our
    // addresses and not claim to originate from us.
    //
    srcaddr(192.168.42.0/24) || ! dstaddr(192.168.42.0/24) {
            reject [120];
    }
    //
    // All packets to our untrusted set of machines are dropped
    //
    dstaddr(192.168.42.64/27) {
            reject [120];
    }
    //
    // Filter out some specific services
    //
    switch ipprotocol {
    case tcp:
        //
        // First, allow already established sessions.
        //
        established {
                accept;
        }
        //
        // Now filter out certain ports
        //
        switch dstport {
        case finger/tcp:    // Disallow finger
        case nfs/tcp:       // Disallow TCP-NFS
        case exec/tcp:      // Disallow rexec
        case login/tcp:     // Disallow rlogin
        case cmd/tcp:       // Disallow rcmd
        case printer/tcp:   // Disallow printer
        case 6000-6099:     // Disallow access to X Servers
                reject [120];
                break;
        }
        break;

    case udp:
        switch dstport {
        case radius/udp:    // Disallow RADIUS server
        case radacct/udp:   // and RADIUS accounting
        case tftp/udp:      // Disallow tftp
        case nfs/udp:       // Disallow NFS
        case sys log/udp:   // Disallow syslog
                reject [120];
                break;
        }
         break;

    case icmp:
            permit;             // ICMP is always okay
            break;

    default:
            reject [120];       // We don’t know this protocol
    }
    //
    // We allow everything else
    //
    permit;
}

//
// Filter packets generated on our local network
//
input interface (exp0) {
    //
    // Make sure no-one internally is trying to spoof the
    // outside world.
    //
    !srcaddr(192.168.42.0/24) || dstaddr(192.168.42.0/24) {
            reject [120];
    }
    //
    // All packets from our untrusted set of machines are dropped
    //
    srcaddr(192.168.42.64/27) {
            reject [120];
    }
    //
    // We allow everything else
    //
    permit;
}
```

See the `ipfwcmp`(8) manual page for further details about the syntax used to write filters.

Our sample site has saved this filter under the name `/etc/rc.filters/forward.ipfw`. At the top of `/etc/netstart`, **BEFORE CONFIGURING ANY INTERFACES**, the following command is then added:

```
 ipfw forward -push /etc/rc.filters/forward.ipfw
```

This compiles the filter and pushes it on the stack of forward filters (in this case, the only filter). If the packet does not match any of the rules, it will drop through the bottom of the filter. If there were another filter below this, the packet would be run through that filter. Since this is the last filter in the chain, the packet will simply be rejected.

To see how many filters are already installed, the following command can be used:

```
 ipfw forward
```

An example output from this command might be:

```
   13: BPF filter of 48 bytes
    6: BPF filter of 2360 bytes
    5: BPF filter of 2312 bytes
    2: BPF filter of 2312 bytes
```

This shows 4 filters on the stack, the first is very small and is actually a temporary hole for a site. The other 3 are all revisions of the same filter. When you want to replace a filter you generally push the new filter on the stack and then pop the old filter off the stack, only after you are sure the new filter is working. In the above case we might want to get rid of the temporary hole:

```
   # ipfw forward -pop 13
   # ipfw forward
   6: BPF filter of 2360 bytes
   5: BPF filter of 2312 bytes
   2: BPF filter of 2312 bytes
```

and the two old filters:

```
   # ipfw forward -pop 2 5
   # ipfw forward
   6: BPF filter of 2360 bytes
```

The first number on each line is the serial number of the filter. The system automatically assigns a new and unique serial number to each filter as it is installed.

A filter may be looked at, but only the BPF assembly can be produced. To see the BPF assembly associated with a filter use the commands:

```
   # ipfw forward -output forward.filter
   # ipfwdump < forward.filter
   Serial #6
   BPF Filter:
   L0:     LD      R[0x1]
   L1:     JEQ     #0x1, L2, L42
```

```
L2:     LD      [0xc : 4]
L3:     AND     #0xffffff00
L4:     JEQ     #0xc0a82a00, L8, L5
L5:     LD      [0x10 : 4]
L6:     AND     #0xffffff00
L7:     JEQ     #0xc0a82a00, L9, L8
L8:     RET     #0x40000078
L9:     LD      [0x10 : 4]
```

Statistics associated with the filter may be displayed using the `-stats` flag to `ipfw`:

```
# ipfw forward -stats
forward filter statistics:
     38163 packets rejected
             38163 reported
  17601837 packets accepted
                 0 reported
                 0 errors while reporting
                 0 unknown disposition
```

This shows that we rejected 38163 packets. Since each of our reject lines is written as "`reject [120];`" we are reporting the first 120 bytes of each packet. This is why we see the same number of reported packets as we see rejected packets. The `ipfwlog`(8) command can be used to watch the packets that are being reported.

For further information please read the `ipfw`(8), `ipfwcmp`(8), and `ipfwlog`(8) manual pages.

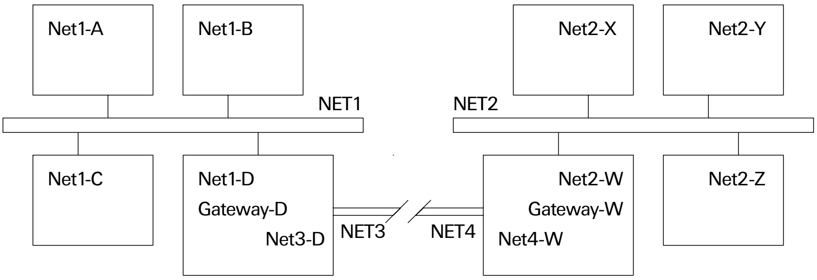
Setting up a packet filter for a site is not trivial work. You must be very careful that you are actually stopping the traffic you want to stop and not stopping the traffic you need to let go through. The above example is probably not a good base as it makes the assumption that most packets are good and we will just stop the bad ones. It is generally much better to assume all packets are bad and we make exceptions for the good ones. This example also does not include an input or output filter on the gateway, both should exist.

It is **VERY IMPORTANT** that you understand the impact of your filters on your network. You must be careful to not only drop packets that could be dangerous, but you must be careful not to drop packets that are needed. If your users perceive your filters as being too restrictive and getting in the way of getting their work done, they may try and find ways around them. This can often lead to more serious security problems.

The book "*Firewalls and Internet Security; Repelling the Wily Hacker*" by Steve Bellovin and William R. Cheswick is highly recommended.

Configuring Virtual Private Networks

BSD/OS 4.0 supports the ability to set up a Virtual Private Network (VPN). In BSD/OS, a VPN is the ability to set up an authenticated and/or encrypted tunnel between separate networks. Authentication means you can control who has access to your network, integrity means that no one can modify the packets in transit between the networks without being detected, and encryption means that the contents of the packets are encrypted when in transit between the networks. Here is a typical situation:



In this situation, the tunnel is set up between Gateway-D and Gateway-W to allow Net1 and Net2 to be treated as a VPN.

On Gateway-D, a tunnel route would be added something like this:

```
# route add Net2 Net4-W -tunnel -auth -encrypt -ifa Net1-D
```

Similarly on Gateway-W:

```
# route add Net1 Net3-D -tunnel -auth -encrypt -ifa Net2-W
```

The "-ifa" is necessary in order for the correct source address to be used for traffic that originates on the gateway machine when communicating with the other network. By default, if the source address is not specified, the address of the outgoing interface is used. Traffic from Gateway-D to Net2-X would have a source address of Net3-D. The "-ifa Net1-D" portion tells the kernel to use Net1-D as the source address when using the tunnel.

NOTE: The command should be:

```
# route add Net2 Net4-W -tunnel -auth -encrypt -ifa Net1-D
```

but "-ifa xxx" doesn't currently work with a "route add".

On Gateway-D and Gateway-W, a pre-shared key needs to be set up. The <common-key> can be up to 128 bytes long. On Gateway-D:

```
# add_preshr_key <Numeric IP address for Net4-W> <common-key>
```

On Gateway-W:

```
# add_preshr_key <Numeric IP address for Net3-D> <common-key>
```

Then, `ikmpd`(8) is run on both gateways.

In addition to setting up the appropriate tunnels, IP filters (see *IP Packet Filter*) should be set up on the gateways to protect the networks from other traffic (this is the "private" part of VPN).

In the example above, on the gateways you would want to allow AH (protocol 51) traffic between Net1 and Net2, all ICMP traffic into Net1 and Net2, and reject all other IP protocols.

In the example above, on Gateway-D you would want to allow AH (protocol 51) traffic from Net2, and all ICMP traffic. On Gateway-W, you would allow AH traffic from Net1 and all ICMP traffic..

Configuring a Web Server

Web servers provide information from your system to other sites. Web clients (also known as browsers) access that information. For your web client, we recommend the Netscape browser, found in `/usr/X11/bin/netscape`. The Netscape browser requires no configuration. This chapter discusses configuring Web servers to share data from your system with the rest of the Internet.

Configuring Apache

To configure the Apache World-Wide Web server, login as root and edit the `/var/www/conf/httpd.conf` server configuration file and customize it for your site. (It should require little modification for most sites.)

To get started, you will only need to customize the following two items:

```
ServerAdmin webmaster@www.BAR.COM
ServerName www.BAR.COM
```

where `www.BAR.COM` is the name of your server. You might wish to customize other items and the other two configuration files in the `/var/www/conf` directory later.

Next, run `/usr/sbin/config_www`, which will prompt you for your company contact information, including the organization name, mailing address, voice and fax numbers, and email contact. From that information, the configuration script will generate a very basic "home page" for your organization in the file `/var/www/docs/index.html`. You can (and should) customize that file. You will want to add other pages as your site grows and as you want to make additional information available. (The best way to get ideas about how to set up pages is to browse other Web sites and review their pages. This is particularly useful since you can download pages from other sites and use them as templates to create your own pages.)

After the `config_www`(8) utility creates the `/var/www/docs/index.html` file, it starts the Apache `httpd` process. The server will automatically be started after future reboots.

Additional configuration information about the Apache server is available online in the files found in the `/var/www/docs/apache/` directory. You can access this information from the console before you are running the server by running the text-only browser, `lynx`:

```
% /usr/contrib/bin/lynx /var/www/docs/apache/index.html
```

To kill and restart the web server run the following commands (as root):

```
# kill ‘cat /var/run/httpd.pid‘
# /var/www/bin/start-apache
```

If you have problems, check the `/var/log/httpd/error_log` file for any errors. However, if the server isn't starting properly for some reason (e.g., configuration file errors) the errors will not appear in this file, so you'll need to run Apache manually to see them:

```
# httpd -d /var/www -f /var/www/conf/httpd.conf
```

The Apache Group also maintains a web server with current information about Apache at `http://www.apache.org/`.

Other Servers

The design of your World-Wide Web server has a great deal to do with what interface is presented to the user (think of the server as the application, and the browser as the front-end). It is possible, and sometimes desirable, to run multiple web servers at your site for different purposes.

Some alternatives are the CERN server, the NCSA server, and the Plexus server (a web server written entirely in perl). There are lots of other servers available as well; each has its own set of features.

You can find information about all of the various servers and lots of other Web related things at `http://www.w3.org/` (specifically, the server information can be found at `http://www.w3.org/pub/Servers`).

If your application requires support for secure transactions, secure modules are available for the Apache server from multiple vendors. The Plexus server is written in `perl` and makes a great platform for doing flexible, custom servers.

Surf, explore, and have fun.

Announcing Your Server To The World

Before you announce your web server to the world, it is a good idea to set up a generic hostname for your server (e.g., `www.widgets.com`) that can be easily moved from machine to machine as needed with a simple change to your DNS configuration.

You should also announce your "webmaster" email alias as your site's contact point, so you can easily move it around to different people (it's no fun having to update hundreds of documents when someone changes jobs). Finally, be sure to keep a list of the places you register in case anything ever changes and you need to update them.

See `http://www.w3.org/pub/WWW/Provider/` for a list of helpful resources about registering your server.

If you have access to Usenet, you can post a notice on the `comp.infosystems.announce` newsgroup.

To register with ALIWEB (a global index service), see `http://web.nexor.co.uk/aliweb/doc/aliweb.html`.

The WWW Directory Hierarchy

The file structure in the `/var/www` tree looks like this:

Directory	Description
`/var/www/docs`	exported document tree
`/var/www/cgi-bin`	server extension scripts (e.g., bsdi-man)
`/var/www/cgi-disabled`	server extension scripts that are not enabled by default. The provided scripts are mostly examples and test scripts and should be used with care as they are not carefully screened for security problems.
`/var/www/conf`	server configuration files
`/var/www/icons`	the Apache icons, mainly used by Apache when it automatically generates an index for a directory. This directory is mapped into the web-space as `/apache-icons` .
`/var/www/bin`	miscellaneous server support binaries
`/var/log/httpd`	log files for 'hits' and transactions; can grow quite large on busy servers

Additional Information

For additional information on web server configuration, we recommend *Managing Internet Information Services*, referenced in the chapter entitled *Further Documentation*.

Configuring Virtual Hosts

It has become a common practice to offer a presence on the network via a virtual domain. A virtual domain can provide some or all of WWW, FTP, and email access without a dedicated host. This is done by configuring a BSD/OS system to respond to multiple IP addresses, giving each IP address a name in the Domain Name Service, and configuring the FTP, WWW, and email servers to recognize the additional domain name(s) and IP address(es).

There are limits to the number of IP addresses that a system can support. The GENERIC kernel shipped with BSD/OS will support about 8,000 virtual IP addresses. To configure a kernel to support more virtual IP addresses, increase the `maxusers` option in the kernel configuration file.

This chapter describes the steps necessary to register a virtual domain and configure your BSD/OS system to support it.

Configuring the Virtual Host Address

The first step is to configure the IP address to be used by the virtual host. The recommended way to do this is by adding an additional address on the loopback interface with the `ifconfig`(8) command.

For example, if `foobar.com` was assigned an IP address of `10.5.40.25`, use this command to add the alias:

```
  # ifconfig lo0 add 10.5.40.25
```

Once you have entered the `ifconfig` command you should be able to ping the virtual IP address from the local machine (but not yet from other machines).

Proxy ARP

If the IP address assigned to the virtual domain is within the range of the IP network assigned to an attached LAN interface, a proxy ARP entry will need to be installed to enable hosts and routers on the LAN to find the address.

Continuing our example, the appropriate command for `foobar.com` would be:

```
  # arp -s `/usr/libexec/linkaddr 10.5.40.25` pub
```

(Note the use of backquotes.) The `linkaddr` command will return the provided IP address of the virtual host and the Ethernet address of the appropriate Ethernet card. Once the `arp` command has installed an entry in the arp table for the virtual host, the other hosts on the local network should be able to successfully `ping` the virtual host's IP address.

In order to have these `ifconfig` and `linkaddr` commands executed automatically at boot time, add the virtual host's IP address to the file `/etc/virtualip`. The format of the file is one IP address per line. So, a file with the following entries:

```
  10.5.40.25
  10.5.40.26
```

will configure virtual hosts for IP addresses 10.5.40.25 and 10.5.40.26.

Routing

If a (sub)network has been created just for the IP addresses of virtual domains, it will be necessary to add routing entries and potentially run a routing protocol (e.g., `gated`) to announce this network. See the `gated`(8) documentation for more information about configuring `gated`.

DNS Setup

Once the virtual host can be accessed by its IP address, the next step is to assign a virtual name to the IP address using the Domain Name Service. To create a virtual domain you need to register the virtual domain name. This is done via the same procedure as registering any other domain. The best place to look for instructions to do this is `http://rs.internic.net/rs-internic.html`.

Configuring DNS

As part of registering the virtual domain, a primary nameserver is listed. If that nameserver is a BSD/OS host, use MaxIM to configure the primary DNS service for the virtual domain. See the chapter on *Configuring the Domain Name System* for more information.

Configuring E-mail

The first step to supporting a virtual host's email is to create a Mail Exchanger (i.e., MX), record for the virtual host in the DNS. This is done on the machine that is the primary DNS server for the virtual domain. The MX record instructs all other hosts on the internet to send mail destined for the virtual host to the MX host with the lowest preference value. Please refer to the section in this document on *Configuring the Domain Name System* for details about creating an MX record for a virtual host. An example of MX records for a host, called www, in the virtual domain, called virtual.com, is below:

```
 www    IN    MX    10      mail.myisp.com.
        IN    MX     5      mail.mydomain.com.
```

When a host on the internet wants to send mail to www.virtual.com, it will first try to contact the MX host with the lowest preference for the virtual host. In this case it will try to send the message to `mail.mydomain.com` because it has a preference of 5, which is lower than 10. If that host can not be contacted, the MX host with the next lowest preference will be contacted. In this case, that host would be `mail.myisp.com`. If mail can be successfully delivered to `mail.myisp.com`, `mail.myisp.com` will try and forward the mail to the MX host with the lowest preference value. If this MX host can not deliver the message immediately, it will queue the message and try again later.

On the host that has the MX record with the lowest preference value, `sendmail` needs to be instructed to dispatch the message to a real user. This is accomplished by using `sendmail`('s) *Virtual User Table* feature. The *Virtual User Table* is a `sendmail` database that maps virtual users to real user names.

For example, mail addressed to the virtual user `sales@virtual.com`, might really be intended for the the real user's email address of `john@bsdi.com`, where "john" is a real user on the real host "bsdi.com".

The first step to configuring the *Virtual User Table* is to add each virtual host that is MX'ed to the local machine to sendmail's class "T". This is done by adding the virtual host to `/etc/sendmail.cT`, for example:

```
www.virtual.com
```

The `sendmail` daemon needs to be killed and restarted whenever the class "T" file is modified. To find the process id of the currently running `sendmail` daemon, so that it can be killed, the following command can be executed:

```
# cat /var/run/sendmail.pid
132
sendmail -bd -q30m
```

The first line of the output is the process id of the currently running `sendmail` program. The second line is the command line that was used to start it. To restart the `sendmail` daemon, kill the currently running process and start a new one:

```
# kill 132
# sendmail -bd -q30m
```

The second step is to add the virtual user email mapping to the *Virtual User Table* database file. The database file is actually a "db" file. First edit the flat text file `/etc/virtusertable` by adding a line consisting of the virtual email username, and the real username it is to be mapped to, separated by a tab character:

```
sales@www.virtual.com     john@bsdi.com
```

Once the flat text file has been modified, the corresponding "db" file needs to be generated with the `makemap`(8) command:

```
# makemap hash /etc/virtusertable < /etc/virtusertable
```

Unlike the class "T" file, it is not necessary to restart the sendmail daemon after changing the *Virtual User Table* database. But, do not forget to run the `makemap`(8) command, modifying only the flat text version of the database will not be sufficient. The `makemap` command does not recognize any comments. So, the flat text version of the database file is not allowed to have any comments in it.

Once a virtual user has been added to the *Virtual User Table* database, you can verify that the virtual user is correctly mapped to the real user by running `sendmail` in test mode:

```
# sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 sales@virtual.com
rewrite: ruleset   3   input: sales @ virtual . com
```

```
rewrite: ruleset  96   input: sales < @ virtual . com >
rewrite: ruleset  96 returns: sales < @ virtual . com . >
rewrite: ruleset   3 returns: sales < @ virtual . com . >
rewrite: ruleset   0   input: sales < @ virtual . com . >
rewrite: ruleset  98   input: sales < @ virtual . com . >
rewrite: ruleset  98 returns: sales < @ virtual . com . >
rewrite: ruleset  97   input: john @ bsdi . com
rewrite: ruleset   3   input: john @ bsdi . com
rewrite: ruleset  96   input: john < @ bsdi . com >
rewrite: ruleset  96 returns: john < @ bsdi . com . >
rewrite: ruleset   3 returns: john < @ bsdi . com . >
rewrite: ruleset   0   input: john < @ bsdi . com . >
rewrite: ruleset  98   input: john < @ bsdi . com . >
rewrite: ruleset  98 returns: john < @ bsdi . com . >
rewrite: ruleset  90   input:
               < bsdi . com > john < @ bsdi . com . >
rewrite: ruleset  90   input:
               bsdi . < com > john < @ bsdi . com . >
rewrite: ruleset  90 returns: john < @ bsdi . com . >
rewrite: ruleset  90 returns: john < @ bsdi . com . >
rewrite: ruleset  95   input: < > john < @ bsdi . com . >
rewrite: ruleset  95 returns: john < @ bsdi . com . >
rewrite: ruleset   0 returns:
      $# esmtp $@ bsdi . com . $: john < @ bsdi . com . >
rewrite: ruleset  97 returns:
      $# esmtp $@ bsdi . com . $: john < @ bsdi . com . >
rewrite: ruleset   0 returns:
      $# esmtp $@ bsdi . com . $: john < @ bsdi . com . >
```

As you can see from the above output, mail sent to `sales@virtual.com` will be forwarded to `john@bsdi.com`. If email arrives for a user in `virtual.com` that is not in the *Virtual User Table* database, the message will be returned with a "user unknown" error:

```
> 3,0 nobody@virtual.com
rewrite: ruleset   3   input: nobody @ virtual . com
rewrite: ruleset  96   input: nobody < @ virtual . com >
rewrite: ruleset  96 returns: nobody < @ virtual . com . >
rewrite: ruleset   3 returns: nobody < @ virtual . com . >
rewrite: ruleset   0   input: nobody < @ virtual . com . >
rewrite: ruleset  98   input: nobody < @ virtual . com . >
rewrite: ruleset  98 returns: nobody < @ virtual . com . >
rewrite: ruleset   0 returns:
                   $# error $@ 5 . 1 . 1 $: "user unknown"
```

It is possible to create a default, or "catch-all" mapping, so that any virtual username not specifically listed in the *Virtual User Table* database, will be forwarded to an address. To create a catch-all mapping for the virtual host `www.virtual.com`, add the following line to the `/etc/virtusertable` file:

```
@virtual.com     tommy@bsdi.com
```

Now, the flat text database file should look like this:

```
sales@virtual.com     john@bsdi.com
@virtual.com          tommy@bsdi.com
```

Don't forget to run makemap(8)!

```
 # makemap hash /etc/virtusertable < /etc/virtusertable
```

Since the class "T" file was not modified, it is not necessary to restart the sendmail daemon. Now, any message sent to sales@virtual.com will be sent to john@bsdi.com; any other username in virtual.com will be sent to tommy@bsdi.com:

```
 # sendmail -bt
 ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
 Enter <ruleset> <address>
 > 3,0 nobody@virtual.com
 rewrite: ruleset   3   input: nobody @ virtual . com
 rewrite: ruleset  96   input: nobody < @ virtual . com >
 rewrite: ruleset  96 returns: nobody < @ virtual . com . >
 rewrite: ruleset   3 returns: nobody < @ virtual . com . >
 rewrite: ruleset   0   input: nobody < @ virtual . com . >
 rewrite: ruleset  98   input: nobody < @ virtual . com . >
 rewrite: ruleset  98 returns: nobody < @ virtual . com . >
 rewrite: ruleset  97   input: tommy @ bsdi . com
 rewrite: ruleset   3   input: tommy @ bsdi . com
 rewrite: ruleset  96   input: tommy < @ bsdi . com >
 rewrite: ruleset  96 returns: tommy < @ bsdi . com . >
 rewrite: ruleset   3 returns: tommy < @ bsdi . com . >
 rewrite: ruleset   0   input: tommy < @ bsdi . com . >
 rewrite: ruleset  98   input: tommy < @ bsdi . com . >
 rewrite: ruleset  98 returns: tommy < @ bsdi . com . >
 rewrite: ruleset  90   input:
               < bsdi . com > tommy < @ bsdi . com . >
 rewrite: ruleset  90   input:
               bsdi . < com > tommy < @ bsdi . com . >
 rewrite: ruleset  90 returns: tommy < @ bsdi . com . >
 rewrite: ruleset  90 returns: tommy < @ bsdi . com . >
 rewrite: ruleset  95   input:
                          < > tommy < @ bsdi . com . >
 rewrite: ruleset  95 returns: tommy < @ bsdi . com . >
 rewrite: ruleset   0 returns:
   $# esmtp $@ bsdi . com . $: tommy < @ bsdi . com . >
 rewrite: ruleset  97 returns:
   $# esmtp $@ bsdi . com . $: tommy < @ bsdi . com . >
 rewrite: ruleset   0 returns:
   $# esmtp $@ bsdi . com . $: tommy < @ bsdi . com . >
```

You can see that the nobody@virtual.com address, which previously generated an error, will now be delivered to the catch-all user for the virtual domain, tommy@bsdi.com. The sales@virtual.com email address will still be sent to john@bsdi.com. The ordering in the database file of the catch-all and user specific virtual usernames is unimportant.

A virtual user map entry is not the same as an alias. You can not map a virtual user to more then one real username. If the virtual user must be mapped to more the one real user, map the virtual user to a real user (on the local host) that is aliased to a list of real users. Consider creating a virtual user map entry:

```
  info@virtual.com    virtualinfolist
```

and then creating an alias for "virtualinfolist" in the aliases database file /etc/aliases:

```
  virtualinfolist: ruser1@bsdi.com, ruser2@other.domain.com, ruser3
```

This will send mail destined for info@virtual.com to the three users, ruser1@bsdi.com, ruser2@other.domain.com and ruser3.

Another restriction of the virtual user map is that the virtual user cannot be mapped to another virtual user. It must be mapped to a user on a host that is not in the class "T". If the virtual user is mapped to a host that is in class "T", mail to that user will generate a "user unknown" error.

Configuring Sendmail to forward via UUCP

It might be necessary to provide mail services to hosts via UUCP and have the UUCP host's name appear to be on the Internet. This can be done by creating an MX record for the uucp host to the local machine. The sendmail program can then be configured to forward mail destined for the DNS name of the UUCP host by using the mailer table database. The mailer table database identifies hostnames and selects specific mailers to be used for delivery to that host.

In this example, a machine has a DNS name of uucp.domain.com and a UUCP hostname of uucust. The primary DNS server for the domain domain.com should supply an MX record for uucp.domain.com to the local host. A mapping that assigns an appropriate UUCP mailer and UUCP hostname to the UUCP DNS name will instruct sendmail to forward the email, via UUCP to the UUCP host.

The mailer table file is /etc/mailertable. To create a mapping add the following line to the flat text version of the mailer table database file /etc/mailertable:

```
  uucp.domain.com    uucp-dom:uucust
```

Then build the "db" version of the mailertable database file by using the makemap command:

```
  # makemap hash /etc/mailertable < /etc/mailertable
```

It is not necessary to restart the sendmail daemon after modifying the mailer table database file, but don't forget to run the makemap(8) command. Also, note that the makemap(8) command does not recognize comment lines, so you can not add comments to the /etc/mailertable file.

To test that the new mapping has been recognized, run sendmail in test mode:

```
  # sendmail -bt
  ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
  Enter <ruleset> <address>
  > 3,0 user@uucp.domain.com
  rewrite: ruleset   3   input: user @ uucp . domain . com
```

```
rewrite: ruleset  96   input: user < @ uucp . domain . com >
rewrite: ruleset  96 returns: user < @ uucp . domain . com . >
rewrite: ruleset   3 returns: user < @ uucp . domain . com . >
rewrite: ruleset   0   input: user < @ uucp . domain . com . >
rewrite: ruleset  98   input: user < @ uucp . domain . com . >
rewrite: ruleset  98 returns: user < @ uucp . domain . com . >
rewrite: ruleset  95   input:
< uucp-dom : uucust > user < @ uucp . domain . com . >
rewrite: ruleset  95 returns:
$# uucp-dom $@ uucust $: user < @ uucp . domain . com . >
rewrite: ruleset   0 returns:
$# uucp-dom $@ uucust $: user < @ uucp . domain . com . >
```

From the above you can see that a message destined for `user@uucp.domain.com` will be forwarded using the `uucp-dom` mailer to the UUCP host named `uucust`. The `uucp-dom` mailer uses the UUCP delivery mechanisms, but preserves the domain name style addressing. This is usually the correct UUCP mailer to use, but other UUCP mailers provided by the `sendmail.cf` file can be used instead, as appropriate.

Configuring Web Servers

BSD/OS 4.0 is shipped with the `apache` WWW server, which includes virtual domain support. There are other servers capable of serving virtual domains, the details of which are beyond the scope of this document.

Everything you need to configure virtual domains for Apache is found in its `conf` directory, normally found in `/var/www`. The items to check and/or modify are in the `httpd.conf` file.

At the end of the distributed `httpd.conf` file there is a sample configuration section for virtual web servers bracketed by `<VirtualHost>` and `</VirtualHost>`. Most `httpd.conf` and `srm.conf` directives can go within a `VirtualHost` section to tailor the behavior of the server. Here is an example:

```
<VirtualHost www.foobar.com>
   ServerAdmin webmaster@www.foobar.com
   DocumentRoot /var/www/httpd/htdocs/foobar
   ServerName www.foobar.com
   ErrorLog /var/www/logs/foobar/error_log
   TransferLog /var/www/logs/foobar/access_log
</VirtualHost>
```

You might wish to use the `Redirect` directive if you are moving pages to be under their own virtual web server. The `Redirect` directive enables you to rename old links so that they continue to work after you rearrange your server.

Most of the directives in a `VirtualHost` section can also be global, but it rarely makes sense to do so. One directive that should not be in a `VirtualHost` section is `BindAddress`. You should either not have any `BindAddress` directive at all, or you should have a global `BindAddress` line that looks like this:

```
 BindAddress *
```

This tells the server what IP address to listen on. For virtual hosting to work it must be "*" (the default).

For more information on apache, see the httpd(1) manual page.

Configuring the FTP Server

BSD/OS now ships with a new ftp server. In the past we have shipped an older version of wu-ftpd with local modifications. Due to the number of security advisories against wu-ftpd in the past, BSDI has moved to a new code base that we intend to support internally. While the new server does not have exactly the same feature set as wu-ftpd, it does have the most desirable facilities. The new server is not intended to be either a drop in replacement or contain every last feature of wu-ftpd.

While BSDI does not recommend its use, the latest wu-ftpd, 2.4.2-beta-16, is included with BSD/OS as /usr/contrib/libexec/wu-ftpd. To use wu-ftpd rather than the new ftp server you must change your inetd.conf file. Depending on your usage of tcp wrappers, you have a line in your inetd.conf file that looks something like one of the following two lines:

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

You should change this line to either (shown with lines folded):

```
ftp stream tcp nowait root /usr/libexec/tcpd
     /usr/contrib/libexec/wu-ftpd -l -A
ftp stream tcp nowait root /usr/contrib/libexec/wu-ftpd
     wu-ftpd -l -A
```

Note that this version of wu-ftpd does not support virtual hosts via the ***-F*** and ***-m*** options. Please review ftpaccess(5) and read about the "virtual" configuration option. This version should be better than the version shipped with earlier BSD/OS releases and is maintained by wu-ftpd-bugs@academ.com. (The support for login.conf(5) is a local BSDI addition. Future releases of BSD/OS will only include this modification if it is accepted back by the maintainer.) There may be other differences that BSDI is not aware of.

BSDI does recommend the use of the default ftp server, ftpd. If this is a new installation you will not need to make any changes to the /etc/inetd.conf file. If this is an upgrade you many need to change the ftpd line of the /etc/inetd.conf file to not include any options to ftpd (all options can be specified in the /etc/ftpd/config file.)

For a full description of the /etc/ftpd/config file please review the ftpconfig(5) manual page.

There are 3 major features of ftpd that deserve discussion here. Incoming directories, virtual hosting, and session limits.

The config file contains variables, flags, and directives. These may be limited to a specific virtual host, only guest accounts, or guest accounts for a specific virtual host. To limit a block of the config file to guest accounts, you must start that block with the line:

```
<Guest>
```

and end the block with

```
</Guest>
```

These must be on their own lines and not included with other text. The ftpd server has a built-in version of the `ls(1)` command. This means you no longer need a bin directory or a shlib directory in your guest directory. You still need `/etc/pwd.db` and `/etc/group` files if you want `ls` to list symbolic user/group names rather than numbers. (This is not required.)

To start a virtual host block (and define that virtual host), you must start the block with

```
<VirtualHost hostname>
```

and end the block with

```
</VirtualHost>
```

Incoming directories are restricted for guest (anonymous) sessions. Only directories (and their descendents) explicitly listed in the config file may have files placed in them. The incoming directive is only valid within a guest session. For example, suppose you want to have a directory called "incoming" at the top level of your guest ftp directory. You wish to allow guests to deposit files here (but not read or delete them) and you want the resulting files to be owned by the account "support" and the group "archive". Within your guest block you should have the line:

```
Incoming /incoming support archive 660
```

This will allow guests to deposit files in the `/incoming` directory (this is relative to the directory used for guest sessions.) Files deposited there will be owned by *support*, group *archive* and have the mode of *660* (read/write by user and group). Guest users will not be able to read or delete these files. Of course, you must first create the incoming directory. To do this you should first change directories to the directory used for guest sessions. Then:

```
# mkdir incoming
# chown support.archive incoming
# chmod ug=rwx,o=wx incoming
```

You should note that any subdirectory (up to 8 levels deep) from `/incoming`, that is writable by a guest ftp account, will also be marked as an incoming directory. To allow guests to create their own directories you can change the Incoming line to read:

```
Incoming /incoming support archive 660 773
```

The last argument both allows subdirectory creation (you must make sure the MKD command is turned on!) and specifies the mode of newly created directories. Just like files, they will be owned by *support* and have group *archive*. Mode 773 is the same as ug=rwx,o=wx.

You may want to create a ".message" file in the Incoming directory. This file will be displayed each time a user enters this directory.

Virtual hosts, as indicated above, are started with the line:

```
<VirtualHost hostname>
```

and end with the line:

```
</VirtualHost>
```

Everything between these lines applies only to that virtual host. The hostname specified should resolve to a single IPv4 or IPv6 address. (If it resolves to more than a single address only the first address found will be used).

A typical virtual host block would look like:

```
<VirtualHost ftp.mycompany.com>
    AnonymousDir  /var/spool/ftp.mycompany.com
    StatFile      /var/log/ftpd/stats-ftp.mycompany.com
    <Guest>
        MKD       On
        Incoming  /incoming root wheel 600
        Incoming  /support support archive 660 773
    </Guest>
</VirtualHost>
```

The ftp server also has the ability to manage the number of guest sessions allowed. To use this facility, ftpd must not be started from `inetd.conf`. You must comment out the ftpd line in the `/etc/inetd.conf` and start ftpd from `/etc/rc.local` with the command:

```
/usr/libexec/ftpd -D -p /var/run/ftpd.pid
```

You can cause ftpd to re-read the configuration file by sending it a SIGHUP:

```
# kill -HUP ‘cat /var/run/ftpd.pid‘
```

To specify the maximum number of guest sessions that are allowed set the **MaxUsers** within a guest block:

```
<Guest>
    MaxUsers    50
</Guest>
```

This will limit the number of guests (per virtual host) to 50. Both guest and non-guest sessions are counted, but only guest sessions are limited. If you had 50 non-guest users currently logged in, then guest users would not be able to log in. As indicated, this may be specified in a virtual host block. Each virtual host keeps track of its own number of sessions.

Configuring Virtual Hosts for telnet

Telnetd is a easy to enable for virtual hosting. Just add the `-m` flag to its invocation in `/etc/inetd.conf`.

Security Considerations for Virtual Hosts

If you are running a packet filter or have customized your TCP wrapper configuration, you should ensure that you have taken your virtual machines into account.

Consider the implications of someone noticing that you are hosting “virtual” domains. There are a number of avenues through which this can be discovered:

- The `telnet` command does not support virtual hosts and its default login banner contains the official machine name. You might wish to change this.
- The `sendmail` command will also reveal the official name of the machine on which it is running. This information will appear in the headers of mail; it will also be available to anyone who `telnets` to the smtp port on the virtual machine.
- The `apache` command will let you specify the machine name that it returns. It is a good idea to set this even if security is not a major concern, as it leads to less confusion.
- The standard `ftpd`(8) will also reveal the official name of the machine it is running on.

Configuring an HTTP Proxy (Squid)

How Squid Works

Squid is a caching HTTP proxy and accelerator that can also service virtual hosts. When run as a proxy, Squid can significantly reduce Web related traffic on your Internet link, while simultaneously increasing the speed with which content is delivered to your local users. In accelerator mode, Squid runs "in front of" your HTTP server and caches frequently requested pages. This reduces the load on your HTTP server, visitors to your pages might also see improved responsiveness.

Multiple Squid caches can communicate among themselves to deliver content from the "nearest" (in terms of response time) cache, using a protocol called ICP (Internet Cache Protocol). This facility could be used by an ISP, for example, by configuring a large cache at the Internet connection and smaller "child" caches at remote POPs (points of presence). The result would be that the large "parent" cache would hold copies of all of the cachable Web pages requested by the ISP's clients. The "child" caches located at the POPs would hold copies of the pages requested by the clients local to that POP. Such a configuration can save considerable bandwidth.

Consider this (somewhat simplified) example of what would happen with a popular page, e.g., the Netscape home page. The first browser to request the page would cause the page to be downloaded from Netscape's server. A copy would be kept in both the parent cache and in the child cache at the POP from which the request came. If the next request for the page came from a browser at a different POP the request would be satisfied from the parent cache and a copy would be stored in the child cache at the second POP. Requests for the page from browsers connected to either of the two POPs where the page was held in the cache would be satisfied directly from the child cache at the POP.

A well tuned Squid cache might see hit rates in the 40 to 60 percent range (assuming a typical user population), significantly cutting the amount of HTTP and other traffic on your Internet link.

Getting Started

Squid is started from the `/etc/rc` file, which runs the `start-squid` script that resides in the directory `/var/www/squid/bin` if the file `squid.conf` exists in the Web configuration directory `/var/www/conf`. The `start-squid` script can be modified to tailor Squid's environment to suit your needs.

Squid configuration files are stored in the `/var/www/conf` directory. The file `squid.conf.default` contains the complete set of configuration options and information on their default values. The files `squid-accelerator.sample` and `squid-proxy.sample` are sample configuration files for simple accelerator and proxy configurations.

Squid log files are stored in the `/var/log/squid` directory. The log files are managed by the `squid.daily` and `squid.weekly` scripts, which reside in the `/var/www/squid/bin` directory. These scripts are called from the system `/etc/daily` and `/etc/weekly` scripts.

Proxy Mode

A simple proxy-only configuration is available in the file `squid-proxy.sample`. To get started with Squid as a proxy, all that is necessary is to copy this configuration file to `squid.conf`, and start Squid. The following commands will do this:

```
# cd /var/www/conf
# cp squid-proxy.sample squid.conf
# chown www.www squid.conf
# chmod 600 squid.conf
# /var/www/squid/bin/start-squid
```

The reason for the `chmod` command is because the Squid configuration file can contain plain text passwords for the `cachemgr.cgi` interface.

Once you've started Squid, you will find lines describing Squid's actions as it started in the `/var/log/squid/cache.log` file, ending with lines that look like:

```
Accepting HTTP connections on FD 18.
Accepting ICP connections on FD 19.
Ready to serve requests.
```

If this is not what you see, review the messages to determine the cause of the problem. Once the `/var/www/conf/squid.conf` file exists, Squid will be started automatically when the system boots.

All that remains to be done is to configure your clients to use Squid as their proxy (see the section below entitled *Client Configuration* for additional information).

Accelerator Mode

Squid can act as a 'front-end' that caches web pages for extremely fast response, in a configuration called 'accelerator'. A simple accelerator-only configuration is available in the file `squid-accelerator.sample`. This configuration assumes that you are providing HTTP service on port 80 (the standard HTTP port) and that your HTTP server is on the local machine and will be moved to port 81.

To get started with Squid as an accelerator, begin by copying the `squid-accelerator.sample` configuration file to `squid.conf`:

```
# cd /var/www/conf
# cp squid-accelerator.sample squid.conf
# chown www.www squid.conf
# chmod 600 squid.conf
```

The reason for the `chmod` command is because the Squid configuration file can contain plain text passwords for the cachemgr.cgi interface.

If your HTTP server is not running on the same machine as Squid, or if you wish to use a different port number, you will need to edit the configuration file and change the `httpd_accel` line to specify the machine and/or port where your HTTP server is running. If you wish to run in virtual host mode you will also need to edit the configuration file (see below).

Next, configure your HTTP server to run on a port 81 (or another port that you choose). For the Apache server, this is done by editing the `/var/www/conf/httpd.conf` file and changing the Port line:

```
Port 81
```

After restarting your HTTP server, start Squid:

```
# kill -1 ‘cat /var/run/httpd.pid‘
# /var/www/squid/bin/start-squid
```

Once you've started Squid, you will find lines describing Squid's actions as it started in the `/var/log/squid/cache.log` file, ending with lines that look like:

```
Accepting HTTP connections on FD 18.
Accepting ICP connections on FD 19.
Ready to serve requests.
```

If this is not what you see, review the messages to determine the cause of the problem. Once the `/var/www/conf/squid.conf` exists, Squid will be started automatically when the system boots.

Next, use your Web browser to ensure that your HTTP server is accessible. No special browser configuration necessary when running Squid in accelerator mode.

Accelerator Mode with Virtual HTTP Server Support

If all of your virtual HTTP servers run on the same machine as Squid and are all on the same port, the configuration changes for virtual host support are quite simple: just edit the `httpd_accel` line in the Squid configuration file to read:

```
httpd_accel virtual your.real.httpd.port
```

Remember that Squid logs requests to a single log file. When processing your log files you might wish to separate out the accesses by virtual host.

If you have a more complex configuration you might find the Squid `redirect_program` facility useful. This facility allows you to use an external process to rewrite requested URLs. You can find more information on this in the Squid FAQ.

Special Considerations When Running an Accelerator

When Squid is running in accelerator mode, it is handling requests that would otherwise go to your HTTP server. This means that in order to get an accurate picture of the traffic at your Web site you must merge the logs kept by Squid with your HTTP server's logs.

Squid has a flexible mechanism for determining which pages are cached and which are not. If your Web site uses security features or generates pages dynamically you will want to become familiar with these controls and configure Squid to not cache these pages (see the section below entitled *Other Sources of Squid Information* for additional information).

Some HTTP servers will rewrite URLs they return if they are running on a non-standard port. This will cause clients accessing these URLs to bypass the accelerator. To avoid this problem (without having to modify the HTTP server) you can run the server on a different machine (or at a different IP address) on port 80. For virtual servers it will be necessary to use the `redirect_program` facility if you run into this problem and are running the HTTP server on the same machine as Squid.

Client Configuration

In order for HTTP browsers to take advantage of Squid, they must be configured to direct their requests through Squid instead of directly to the HTTP server. The method used to do this varies according to the browser in question. Instructions for some common browsers are included here.

If you wish to enforce the use of Squid as a proxy you can configure your firewall to filter HTTP traffic that does not originate from your proxy.

Netscape Navigator

The Netscape Navigator offers two methods of proxy configuration: manual and automatic.

For Navigator 4.x manual configuration is performed by selecting the `Edit/Preferences` menu and clicking on the "expand arrow" next to the `Advanced` menu item and finally selecting the `Proxies` selection. Switching to the right side of the window, select the `Manual proxy configuration`. Now click on the `View` button and fill in the Manual Proxy Configuration form. Squid can proxy FTP, Gopher, HTTP and Security (HTTPS) requests for Web browsers. For each protocol that you wish to proxy fill in the fully qualified hostname of the machine on which Squid is running and the port number on which Squid listens for HTTP connections (by default, port 3128).

For Navigator 3.x manual configuration is performed by selecting the `Edit/Preferences/Advanced/Proxies` menu and clicking on the `Manual Proxy Configuration` button. Now click on the `View` button and fill in the Manual Proxy Configuration form. Squid can proxy FTP, Gopher, HTTP and Security (HTTPS) requests for Web browsers. For each protocol that you wish to proxy fill in the fully qualified hostname of the machine on which Squid is running and the port number on which Squid listens for HTTP connections (by default, port 3128).

For Navigator 3.x manual configuration is performed by selecting the `Options/Network Preferences/Proxies` menu and clicking on the `Manual Proxy Configuration` button. Now click on the `View` button and fill in the Manual Proxy Configuration form. Squid can proxy FTP, Gopher, HTTP and Security (HTTPS) requests for Web browsers. For each protocol that you wish to proxy fill in the fully qualified hostname of the machine on which Squid is running and the port number on which Squid listens for HTTP connections (by default, port 3128).

There are two steps to setting up automatic proxy configuration. A JavaScript configuration program must be written and stored on an HTTP server in a location accessible to the browsers that you wish to have use automatic proxy configuration. Netscape provides documentation on writing these files at:

```
http://home.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html
```

Once you have installed the configuration program, the client end of automatic proxy configuration is done by selecting the Options/Network Preferences/Proxies menu and clicking on the Automatic Proxy Configuration Button. Next, fill in the URL of your proxy configuration file in the Location box. That is all there is to it.

Internet Explorer

Both automatic and manual configuration methods are supported.

To manually configure Internet Explorer 4.0 to use Squid select the `View/Internet Options` menu option. >From there go to the `Connection` tab and in the proxy server box check "Access the internet using a proxy server" and fill in the requested information. Select the `Advanced` button if you require finer control.

To use automatic configuration it will be necessary to refer to the Internet Explorer documentation (the URL given is correct as of this writing).

```
http://www.microsoft.com/ie/corp/deployguide.htm
```

Mosaic and Lynx

Proxy configuration for the BSD/OS (and other Unix-like operating system) versions of Mosaic and Lynx is done by setting environment variables. For users of the Borne shell and its derivatives (sh, bash, ksh and zsh):

```
$ export http_proxy="http://your.squid.host:3128/"
$ export gopher_proxy="http://your.squid.host:3128/"
$ export ftp_proxy="http://your.squid.host:3128/"
```

For users of the C shell and its derivatives (csh and tcsh):

```
% setenv http_proxy "http://your.squid.host:3128/"
% setenv gopher_proxy "http://your.squid.host:3128/"
% setenv ftp_proxy "http://your.squid.host:3128/"
```

Typically these commands would be added to the user shell start up files.

Other Sources of Squid Information

Start your search for additional information on Squid by going to:

```
http://squid.nlanr.net/
```

There you will find a Squid FAQ, information on subscribing to the "squid-users" mailing list, the latest versions of the Squid source code and other useful information.

Monitoring Squid's Operation

Squid provides an interface for monitoring the operation of your cache. The monitoring program is named `cachemgr.cgi` and is located in the `/var/www/squid` directory. Communication with the Squid proxy is through a private protocol called "cache_object".

The access controls for the Squid cachemgr interface are in two parts. First, the HTTP server must allow you to run `cachemgr.cgi`. Second, Squid must be configured to accept queries from the machine where `cachemgr.cgi` is running.

By default, the Apache HTTP server is configured to allow access only to the `/var/www/squid` directory to browsers running on the local machine. To enable access for remote browsers, review the section between the `<Location/squid-status>` and `</Location>` tags in `/var/www/conf/access.conf` and change it as necessary for your needs. A common configuration would be to allow access from your domain, like so:

```
<Location /squid-status>
<Limit GET>
order deny,allow
deny from all
allow from .<your domain name here>
</Limit>
</Location>
```

Once you have configured your HTTP server to run the `cachemgr.cgi` script, you must review the Squid configuration. By default, Squid will only accept connections with the `cache_object` protocol from the local machine. Here is the relevant section of the default rules from the sample Squid configuration files:

```
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255

# Only allow access to the cache manager
# functions from the local host.
http_access deny manager !localhost
```

More liberal configurations might allow access from the local network:

```
acl manager proto cache_object
acl localnet src <local network number>/<netmask>

# Only allow access to the cache manager
# functions from the local net.
http_access deny manager !localnet
```

Or from your any system in your domain:

```
acl manager proto cache_object
acl bsdi domain .bsdi.com

# Only allow access to the cache manager
# functions from our domain.
http_access deny manager !bsdi
```

You may now access the `cachemgr` interface with the URL:

```
http://your web server/squid-status
```

Some of the management functions (such as shutting down the proxy or viewing the configuration file) are password protected. To enable these functions, you will need to add `cachemgr_passwd` lines to `squid.conf`. For example:

```
cachemgr_passwd secret shutdown squid.conf
```

See the `squid.conf.default` file in the `/var/www/conf` directory for more examples. Note that password is logged in the `access.log` file, which should not have other read permissions. Since the passwords in `squid.conf` are in plain text, access to this file should also be restricted. In addition the password is passed over the network as plain text.

Log file management

Squid uses up to four log files, all of which are stored in the `/var/log/squid` directory.

The log files are:

- `access.log`: Logs client request activity. Contains an entry for every HTTP and ICP request received. If the `log_mime_hdrs` configuration option is enabled the HTTP headers from the browser's request and from the HTTP server's reply are also logged. Note: unless the browser is making HTTP 1.0 or later requests the server will not return any headers.
- `cache.log`: Cache logging file. The "debug_options" line in the Squid configuration file might be used to fine tune the logging level.
- `store.log`: (optional) Logs the activities of the storage manager. Shows which objects are ejected from the cache, which objects are saved and for how long.
- `useragent.log`: (optional) Logs the `User-Agent` headers sent by browsers connecting to your proxy. Note that this information is also logged in the `access.log` if you enable logging of HTTP headers with the `log_mime_hdrs` configuration option.

To turn off the optional log files, enter the word `none` as the name of the log file. For example:

```
User-Agent none
```

Squid log files can grow to be quite large. A busy proxy, requesting full logging can easily accumulate 200 MB or more of log files per day. You will want to plan for this when allocating file system space (remember that when compressing log files you will temporarily need space for both the compressed and uncompressed version of the file). A busy proxy will also benefit from having the log files and the on-disk cache (kept in `/var/www/squid/cache` by default) on separate disk drives.

As distributed, Squid clean-up tasks, including cleaning up Squid log files, are handled by two shell scripts: `squid.daily` and `squid.weekly` in the `/var/www/squid/bin` directory. These scripts are called from the system `/etc/daily` and `/etc/weekly` scripts. The distributed `squid.daily` script does nothing. The distributed `squid.weekly` script generates a summary of the weeks usage that is mailed to the `cache_manager` and rotates the log files in a four week cycle. For a small proxy, this is probably adequate, although it means that after four weeks, old log data is discarded.

If you are running in accelerator mode, it is important to remember that your HTTP server will not be logging requests that are satisfied out of Squid's cache. To get an accurate picture of the traffic seen by your Web site it is necessary to merge the data from the Squid and HTTP server logs.

Squid has its own log format and is also capable of emulating the Common Log Format used by many HTTP servers (and parsed by many packages for analyzing HTTP server logs). The Squid log format has the advantage of containing more information, some of which is used by the Squid log analysis tools (see the section entitled *Other Sources of Squid Information* for additional information. The Perl scripts to analyze Squid log files reside in the `/var/www/squid/bin` directory. Additional log file manipulation scripts, including a script to convert from Squid to the Common Log Format are available at:

```
http://squid.nlanr.net/Squid/Scripts/
```

Tuning the Cache

Squid offers considerable flexibility in tuning the behavior of the cache. The default configuration is conservative about caching, preferring to err on the side of unnecessarily refreshing an object.

After you have been running Squid for a while, you might wish to tune your configuration to increase caching efficiency.

Additional Tips

If you are running multiple Squid processes on a single system (for example a proxy and accelerator), make sure that each Squid is configured with its own cache directory (using the `cache_dir` configuration option). Also make sure that each Squid uses a separate file for recording its PID (using the `pid_filename` configuration option).

For particularly busy caches, you should use multiple cache directories, each on a different disk drive.

Solving Problems

The most common Squid problems involve resource exhaustion on busy caches. Squid achieves its high performance by making extensive use of in-memory caching. As a result, the memory usage of a busy cache can get **very** large. As a starting point, you can estimate that Squid will consume approximately three times the amount of memory specified by the `cache_mem` configuration directive.

Start your investigation of any problem by reviewing your log files and attempting to define the problem clearly. If Squid is running but appears not to be doing what you'd expect, you might find it helpful to use a browser to compare the results of a direct request to a request made through Squid. If the problem seems to involve an interaction with a particular site or server, you might find `telnet(1)` to be a useful tool for investigating the headers sent to Squid from the remote server. You might also find it useful to enable logging of HTTP headers (by enabling `log_mime_hdrs` in `squid.conf`).

Don't forget to consult the Squid FAQ and the archive of the squid-users mailing list. Both are available at:

```
http://squid.nlanr.net/
```

A clear statement of the problem sent to the `squid-users` mailing list (<squid-users-request@nlanr.net>) might also elicit help.

Frequently Requested Pages Are Not Being Cached

Usually, this is either a problem with the tuning of the cache (perhaps the refresh policies, `refresh_pattern`, maximum cached object size limits, or `maximum_object_size` in `squid.conf` are too conservative), the object is changing frequently, or the Webmaster at the remote site is taking steps to prevent the page from being cached.

If the Webmaster is deliberately taking steps to keep the page from being cached (for example setting the 'Expires' header to the same time as the request for the page), you'll need to determine if the page content is actually changing that often. If so, this behavior is appropriate. If not, it might be worth your trouble to point this out to the Webmaster, but some sites do this intentionally, to prevent even relatively static pages from being cached (probably to enable them to keep a more accurate count of the number of hits that a particular page receives).

Here is a telnet session investigating why Microsoft's home page is not being cached, even though the last modified time on the page was over 36 hours ago:

```
$ telnet www.microsoft.com 80
Trying 207.68.137.9...
Connected to www.microsoft.com.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.0 200 OK
Server: Microsoft-IIS/2.0
Date: Mon, 28 Oct 1996 17:52:04 GMT
cache-control: private
Content-type: text/html
Expires: Mon, 28 Oct 1996 17:52:04 GMT
Set-Cookie: DENALISESSIONID=KRSCBBXQADJELYYW; path=/
Cache-control: private

Connection closed by foreign host.
```

Microsoft clearly does not want this page to be cached, as the HTML headers show. They are using the HTTP 1.1 cache-control header to specify that this page is private. In addition, the expires header is set to the current time, which will also cause a proxy not to cache the page.

You can use the `refresh_pattern` configuration directive to set a minimum amount of time that a page will be considered fresh. This can be set on a site by site (or even page by page) basis. For example, adding the following line to the `refresh_pattern` section of your `squid.conf` file would cause Microsoft's home page to be retained for six hours (360 minutes) before loading a new version:

```
refresh_pattern ^http://www.microsoft.com/$ 360 0% 0
```

Since Microsoft explicitly expires that page as soon as you get it, the last two fields in the refresh pattern (the last modified factor and the maximum life of the page) have no influence.

FATAL: xmalloc: Unable to allocate NNNN bytes!

This is almost always an indication that Squid has run up against the process maximum data size limit. Either increase the limit (in the `/var/www/squid/bin/start-squid` file) or configure Squid to use less memory (by changing `cache_mem` in the `/var/www/conf/squid.conf` file).

Access is denied to squid-status pages

As distributed, access is only allowed from the local host. You must modify the access rules in order to view the pages from any other machine. See the section entitled *Monitoring Squid Performance* for additional information.

FTP requests are not being proxied

The most frequent cause of this problem are due to attempts to use Squid as a general purpose FTP proxy. Squid can only proxy requests received via HTTP.

Configuring an NIS Client

Network Information Service (NIS)

If your site uses the Network Information Service (NIS) for hosts, password and group files and other management information, you can configure your BSD/OS 4.0 system to act as an NIS client. (Note, BSD/OS will not act as an NIS server, only as a client.)

Enabling NIS on your BSD/OS system requires three changes:

1 Set the NIS domain name in the `/etc/netstart` file. Edit the `/etc/netstart` file and append your NIS domain to the `nis_domain=` line. For example, if your NIS domain name is `foo.com`, the line should read:

```
nis_domain=foo.com
```

This change will take effect when the system is rebooted.

2 Review the `/etc/irs.conf` file. Each uncommented line in this file has either two or three fields. The first field is the name of the information that is being managed, and the second is the access method used to retrieve that information. The third field specifies what happens when information can be obtained from more than a single source.

You should uncomment the lines of this file that have "nis" as their access method. You should also verify that the file correctly reflects the hierarchy of information for your domain. The default hierarchy is likely to be correct, but you should make sure.

3 Uncomment the line in your system's `/etc/rc.local` file that causes the `ypbind`(8) daemon to be started each time the system boots.

You should also be aware that NIS passes authentication information – including weakly-encrypted passwords – across networks in an insecure manner, and is therefore inherently more vulnerable to attacks than other authentication methods such as Kerberos.

BSD/OS does not use the + or - at the beginning of a password file line to indicate special handling of login names or netgroups, or to indicate that NIS should be consulted. The `/etc/irs.conf` file is the only means to turn on or off use of NIS for any type of map.

For further BSD/OS-specific information, see the `irs.conf`(5) and `ypbind`(8) manual pages.

For additional information on NIS configuration, we recommend *Managing NFS and NIS*, referenced in the chapter entitled *Further Documentation*.

Configuring Sendmail

The Sendmail configuration file (`/etc/sendmail.cf`) distributed with BSD/OS 4.0 requires no additional configuration for normal use at sites with common configurations. Read this chapter if you have extraordinary requirements.

The Sendmail V8 shipped with BSD/OS has a flexible configuration scheme that is better suited for managing sites with complex `sendmail` configurations than were historic versions of sendmail. It enables you to share site-wide `sendmail` configurations while still enabling local administrative domains to customize other parts (e.g., local UUCP connections or mail domains).

The `/usr/share/sendmail` hierarchy contains all the files used to build the sendmail.cf file from a configuration file. Most of the action takes place in `/usr/share/sendmail/cf`, where you will find several sample configurations, (including `bsdi.mc`, which is where most sites should start, using the other `.mc` files for examples of specific features).

To build a `sendmail.cf` file from a `.mc` file, change directories into `/usr/share/sendmail/cf` and run the following command:

```
# m4 bsdi.mc > /tmp/sendmail.cf
```

where "bsdi.mc" should be replaced with the master configuration file you have configured.

Save your current `/etc/sendmail.cf` and then install the new one. Your created *XXX*`.mc` file and the generated `sendmail.cf` file both have instructions on restarting the server, which must be done for the changes to take affect. The suggested command is:

```
sh -c 'set `cat /var/run/sendmail.pid`; kill $1; shift; eval "$@"'
```

but if you look at the `/var/run/sendmail.pid` file, you can see that the first line is the process id of the `sendmail` process that you need to kill, and the second line is the command you should execute to start the new `sendmail`.

Details of `sendmail` configuration and operation can be found in `/usr/share/sendmail/README` and `/usr/share/doc/sendmail/*`, also available using `man(1)`:

```
% man -M sendmail intro
% man -M sendmail op
```

BSD/OS also includes the `sendmail` verification utility, `checksendmail(8)`. Create a set of typical addresses for your site and call it address.resolve. Run `checksendmail` (probably using `-C` and an alternate configuration file) and examine the output. It should show correct resolution of all addresses (when treated as "To" addresses and when treated as "From" addresses). See the `checksendmail` manual page for additional information.

Sendmail Compatibility

The `sendmail` program in BSD/OS 4.0 does not expand ampersand ("&") characters found in the GECOS field of the system password file to the full user name. Be sure to expand ampersands in the GECOS field of your `/etc/master.passwd` file entries if you have used this feature in the past.

Configuring Netnews

Netnews (the INN server and the news readers) is installed as part of the standard system. If you just want to read netnews and you already have an NNTP server on your local network, you can read news by setting your NNTPSERVER environment variable to the name of your local NNTP server machine and using the remote versions of the newsreaders (`rtin`(1) or `rtrn`(1)).

The rest of these instructions explain how to set up a local news system.

The news server shipped with BSD/OS is the INN (InterNet News) server. This server is structured for efficient news transfer via NNTP or batched `uucp` connections. This document describes NNTP configuration, but configuration for `uucp` feeds is similar (see the `batcher`(8) manual page).

The INN server supports two types of incoming NNTP connections: news feeds and news readers. These are handled by separate configuration files.

News Directory Layout

The news binaries and support programs all reside in the `/usr/contrib/bin` and `/usr/contrib/lib/news` directories. The files in those directories should not need to be modified locally and will probably change from distribution to distribution. The news administrator will probably want to add the directory `/usr/contrib/lib/news/bin` to their shell command search path for ease of use. Regular users should never need to execute programs from that directory.

The news configuration files reside in `/var/news/etc`. This directory includes the `active` and `history` files, so it could grow to require hundreds of megabytes of disk space, depending on how much of a full news feed you receive, how much history you keep, etc. The history file is re-built nightly as old history entries are expired, so it will approximately double in size during the nightly expire run. Make sure you have plenty of free space on the partition that contains `/var/news/etc`.

The news articles themselves are stored in the `/var/news/spool` directory. The space requirements of the spool directory will depend on how many groups you will be receiving and how long you plan to keep articles before expiring them. Ask the site from which you will be receiving news how large their news spool directory is and how long they retain articles – that will give you an idea of how much space you will need.

Current full feeds are on the order of GIGABYTES PER DAY – seven days per week, 365 days per year. Obviously, this is a significant network bandwidth requirement and a significant disk requirement – but represents only a small load on the server.

Multi-gigabyte disks are a good starting place for the /var/news filesystem if you intend to receive even a large fraction of a full news feed.

Note that /var/news is best contained in a single physical partition rather than scattered across several by using symbolic links. The inn news software normally uses hard links for articles posted to multiple newsgroups. Configuring inn to know which groups are on which disk can be very painful.

Obtain active and newsgroups Files

If you are setting up a system to handle networked groups (as opposed to local groups only), you will need to obtain recent active and newsgroups files. You can use anonymous ftp to get the latest versions of these files from ftp.uu.net in the directory networking/news/config or you can get the latest files from the system that will be supplying you with your feed. These files should be copied to:

```
/var/news/etc/active
/var/news/etc/newsgroups
```

If you are setting up a server for local groups only, you can just start with the versions of the active and newsgroups files shipped with the distribution.

Make sure the files are owned by user news and group news and are mode 0664:

```
# chown news.news /var/news/etc/active
# chown news.news /var/news/etc/newsgroups
# chmod 0664 /var/news/etc/active
# chmod 0664 /var/news/etc/newsgroups
```

Edit Configuration Files

The next step in news configuration is to modify the configuration files for your local site. All of these files are located in the /var/news/etc directory and each has a manual page that describes the format of the files. There are examples in each of the distribution versions as well.

- control.ctl – This file specifies the handling of control messages. You shouldn't need to change it to get started.
- expire.ctl – This file controls how articles are expired. The default is to expire articles after 5 days. You should tailor this file to your site. For example, to keep articles 5 days by default and to keep local articles (bsdi groups in BSDI's case) forever, you might use:

```
/remember/:10

##  Default
*:A:1:5:10

##  Some particular groups stay forever...
bsdi.*:A:never:never:never
```

- hosts.nntp – The hosts.nntp file lists the names (or IP addresses if you prefer) of machines that will be feeding you news via NNTP. Also listed in this file will be the passwords required for authentication of the remote system (if you wish to use password authentication). Many (maybe even most) service providers do not use passwords, so be sure to check whether yours does before setting up a password.

- `inn.conf` – The `inn.conf` file specifies information about your local site. Here's a sample file that BSDI uses:

  ```
  domain:            BSDI.COM
  fromhost:          BSDI.COM
  moderatormailer:   news
  pathhost:          news.BSDI.COM
  organization:      Berkeley Software Design, Inc.
  server:            news.BSDI.COM
  ```

- `innwatch.ctl` – The innwatch.ctl file controls the behavior of the `innwatch` script. This should not require any changes to get started.
- `newsfeeds` – The newsfeeds file specifies what you will do with articles when they are received. This is by far the most complicated configuration file and it will typically have at least four uncommented entries: "ME", "overview", "crosspost", and your primary news provider.

  The "ME" entry specifies the default subscription list that is automatically pre-pended to the subscription list for other sites. This is useful for simplifying the other entries in the file. For example, control groups, junk, and the local groups should not be propagated to remote sites. BSDI uses the following "ME" entry:

  ```
  ME\
  :*,!control,!junk,!bsdi.*::
  ```

  The overview entry is used to generate the overview databases used by threading news readers. This entry should work for all sites:

  ```
  # Create overview databases
  overview:*:Tc,WO:/usr/contrib/lib/news/bin/overchan
  ```

  The crosspost entry is used to generate the linked names for all crossposted articles. This entry should work for all sites:

  ```
  # Create links for cross posted articles;
  # see -L option for innd(1)
  crosspost:*:Tc,Ap,WR:/usr/contrib/lib/news/bin/crosspost
  ```

  Other entries are used for feeding news to remote sites (don't forget the entry to feed news back to your main feed provider so locally posted articles are forwarded back to the world). BSDI's entry for our feed back to UUNET looks like:

  ```
  uunet:/!bsdi:Tf,Wnm:
  ```

  See the sample newsfeeds file and the `newsfeeds`(5) manual page for complete information on the newsfeeds file.
- `nnrp.access` – The `nnrp.access` file specifies which hosts are allowed to connect to read news via NNTP (as opposed to transferring news that is specified in the `hosts.nntp` file). Typically, you'll want to let hosts on your

local network connect to read and post news. For example, news.BSDI.COM allows access from any host in the BSDI.COM domain:

```
bsdi.com:RP:::*
*.bsdi.com:RP:::*
```

- `nntpsend.ctl` – The `nntpsend.ctl` file maps the feed name from the newsfeeds file to the fully qualified name of the site and specifies other arguments for the `nntpsend` program. BSDI uses the arguments:

  ```
  uunet:news.uu.net:1m:-T1800 -t300
  ```

  for our reverse feed back to UUNET.
- `organization` – You will need to create the `organization` file. While this file is not used by the INN news system itself, it is used by the local versions of `trn` and `tin` to specify the name of the local organization when the `ORGANIZATION` environment variable is not set. BSDI's organization file contains:

  ```
  Berkeley Software Design, Inc.
  ```
- `passwd.nntp` – The `passwd.nntp` file specifies the passwords to use when connecting to a remote site. This file is not used if you are not using password authentication.

Starting the Server and Scheduling Cron Jobs

The news server is started by the `/var/news/etc/news.rc` script from `/etc/rc.local`. There is a sample line in the distributed version of `/etc/rc.local` to start news. You should uncomment that line (so that news will be started automatically on future system boots) and then start it by hand one time:

```
# su -m -c news news -c /var/news/etc/rc.news
```

You should also set up `/etc/crontab` to run various news commands periodically. There are sample lines in the distributed version of `/etc/crontab`. The `news.daily` script should be run once per day (usually late at night) to expire news, clean up logfiles, etc. The `rnews` command, with the `-U` option specified, should be run periodically to process news articles that were delivered while the local news server wasn't responding for some reason. The `nntpsend` program should be run periodically to transfer news from your machine to your nntp neighbors. The times for this transfer should be changed from the sample version if there are likely to be many similar configured hosts. Lots of people start to transfer news on 0, 15, 30, and 45 minutes after every hour, so those would be particularly poor choices. Try to pick something at odd times.

Once you've started `news` and set up the `cron` jobs, you should be all up and running. Tell your upstream neighbors to turn on your feeds (a small partial feed is better than a full feed to start with since failures cause fewer problems). Once you're sure you've got everything set up right, you can open the floodgates.

Testing Your Configuration

You should test your configuration by creating a local group (see *Creating Local Groups* later in this chapter) and post an article using one of the local news readers. You might have to wait a couple of minutes before the article appears in the newsgroup with the correct threading information, but once it does, check to to make sure the headers are reasonable.

Once local articles look OK, try posting a message to the test group for your local area (e.g., BSDI usually uses `co.test`, Colorado's test group, since we're located in Colorado). In the San Francisco bay area the correct group would be `ba.test`. Some sites have automated responses to articles posted to test groups, so you should expect to see several responses from various sites saying they received your test message.

Also, don't forget to test remote newsreaders from various sites in your domain if you configured the `nnrp.access` file for local NNTP access.

Using ctlinnd for Administration

Many of the administrative commands for news are performed through the `ctlinnd`(8) program. Use `ctlinnd` to tell the daemon to re-read a config file, stop processing articles, create new groups, etc. You should familiarize yourself with `ctlinnd` by reading the manual page. It also has an extensive usage statement if you give it the `-h` (help) option.

The news system keeps extensive logs of its operations in the `/var/log/news` directory. Files there are organized by the interest level of the message. These files are rotated by the nightly `news.daily` script that also summarizes them and forwards the interesting parts to the news administrator (root by default).

Creating Local Groups

Create or remove local groups with the `ctlinnd` command. For example, to create the local group `bsdi.test`, use the command:

```
 # ctlinnd newgroup bsdi.test y polk@BSDI.COM
```

This would create the new group `bsdi.test`, add it to the active file as a group for which you would accept articles (the "y" flag in the active file means this group is accepted), and log the fact that it was created by polk@BSDI.COM.

Also use `ctlinnd` to remove local groups with the `rmgroup` option.

innwatch

The `innwatch` daemon is started from `/var/news/etc/rc.news` and keeps a watch over the daemon, the available diskspace in the spool directory, and other parameters specified in its configuration file `/var/news/etc/innwatch.ctl`. It will typically send email to the news administrator when it finds anything amiss. No additional configuration should be required for most sites.

Other Information Sources

There is lots more information on INN in the INN FAQ that is posted weekly to the news.answers netnews group and is often also available from: `http://www.cis.ohio-state.edu/hypertext/faq/usenet/usenet/software/inn-faq/top.html`.

Configuring Anonymous FTP

BSD/OS now ships with a new ftp server. In the past we have shipped an older version of wu-ftpd with local modifications. Due to the number of security advisories against wu-ftpd and the general poor state of that code, BSDI has moved to a new code base that we intend to support internally. While the new server does not have exactly the same feature set as wu-ftpd, it does have the most desirable facilities. The new server is not intended to be either a drop in replacement or contain every last feature of wu-ftpd.

While BSDI does not recommend its use, the latest wu-ftpd, 2.4.2-beta-16, is included with BSD/OS as `/usr/contrib/libexec/wu-ftpd`. To use wu-ftpd rather than the new ftp server you must change your `inetd.conf` file. Depending on your usage of tcp wrappers, you have a line in your `inetd.conf` file that looks something like one of the following two lines:

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

You should change this line to one of the following (shown here with lines folded):

```
ftp stream tcp nowait root /usr/libexec/tcpd
        /usr/contrib/libexec/wu-ftpd -l -A
ftp stream tcp nowait root /usr/contrib/libexec/wu-ftpd
        wu-ftpd -l -A
```

Note that this version of wu-ftpd does not support virtual hosts via the `-F` and `-m` options. Please review `ftpaccess`(5) and read about the “virtual” configuration option. This version should be better than the version shipped with earlier BSD/OS releases and is maintained by `wu-ftpd-bugs@academ.com`. (The support for `login.conf`(5) is a local BSDI addition. Future releases of BSD/OS will only include this modification if it is accepted back by the maintainer.) There may be other differences that BSDI is not aware of.

BSDI *does recommend* the use of the default ftp server, `ftpd`. If you are performing a new installation of BSD/OS you will not need to make any changes to the `/etc/inetd.conf` file. If this is an upgrade you many need to change the ftpd line of the `/etc/inetd.conf` file. There should be no options on the ftpd line of the file. All options can be specified in the `/etc/ftpd/config` file .

For a full description of the `/etc/ftpd/config` file please review the `ftpconfig`(5) manual page.

There are 3 major features of ftpd that deserve discussion here. Incoming directories, virtual hosting, and session limits.

The config file contains variables, flags, and directives. These may be limited to only guest accounts, a specific virtual host, or guest accounts for a specific virtual host. To limit a block of the config file to guest accounts you must start that block with the line:

```
<Guest>
```

and end the block with

```
</Guest>
```

These must be on their own lines and not be included with other text. The ftpd server has a built in version of the `ls(1)` command. This means you no longer need a `bin` directory or an `shlib` directory in your guest directory. You still need an `etc/pwd.db` and `etc/group` if you want `ls` to list symbolic user/group names rather than numbers. (This is not required.)

To start a virtual host block (and define that virtual host) you must start the block with

```
<VirtualHost hostname>
```

And end the block with

```
</VirtualHost>
```

Incoming directories are restricted for guest (anonymous) sessions. Only directories (and their descendents) explicitly listed in the config file may have files stored in them. The *Incoming* directive is only valid within a guest session. For example, suppose you want to have a directory called "incoming" at the top level of your guest ftp directory. If you wish to allow guests to deposit files here (but not read or delete them) and you want the resulting files to be owned by the account "support" and the group "archive." Within your guest block you should have the line:

```
Incoming /incoming support archive 660
```

This will allow guests to deposit files in the `/incoming` directory (this is relative to the directory used for guest sessions). Files deposited there will be owned by *support*, group *archive* and have the mode of 660 (read/write by user and group). Guest users will not be able to read or delete these files. Of course, you must first create the *incoming* directory. To do this, first change directories to the directory used for guest sessions. Then:

```
# mkdir incoming
# chown support.archive incoming
# chmod ug=rwx,o=wx incoming
```

Note that any subdirectory (up to 8 levels deep) from `incoming` that is writable by the guest ftp account will also be marked as an incoming directory. To allow guests to create their own directories you can change the `Incoming` line to read:

```
Incoming /incoming support archive 660 773
```

The last argument both allows subdirectory creation (make sure the `MKD` command is turned on!) and specifies the mode of newly created directories. Just like files, they will be owned by *support* and have group *archive*. Mode 773 is the same as ug=rwx,o=wx.

You may want to create a ``.message'' file in the Incoming directory. This file will be displayed each time a user enters this directory.

Virtual hosts, as indicated above, are started with the line:

```
<VirtualHost hostname>
```

and end with the line:

```
</VirtualHost>
```

Everything between these lines applies only to that virtual host. The hostname specified should resolve to a single IPv4 or IPv6 address. (If it resolves to more than a single address only the first address found will be used).

A typical virtual host block would look like:

```
<VirtualHost ftp.mycompany.com>
    AnonymousDir    /var/spool/ftp.mycompany.com
    StatFile        /var/log/ftpd/stats-ftp.mycompany.com
    <Guest>
        MKD         On
        Incoming    /incoming root wheel 600
        Incoming    /support support archive 660 773
    </Guest>
</VirtualHost>
```

The ftp server also has the ability to manage the number of guest sessions allowed. To use this facility, ftpd must not be started from `inetd.conf`. You must comment out the ftpd line in the `/etc/inetd.conf` and start ftpd from `/etc/rc.local` with the command:

```
/usr/libexec/ftpd -D -p /var/run/ftpd.pid
```

You can cause ftpd to re-read the configuration file by sending it a SIGHUP:

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

To specify the maximum number of guest sessions that are allowed, set the MaxUsers within a guest block:

```
<Guest>
    MaxUsers    50
</Guest>
```

This will limit the number of guests (per virtual host) to 50. Both guest and non-guest sessions are counted, but only guest sessions are limited. If you had 50 non-guest users currently logged in, then guest users would not be able to log in. As indicated, this may be specified in a virtual host block. Each virtual host keeps track of its own number of sessions.

Configuring UUCP

UUCP is a suite of programs that use ordinary serial lines (usually with modems) to send files and other information from the BSD/OS system to another computer running UUCP. UUCP can be used to join the networked mail and Usenet news community if your site lacks a dedicated Internet connection.

Etymology

The name of the "UUCP" (ewe-ewe-see-pee) suite derives from the phrase "**U**nix to **U**nix **C**o**P**y".

Configuration

UUCP has several configuration files, which are found in the `/etc/uucp` directory:

- `/etc/uucp/CONFIG`: specifies all of the parameters for `uucp`. Execute `uuparams`(8) and read its manual page for more information.
- `/etc/uucp/L-devices`: defines the modem lines and dialers for `uucp` to use. For example:

  ```
  DIR tty01 unused 57600 unused "" P_ZERO
  ACU tty04 unused 38400 hayesvtone "" ATM0S7=90 OK ""
  ACU tty04 unused 19200 hayesvtone "" ATM0 OK ""
  ACU tty04 unused  9600 hayesvtone "" ATM0 OK ""
  ACU tty04 unused  2400 hayesvtone "" ATM0 OK ""
  ACU tty04 unused  1200 hayesvtone "" ATM0 OK ""
  ACU tty05 unused 38400 hayesvtone "" ATM0S7=90 OK ""
  ACU tty05 unused 19200 hayesvtone "" ATM0 OK ""
  ACU tty05 unused  9600 hayesvtone "" ATM0 OK ""
  ACU tty05 unused  2400 hayesvtone "" ATM0 OK ""
  ACU tty05 unused  1200 hayesvtone "" ATM0 OK ""
  ```

  says that `tty01` is a 57600 baud **DIR**ect connection to some other machine (whose name will be given in the `L.sys`(5) entry that refers to this device). No parity is used on this line; the default is even parity for compatibility with older systems.

  Next, `tty04` and `tty05` are said to be modems needing the `hayesvtone` driver, meaning that DTMF is to be used rather than pulse dialing and the modem's command processor is likely to be broken in some way (as on almost all modern so-called Hayes compatible modems). At all baud rates the speaker is turned off, and at 38400 baud the dial timeout is set to 90 seconds since higher carrier rates can take longer to synchronize after the remote modem answers.
- `/etc/uucp/L.sys`: specifies the names of remote machines that `uucp` calls or that call this machine. This file also includes the "chat scripts" that teach `uucp` how to login to remote machines in addition to calling hours and modem types to use. For example,

```
asacomp Never ACU 57600
vixpc   Any   DIR 57600 tty01       in:~5--in:~5 Uvixie word: foo
decwrl  Any   TCP uucp  192.168.1.1 in:~9--in:~9--in:~9 uvixie rd: bar
decwrl  Any   ACU 19200 5551212     in:~9--in:~9--in:~9 uvixie rd: bar
```

In this example, the system `asacomp` is allowed to call us but we don't call them. (The baud rate in this example is just a placeholder, necessary, but irrelevant -- incoming calls know their baud rate.) Next, `vixpc` is a 57600 baud connection over some **DIR**ect device given in `L-devices`(5) (where the only other search criteria is that the line support 57600 baud.) If the character string `in:` (the last part of the string "login:" from the prompt sent by the remote machine) is not forthcoming across the connection in 5 seconds, send a CR. If 9 more seconds elapses without an `in:`, send another CR. If 9 more seconds elapses with no `in:`, give up. If an `in:` arrives, send **`Uvixie`** and wait for a `word:` prompt (the last part of "Password:") and if this does not arrive in about 30 seconds (the default), give up. Finally we see **decwrl** reachable by either TCP/IP (via the **uucp** service on port 540) at the IP address shown, or via a modem connection. If a TCP/IP connection isn't possible for some reason, a 19200 baud phone call will be made in hopes of reaching one of **decwrl**'s modems.

- `/etc/uucp/L.cmds`: a rarely edited file that specifies which commands `uuxqt` can run - such commands are entered via `uux` commands on your neighboring machines.
- `/etc/uucp/USERFILE`: a rarely edited file that specifies which directories `uucp` can access.

The manual pages for each of the above files contain more information (e.g., `man L.sys`). In addition, the installed files contain sample data that is probably similar to what you'll want.

For additional information on UUCP installation and configuration, we recommend *Managing UUCP and Usenet*, referenced in the chapter entitled *Further Documentation*.

Dialer Notes

The "hayes" dialer is the dialer of choice for the vast majority of modern modems. You might also try the "hayesv" dialer if the hayes dialer is problematical (particularly on Telebit modems).

Problem Isolation

Check the output from the `-a` option to `uuparams` to make sure that all the necessary directories have been created (like those for LOGDIR, SPOOLDIR, and LOCKDIR) and that your NODENAME is correct. Use the command:

```
# /usr/libexec/uucico -Ssystemname -x9
```

to debug the connection phase.

uucp's -n Option

The argument for the `-n` option to `uucp`(1) is limited to 8 characters. To reach longer addresses, use an alias in `/etc/aliases`.

Running UUCP

Once you have everything working, you will probably want to set up `/etc/crontab` to run `uuxqt`(8) so that everything is fairly automatic. For example:

```
 # once an hour check for any UUCP jobs
 49 * * * * uucp /usr/libexec/uusched
```

The simple-minded example above will just get you started. It dials at 49 minutes after every hour, a time less likely to be used by other sites than, say, exactly on the hour. Unless both your machine and your UUCP partner's machine call each other on demand, one or both of you should poll the other at an appropriate interval.

Configuring gopher

BSDI does not ship the University of Minnesota ASCII `gopher` client and `gopherd` server due to licensing restrictions from the University of Minnesota. BSDI does ship the X Window System-based `xgopher` client as part of the regular distribution, and any of the WWW viewers, like Netscape Navigator and Mosaic (including the ASCII WWW viewers `www` and `lynx`) may be used to access `gopher` servers on the Internet. To access a `gopher` server using one of the WWW browsers, specify an appropriate URL (e.g., `gopher://hostname/` to access the top level gopher directory on the specified host).

Obtaining UMN Gopher

The UMN `gopher` software (and licensing information) is available from the University of Minnesota via anonymous `ftp` from the machine **boombox.micro.umn.edu** in the directory `pub/gopher/Unix`. As of this writing, the most up-to-date version is `gopher2_3.tar.gz`. This was also the most recent version with the BSD/OS release 3.0.

Configuring TeX

The BSD/OS TeX package is based on Thomas Esser's <te@informatik.uni-hannover.de> teTeX distribution, with minimal changes to build on BSD/OS and to install into a single directory tree rooted at `/usr/contrib/teTeX`.

The three packages retrieved from

```
ftp://sunsite.unc.edu/pub/Linux/apps/tex/teTeX/distrib/sources
```

were

```
teTeX-src-0.4pl7.tar.gz
teTeX-lib-0.4pl8.tar.gz
texmf-src-0.4pl8.tar.gz
```

These packages use the TeX Directory Structure Standard and provide `LaTeX2e`, `psnfss`, `metapost`, and so on. See the files in the directory `/usr/contrib/teTeX/texmf/doc/tetex` for more details. A few files were removed in BSD/OS because their copyright notices did not permit us to redistribute them; see `/usr/contrib/teTeX/README.missing-files` for details. The missing files can be retrieved from the original packages at the official distribution sites.

Installing TeX

Use `installsw`(8) to install the "TeX" package. It installs into `/usr/contrib/teTeX`, and requires approximately 65 MB of disk space. The superuser may update teTeX from its default configuration settings by running the `texconfig`(1) program.

Setting Up To Run TeX

All TeX binaries delivered with BSD/OS 4.0 can be found in `/usr/contrib/teTeX/bin` and are built to search for their various resources in the `/usr/contrib/teTeX` tree. To use the TeX tools, make sure that `/usr/contrib/teTeX/bin` is in your path. Users of `csh` can do this by changing the "`set path=...`" line in their files; `sh`, `ksh`, and `bash` users add it to the "PATH=`...`" line of their `.profile` files.

A Quick Introduction to TeX Resources

The Comprehensive TeX Archive Network (CTAN) is a resource for TeX-related software, styles, and fonts. The principal CTAN URLs are:

```
USA:     ftp://tug2.cs.umb.edu/tex-archive
         http://tug2.cs.umb.edu/ctan/
UK:      ftp://ftp.tex.ac.uk/tex-archive
         http://www.tex.ac.uk/tex-archive/
Germany: ftp://ftp.dante.de/tex-archive
         http://www.dante.de/
```

Configuring SCO/iBCS2 Emulation

BSD/OS 4.0 contains a feature called SCO/iBCS2 binary emulation. This feature is intended to enable customers to run some third party software applications for SCO Unix under BSD/OS. It is not intended to duplicate a SCO environment, but rather to permit third party software for SCO to be integrated into the BSD/OS environment insofar as this is possible.

How to Use SCO/iBCS2 Emulation

The simplest way to use emulation is to run a "sco shell":

```
$ sco sh
```

The "sco shell" command enables SCO emulation in the `sh` shell process. The emulation modifies the pathnames seen by the shell and its children so that typical SCO pathnames are changed into BSD/OS pathnames; for example, the command `ls /usr/tmp` shows the contents of the equivalent BSD/OS directory, `/var/tmp`. When you install third party software, you should always start up a "sco shell", typically as root, so that all the shell scripts used in the installation will have their pathnames modified. The emulation supplies a replacement for the SCO `/etc/custom` program to install pre-packaged software.

Once the third party software is installed, a "sco shell" is still the most convenient environment in which to run the software. However, third party SCO executable binary files can be run directly, without any need for a "sco shell", when this is more convenient. Third party software that uses cursor addressing on the console or runs as an X client generally works fine under emulation. Products that have been successfully installed and run under SCO/iBCS2 emulation include Lotus 1-2-3 v1.1, WordPerfect v5.1 and v6.0, FrameMaker 3.1, and Informix 4.1/5.0. Please call BSDI directly if you have trouble loading these programs.

SCO/iBCS Emulation and Design Overview

"iBCS2" stands for the "Intel386 Family Binary Compatibility Specification 2", a software standard for System V Release 3 Unix on Intel CISC platforms. A number of SVr3 suppliers have claimed adherence to this standard, although there are some variations; this emulation tries to support the SCO Unix variant. The iBCS2 specifies a binary file format called COFF, a number of "system calls" to access kernel services, some shared libraries and some header files. The BSDI emulation provides enough of these services to support third party software, but no more; for example, the header files specified by iBCS2 are not supplied.

The fundamental underpinning of the emulation is the "emulator library". This is a program that mediates between the other programs and the operating system kernel, providing various translation services.

This emulator receives system calls from SCO programs and either services them itself or propagates them to the kernel, converting data types between SCO and BSD/OS formats as necessary. For example, the emulator understands that SCO programs generally use `read`(2) calls to access directory elements, and translates these calls into BSD/OS `getdirentries`(2) calls.

The emulator maintains a filename mapping service: when an emulated program opens a file, the emulator checks a file mapping table and substitutes an alternate pathname if the requested pathname matches one of a set of prefixes. This service causes files to appear to be in the places where SCO programs expect them, without using symbolic links or other clutter in the filesystem. BSD/OS programs that run under the emulator receive a subset of the services required by SCO programs, principally filename mapping.

There are other pieces of the emulation in addition to the emulator proper. The package contains shared C library emulation, which translates SCO standard I/O calls to BSD/OS standard I/O and other similar services and a Network Services Library emulation that converts TLI services into corresponding BSD socket services. The `sco`(1) program exists mainly to load the emulator into a BSD/OS program; it also manipulates the $PATH and $DISPLAY variables to enhance the emulation. Note that BSD/OS programs that are invoked by SCO programs are automatically loaded with the emulator.

Limitations of SCO/iBCS2 Emulation

Although the iBCS2 standard provides a useful starting point, that's all it is. Everything we know about SCO Unix beyond iBCS2 comes from public information and observing the behavior of third party software under the emulator; thus, the emulation is bound to be incomplete. When the emulator or a shared library encounters such a dark area, it might abort. Customers should report such conditions to BSDI support, and we'll try to puzzle them out and emulate them in the future.

There are a number of known shortcomings; here is a list of some of the more noteworthy ones:

- The emulation does not support Xenix binaries. This rules out loading some third party software for SCO from Microsoft, such as Microsoft Word.
- The emulation also cannot run software that requires installation of drivers into the SCO kernel; this rules out products such as DoubleVISION.
- There is no environment for compiling SCO sources or linking SCO objects under BSD/OS; this means that custom database applications or spreadsheet add-ins cannot be created under BSD/OS, although the emulator will run them if they have already been built under SCO.
- Several system calls and many specialized ioctl() calls are not supported by the emulation. The unsupported system calls fall mainly into two categories: system administration and STREAMS. Lack of STREAMS support prevents local access to the X server from SCO X clients; however, TCP access to the

local server works just fine (and the `sco(1)` program modifies $DISPLAY to make TCP connections be the default).

Since SCO doesn't support BSD job control, there are some situations where BSD job control won't stop a SCO program with ^Z. The emulation supplies `terminfo` files only for the console and for `xterm` windows. Some third party software for SCO expects specific System V behavior from certain standard programs; the emulation supplies substitutes for a few of these programs, but most have no replacements. Other deficiencies are listed in the BUGS sections of some of the manual pages.

Useful manual pages to see for other information include `sco(1)`, and `ibcs2(5)`.

Configuring NFS

Configuring an NFS Client

The Network File System (NFS) enables remote access to files as if they were local. Generally, an NFS client "mounts" a filesystem from an NFS server. BSD/OS can act as both an NFS client and a server, simultaneously.

The BSD/OS system is automatically configured to act as an NFS client. To mount a remote NFS filesystem one time, use the `mount`(8) command:

```
# mount rhost:filesystem local_filesystem
```

The only difference between this usage and a local mount command is the use of a colon ":" to separate the name of the remote host and the filesystem being mounted. For example, to NFS mount the `/usr/src` filesystem from the system `foo`, on the local filesystem `/a/mysource`, use the command:

```
# mount foo:/usr/src /a/mysource
```

Note, you must export the absolute pathname of this filesystem. To determine the absolute pathname of a directory, `cd` to the directory and then enter the `/bin/pwd` command. You must enter the full command; entering `pwd` might be insufficient as some shells have a built-in implementation of `pwd`, which can be fooled by symbolic links. Be especially careful when exporting directories in the `/var` filesystem, which is often a symbolic link to `/usr/var`.

To unmount an NFS filesystem, use the `umount`(8) command exactly as you would unmount a local filesystem.

If the remote file system is to be mounted at each reboot, add an appropriate line to `/etc/fstab`. In our example, the additional line would be:

```
foo:/usr/src  /a/mysource  nfs rw 0 0
```

Other options can improve performance and robustness in case of network problems; see `mount`(8)
and `mount_nfs`(8).

Configuring an NFS Server

If the `/etc/exports` file exists when the system is booted, the machine is automatically configured as an NFS server by starting the `mountd`(8) daemon, which is necessary to export filesystems via NFS. To become an NFS server, modify the `/etc/exports` file to specify the filesystems that you wish to export, and reboot your system.

The /etc/exports file does not protect sufficiently against many NFS security breaches. You should filter port 2049/udp and 2049/tcp using packet filters to achieve better security.

The `/etc/exports` file has lines that specify filesystems (or mountpoints within filesystems), hosts that can access those filesystems/mountpoints, and flags to modify permissions or specify other parameters. Here is a tiny `/etc/exports` file that allows the systems `client1` and `client2` to mount the /home filesystem read-write, client3 to mount the /usr/src filesystem read-only, and client4 to mount the /usr filesystem read-write with the remote root user getting full access:

```
/home          client1 client2
/usr/src       -ro client3
/usr           -maproot=0:0 client4
```

See the exports(5) manual page for additional information.

Once the mountd daemon is already running, you can modify the /etc/exports file and notify the mountd daemon of the reconfiguration at any time, by using the following command:

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

Note the use of backquotes in the command! Your changes will not take effect until you have done this.

You can examine the list of NFS exported filesystems for any system by using the command:

```
# showmount -e hostname
```

where hostname is the name of the system.

PC/NFS Configuration

When using PC/NFS, first change /etc/rc to start mountd with the -n flag.

Then, uncomment the line:

```
echo -n pcnfsd; /usr/contrib/bin/pcnfsd
```

in the /etc/rc.local file.

You must also export, using /etc/exports, the filesystem containing the directory tree used by pcnfsd(8) for printer services.. This directory tree is /usr/spool/pcnfs by default, but can be set explicitly in the PC/NFS configuration file, /etc/pcnfsd.conf.

Finally, reboot the system to cause your changes to take effect.

NFS Version 3

Releases of BSD/OS prior to release 3.0 supported only NFS Version 2. More recent versions of BSD/OS support both NFS Version 2 and the newer NFS Version 3. NFS Version 3 has features that make many kinds of transfers run faster, particularly large writes to files.

By default, the BSD/OS client will always try to mount remote filesystems using NFS Version 3. If this attempt fails, the client will try mounting the filesystem using NFS Version 2. The options -2 (long form nfsv2) and -3 (long form nfsv3) to mount force the use of NFS Version 2 and NFS Version 3, respectively.

The BSD/OS server will respond to both NFS Version 2 and NFS Version 3 requests.

Note, the current `amd`(8) implementation always uses NFS Version 2 unless the `nfsv3` map option is specified.

Network Lock Manager

NFS provides support for coordination of simultaneous changes to exported/imported files via the Network Lock Manager protocol. BSD/OS 4.0 is the first BSD/OS release to support this, and it is implemented via the `lockd`(8) daemon.

NFS with Sun Microsystems Machines

People at a small number of sites have had trouble NFS mounting filesystems from Sun Microsystems on BSDI machines. This is due to the Sun machines requiring that the port number used by NFS be within a certain range. The workaround for this problem is to use the `resvport` option to the BSDI `mount` command in the `/etc/fstab` file, or use `-o resvport` when you invoke `mount` directly.

Troubleshooting NFS

Note that errors from `mountd`, et al., are logged in `/var/log/daemon.log`. A complete reference on administering NFS can be found in O'Reilly's *Managing NFS and NIS* book.

The most common problem encountered when configuring NFS is the error message:

```
bind: address already in use
```

This error typically refers to one of the following problems:

1. Authentication errors.

   The information in the `/etc/exports` file on the NFS server is incorrect or incomplete. Check for typographic errors and to ensure that your client name is properly specified. Also, ensure that your client host is properly defined in the Domain Name System (DNS). Refer to the section *Configuring the Domain Name System* in this Installation Guide, and in the *DNS and BIND* book referenced in the chapter entitled *Further Documentation*.

2. Remote Procedure Call (RPC) errors.

   RPC is the mechanism used by NFS for its' client/server model of computing. An RPC daemon named `portmap`(8) is started, that is used by NFS and other programs to register themselves for use by other network programs. In NFS's case, this includes `mountd`, `nfsd` and `nfsiod`. For NFS to function properly, these daemons must be started in the correct order to ensure proper RPC registration.

If you still see the same error message after checking for the problems described above, you should kill the NFS and RPC daemons and restart them. An example of this would be:

```
# ps ax | egrep 'portmap|mountd|nfsd|nfsiod|lockd|statd'
23  ??  Ss     0:01.23 portmap
```

```
41  ??  Ss     0:10.49 mountd
43  ??  Is     0:00.17 (nfsd)
45  ??  IW     0:00.46 (nfsd)
47  ??  IW     0:00.05 statd
49  ??  IW     0:00.10 (lockd)
51  ??  IW     0:00.03 (lockd)
87  ??  I      1:41.22 nfsiod 4
88  ??  I      1:11.74 nfsiod 4
# kill 23 41 43 45 47 49 51 87 88
# portmap
# mountd
# nfsd -u -t 6
# /usr/libexec/statd
# /usr/libexec/lockd
# nfsiod 4
```

Make sure that you restart them in the same order as the example above!

NFS-related Commands and Files

There are many manual pages and documents that describe parts of the NFS system. This section gives brief references to the more important ones. Find the entire list (and a few more) using

```
% man -k nfs
```

nfsd – NFS server

The `nfsd`(8) server runs on a server machine to service NFS requests from client machines. At least one `nfsd` must be running for a machine to operate as an NFS server.

mountd – Service Remote NFS Mount Requests

The `mountd`(8) daemon services NFS mount requests from client machines.

nfsiod – Local NFS Asynchronous I/O Server

The `nfsiod`(8) daemon runs on an NFS client machine to service asynchronous I/O requests to its server.

lockd – Remote and Local Lock Request Server

The `lockd`(8) daemon runs on both the NFS client and server. Running lockd will create two processes. The parent lockd will service NFS lock requests from client machines, and the child lockd will issue NFS lock request to server machines.

statd – Host Status Server

The `statd`(8) daemon runs on both the NFS client and server. It is used by lockd to determine when clients have rebooted, so their locks can be released.

exports** – **Define Remote Mount Points For NFS Mount Requests

The `/etc/exports` file determines what local filesystems are exported to clients by `mountd`. After editing `/etc/exports` you must notify `mountd`(8) for the changes to take effect, as described above. See the `exports`(5) manual page for additional information. Additionally, `showmount`(8) and `nfsstat`(8) can you help you monitor your system's security and performance.

Configuring Filesystem Quotas

The BSD/OS quota system restricts the number of files and/or file space available to users, both as individual users and as groups of users. It supports both "soft" and "hard" limits.

Hard limits cannot be exceeded. Once reached, a hard limit prevents a user from allocating *any* more storage space. Soft limits can be exceeded for a configurable grace period. Once a soft limit has been exceeded for more than the grace period then the soft limit becomes hard. Soft limit grace periods can be set on a per-filesystem basis.

To use filesystem quotas, your kernel must have been configured with the lines

```
options QUOTA
```

in the configuration file (if not, then rebuild it). The distributed kernel contains this option by default.

Quotas are configured on a per-filesystem basis. To apply quotas to a filesystem, add the options `userquota` and/or `groupquota` to the filesystems in `/etc/fstab` that you wish to run quotas on. For example:

```
/dev/wd0d /home ufs rw,groupquota 1 3
/dev/sd1a /var/tmp ufs rw,userquota 1 3
```

Run `quotacheck -a` to initialize the quota files on each filesystem, and then run `quotaon -a` to enable the quota system itself. **Note:** Both of these operations are done automatically at system boot; you only have to do them by hand if you want to initialize the quota system without rebooting your system.

Once everything is initialized, use `edquota`(8) to edit individual quotas. The `edquota` command will invoke your editor with a simple form to fill in. For example, `edquota sanders` gives:

```
Quotas for user sanders:
/u: blocks in use: 75616, limits (soft = 0, hard = 0)
    inodes in use: 3029, limits (soft = 0, hard = 0)
```

A value of zero for either a hard or soft limit indicates that no quota should be imposed. A value of one for a hard limit indicates that no allocations should be permitted. A value of one for a soft limit indicates that allocations should be permitted on only a temporary basis (as determined by the grace period).

Quotas can be set for either blocks or inodes. Blocks are the actual storage units that hold data. Inodes are reference pointers to blocks. A filesystem might run out of either with similar results. A filesystem that has run out of available references to storage units is just as full as one that has run out of storage units. Usage will determine which quantity is the limiting factor for filesystem capacity. A filesystem with many small files will use inodes in a greater proportion to blocks than one in which large files predominate.

You don’t have to edit each user’s information separately; `edquota` can be used to copy one user’s quota settings to another user (see the manual page for details).

Users can use the `quota`(1) command to display their quota information. Use `repquota`(8) to get a summary of all users’ usage.

Note: Quotas are specified in terms of blocks of BLOCKSIZE bytes.

Configuring RADIUS

BSD/OS includes the Basic Merit AAA Server for the RADIUS protocol. It is provided by Merit Network, Inc. and The Regents of the University of Michigan. More information about this RADIUS server, along with information about the Enhanced Merit AAA Server is available at their website: `http://www.merit.edu/aaa`

BSD/OS 4.0 includes the Merit AAA RADIUS server. Sources for this server may be obtained directly from Merit Network, Inc. The older Livingston based server is no longer supported, though the sources for it may be found (when mounted correctly) in:

```
/usr/src/contrib/radiusd
```

on the contributed source CD. If not mounted as suggested it will be found relative to the CD's mount point:

```
contrib/radiusd
```

The Merit AAA RADIUS server has builtin support for BSD Authentication. The new authentication type "BSD-Auth" has been added to support BSD Authentication. This may be used in the `/etc/raddb/users` and `/etc/raddb/authfile` configuration files. See `authfile`(5) for more details on the BSD-Auth authentication type.

By default, the RADIUS server is not enabled and the configuration files associated with it are not installed or upgraded. To install the configuration files the command

```
# radiusc install
```

should be used. Once the standard RADIUS configuration files have been installed they should be customized for the local site. Please read the `/usr/contrib/lib/radiusd/TUTORIAL` for more information about `radiusd` configuration.

To actually enable the RADIUS server, use the command

```
# radiusc start
```

See the `radiusc`(8) manual page for additional information about the radiusc command.

To use a RADIUS server on another machine for authentication (that is, be a RADIUS client), please review the `login_radius`(8) manual page.

About Contributed Software

BSD/OS 4.0 includes many software packages written and maintained by other organizations and individuals, in the hope that they will be useful to our customers. BSDI does not directly support some of these programs. While we would certainly like to hear about any problems that you might experience in using contributed software, we might choose to refer technical questions and problems to the program maintainers. The source code for these programs is provided to all BSDI customers, including those not purchasing "source distributions". This source code is distributed on a separate CD-ROM that is usually mounted as the `/usr/src/contrib` directory. Programs which are considered contributed are usually found in the directory `/usr/contrib/bin`.

Each of the contributed packages includes a file named "BSDI_CONTRIB". This file always notes what version of the contributed software is included and the location from which BSDI retrieved it. It might also include contact information for the software maintainers, information as to the terms and conditions under which the package might be further redistributed, and the changes BSDI made to the package to integrate it into BSD/OS.

The `cdctl` suite of commands is also unsupported.

Adding a Second Disk Drive

Be sure to backup your system before adding new storage media.

As larger systems mature, it often becomes necessary to add more disk space. The first thing to decide is which parts of the directory tree require additional disk space. The df(1) command will display how much disk space is free on each of the currently mounted file systems:

```
# df
Filesystem 1K-blocks    Used  Avail Capacity  Mounted on
/dev/wd0a      15599    5389   9430    36%    /
/dev/wd0h     346807  312840  16626    95%    /usr
mfs:24         15599      19  14800     0%    /tmp
```

You can see from the df output that the /usr directory of the tree is close to being full. It might be time to get another disk and move some of the data from the /dev/wd0h partition to the new disk.

The question then becomes one of determining which directory of the /usr partition should be moved to a different disk. There are general guidelines for disk partitioning on BSD/OS systems that might be helpful here. See the section *Goals of BSD/OS Partitioning* in the *Automatic Disk Configuration With disksetup* chapter for more information.

The du(1) command displays how much disk space is used by a directory hierarchy, i.e., a directory and everything under it. You can use du on directories inside of filesystems to see which directories are using the most disk space in the filesystem. It is most likely one of these that you will want to move to a new disk. Common directories to check for significant disk usage are the locations of the user home directories and the news directory:

```
# du -s -x /usr/home /usr/var/news
183151 /usr/home
289090 /usr/var/news
```

It is clear that on our example system the news directory is using much more disk space then the home directories. So, if a new disk were to be added to this system, the correct choice would be to move the news directories from /usr to the new disk. For our example, we are going to move the news storage to take up half of the new disk. One quarter of the new disk will be used for additional home directory space, mounted under the directory /usr/home1, and the last quarter will be used as additional swap space.

The first step is to partition the new disk. It is best to do this while running in single user mode, but it is not required. To take the machine down to single user mode enter the command:

```
# shutdown now
```

while running as the superuser, i.e., "root". Note, if there are currently other users on the system, you should give them some warning of the system's shutdown and time to stop gracefully what they are doing! See `shutdown`(8) for more information about how to do this.

When `shutdown` finishes running, a single-user prompt (normally "#") will appear on the system's console.

You should then run the `disksetup`(8) command on the new disk. We will briefly describe how to use `disksetup` to add additional disk drives to your system in this chapter. For a more complete discussion of `disksetup`, see the chapter entitled *Automatic Disk Configuration With disksetup*.

To add an IDE disk to the system, you should use the disk name `wd1`:

```
# disksetup -i wd1
```

and to add a SCSI disk to the system, you should use the disk name `sd1`:

```
# disksetup -i sd1
```

where "1" is the second disk drive on the system, "2" would be the third, and so on.

The `disksetup` command will display a number of different screens and ask you a series of questions.

- Do you wish to share this disk with other operating systems? [NO]

  BSD/OS may be installed as the only operating system on the disk, or, sharing the disk with other operating systems, such as DOS.

  Since this is an additional disk you're adding, you will probably want to answer NO to this question and create a BSD/OS-only disk drive.
- Disksetup will then display a geometry table description screen.

  Just press <ENTER> to continue.
- If the disk you're adding has no existing BSD geometry information, `disksetup` will display a screen asking how it should set the disk geometry. You should choose the option "P" to "Probe" the disk. This will almost always retrieve the correct geometry from the disk. In a few cases this might fail for SCSI disks. When that happens, you should manually enter the correct disk information.

  When manually entering the disk information for a SCSI disk that fails the Probe option use a geometry of 2048 sectors/track, 1 head, and however many cylinders are required to get as close as possible to the disk's total sector count without exceeding it.
- If there was already a partition table on the disk, you will be prompted if you want to start with a clean partition table. You will probably want to answer YES to this prompt, unless the disk already has the correct partition table installed.
- The next prompt will ask you if you wish to use the standard BSD/OS disk partition table. You should answer NO to this question. The default partition table is the correct choice for the boot disk, but it is **not** correct for second and additional disks.
- The `disksetup` command will now display the disk partition table.

```
     BSD Partitioning                      wd1 Device to Partition
     <?> for help
 |.-------------------------------------------------------------------|
   FS    Mount   | Length of File System in | Starting|  Ending| Sector
   Type  Point   | Sectors ( MBytes    Cyls)|   Sector|  Sector|    Gap
 ---------------|--------------------------|---------|--------|-------
 a -----         |                          |         |        |
 b -----         |                          |         |        |
 c -----         | 2503872 ( 1222.6  2484.0)|       0 |2503871 |2503683
 d -----         |                          |         |        |
 e -----         |                          |         |        |
 f -----         |                          |         |        |
 g -----         |                          |         |        |
 h -----         |                          |         |        |
 ---------------|--------------------------|---------|--------|-------
   At this point you should define, alter, or delete the file systems.
   BSD/OS requires at least a root filesystem (/), a swap partition and
   a /usr filesystem.

    If you do not understand this screen, press X and then Y to confirm
    the cancel.  When asked if you want the standard setup, answer YES.

    [I]mport [A]dd [D]elete [E]dit [N]ext [X]Abort [?]Help
```

Note that only the "c" partition will be defined. The "c" partition is always defined as the entire disk, and this cannot be changed.

In our example we will first create a partition for the new `/usr/home1` directory, that is approximately one quarter of the disk's capacity. As you add partitions, `disksetup` will automatically add a 'root', 'swap', and '/usr' partition to this new disk. Do not despair the apparently overlap of the names – the names will be set correctly in `/etc/fstab` later.

Press "A" to add a partition:

```
 a 4.2   /        |    24192 (   11.8    24.0)|       0 |  24191 |
```

We then change the capacity to be one quarter of the disk. In the "Length of File System in Sectors" column enter the number of megabytes. `Disksetup` will automatically set the cylinder count for you. In our example, we enter `307M`:

```
 a 4.2   /        |   627984 (  306.6   623.0)|       0 | 627983 |
```

The only thing that is important when adding a second disk is the capacity. The Mount Point information and the FS Type fields are only important during the original installation of BSD/OS, but they may be set as well, if you choose. We then press <Enter> to accept this information.

Then, we select "A" again, to add the second partition, the one we'll use for additional swap space:

```
 b swap  swap     |    98784 (   48.2    98.0)|   627984| 726767 |
```

This will add the "b" partition. We will change the capacity to one quarter of the disk (307 MB) for this partition as well, entering `307M` in the "Length of File System in Sectors" field:

```
 b swap  swap     |   627984 (  306.6   623.0)|  627984 |1255967 |
```

We then press <ENTER> to accept this information, and press "A" again to add a partition for the news files:

```
 h 4.2   /usr     |  1246896 (  608.8  1237.0)| 1255968 |2502863 |
```

In this case the default size was set to the remainder of the disk, so it isn't necessary to edit the "Length of File System in Sectors" field. We then press <ENTER> to accept this information, and the entire table should look as follows:

```
     BSD Partitioning                         wd1 Device to Partition
     <?> for help
 |.-------------------------------------------------------------------|
   FS    Mount     |  Length of File System in | Starting|  Ending| Sector
   Type  Point     |  Sectors ( MBytes    Cyls)|   Sector|  Sector|    Gap
 -----------------|--------------------------|---------|--------|-------
 a 4.2   /         |   627984 (  306.6   623.0)|        0 | 627983 |
 b swap  swap      |   627984 (  306.6   623.0)|  627984 |1255967 |
 c -----           |  2503872 ( 1222.6  2484.0)|        0 |2503871 |    819
 d -----           |                           |         |        |
 e -----           |                           |         |        |
 f -----           |                           |         |        |
 g -----           |                           |         |        |
 h -----           |                           |         |        |
 h 4.2   /usr      |  1246896 (  608.8  1237.0)| 1255968 |2502863 |
 -----------------|--------------------------|---------|--------|-------

     At this point you should define, alter, or delete the file systems.
     BSD/OS requires at least a root filesystem (/), a swap partition and
     a /usr filesystem.

     If you do not understand this screen, press X and then Y to confirm
     the cancel.  When asked if you want the standard setup, answer YES.

     [I]mport [A]dd [D]elete [E]dit [N]ext [X]Abort [?]Help
```

We have now created an "a" partition that spans one quarter of the disk, (to be used for the new user's home directories), a "b" partition that spans a second quarter of the disk, (to be used as secondary swap space), and an "h" partition, which we'll use as a news filesystem. We now press "N" to go to the next screen.

You do not have to load any boot blocks since this is a second disk drive, but we recommend that you do, in case you want to boot from this drive in the future.

Next, after the `disksetup` command has finished, in the case of adding an IDE disk drive, you will need to run the `diskdefect`(8) command. The `diskdefect` program builds a disk defect table on the disk (in two different ways). Note, **only** run this command for IDE disks. Do **not** run this command when adding SCSI disks!

For older IDE disks that **do not have automatic bad-sector replacement**, use this command:

```
 # diskdefect -wsa wd1
```

where "1" is the second disk drive on the system, "2" would be the third, and so on.

For newer IDE disks that **do have automatic bad-sector replacement**, use this command:

```
# diskdefect wd1 0
```

where "1" is the second disk drive on the system, "2" would be the third, and so on.

The `diskdefect` utility will scan the disk for read and write problems and any sectors that fail will be added to the disk defect table. Be patient, this can take a while to complete!

Now that the disk is partitioned, and the disk defect table has been created for IDE disks, we must initialize the filesystem on the new disk partitions. This is done by running the command `newfs`(8) on each new partition *intended for file storage*, e.g., not on swap partitions. In our example, for IDE disks, we'd use the commands:

```
# newfs /dev/rwd1a
# newfs /dev/rwd1h
```

and for SCSI disks, we'd use the commands:

```
# newfs /dev/rsd1a
# newfs /dev/rsd1h
```

where "1" is the second disk drive on the system, "2" would be the third, and so on.

If you have added a partition for additional swap space, you do not need to run `newfs` on that partition. So, in our example, we would not run `newfs` on the "b" partition. It might be desirable to run `newfs` with certain additional options depending on the type of data that is going to be stored in the filesystem. The defaults are always an acceptable choice, but in the case of news filesystems you will be storing a disproportionately large number of small files in the filesystem. The default `newfs` behavior will not maximize storage capacity for this case. For news partitions a good `newfs` command for IDE disks would be:

```
# newfs -f 512 -b 4096 -i 3072 /dev/rwd1h
```

and for SCSI disks would be:

```
# newfs -f 512 -b 4096 -i 3072 /dev/rsd1h
```

where "1" is the second disk drive on the system, "2" would be the third, and so on. This will maximize storage for news' many small files. For information on additional modifications you can make, see the `newfs`(8) manual page.

The partitions are now ready to be mounted to store files.

To mount a partition as a new directory, i.e., one that currently has no data in it, you should simply create the new directory and mount the partition. For the new user home directory created by our example, the commands for IDE disks would be:

```
# mkdir /usr/home1
# mount /dev/wd1a /usr/home1
```

and for SCSI disks would be:

```
  # mkdir /usr/home1
  # mount /dev/sd1a /usr/home1
```

where “1” is the second disk drive on the system, “2” would be the third, and so on. Note, do **not** use the “r” in the device name as was done before!

The `/usr/home1` partition is now ready to store data. Any files stored in subdirectories of `/usr/home1` will be stored on the “a” partition of the new disk drive.

To mount a partition as a new storage location for a directory that already contains data, you will want to copy the data to the new partition, and then remove it from the old partition, before using the new partition. This applies to the “h” partition of the new disk being added in our example, which we intend as a new filesystem for `news`.

The first step is to make sure that your dump tapes are up-to-date! There should be no reason to need them, but when you are moving filesystems from disk to disk there is no reason to take any chance whatsoever of losing your valuable data!

Next, if you are not already running in single user mode, take the system down to single user mode before continuing. This will ensure that you get a consistent snapshot of your information,

Next, mount the new partition in a temporary location. The `/mnt` directory is intended to be used for this purpose. In our example, the command for IDE disks would be:

```
  # mount /dev/wd1h /mnt
```

and for SCSI disks would be:

```
  # mount /dev/sd1h /mnt
```

where “1” is the second disk drive on the system, “2” would be the third, and so on. Note, do **not** use the “r” in the device name as was done before!

Next, use the `pax(1)` command to copy the data from the original directory, `/usr/var/news` in our example, to the new partition currently mounted on `/mnt`.

For both IDE and SCSI disks, the command to use is:

```
  # cd /usr/var/news
  # pax -r -w -p e . /mnt
```

This command will preserve all time stamps and ownership information. The `pax` command might take a while to complete, depending on the size of the directory hierarchy you are copying. Make sure that the `pax` command does not report any errors. If it does report errors, the copy of the information might not be correct, in which case you should **not** proceed further until you have identified and resolved the problem in creating the copy. The most common cause of problems in this step is a mismatch between the size of the new filesystem and the amount of data in the directory hierarchy from which you are copying. You can use the df and du commands to verify that there is sufficient room in the new filesystem to hold the directory hierarchy from which you are copying.

Once the data has been correctly copied to the new disk, we remove all of the data in the old directory:

```
# rm -r /usr/var/news/*
```

The new partition can now be unmounted from /mnt and mounted on the original directory, /usr/var/news. For IDE disks, the commands would be:

```
# umount /dev/wd1h
# mount /dev/wd1h /usr/var/news
```

and for SCSI disks, the commands would be:

```
# umount /dev/sd1h
# mount /dev/sd1h /usr/var/news
```

where "1" is the second disk drive on the system, "2" would be the third, and so on.

To enable swapping on the new swap partition, the "b" partition of the new disk, it is necessary to run the swapon(8) command. In our example, for IDE disks, the command would be:

```
# swapon /dev/wd1b
```

and for SCSI disks, the command would be:

```
# swapon /dev/sd1b
```

where "1" is the second disk drive on the system, "2" would be the third, and so on.

Use the -s option of the pstat(8) command to verify that the device has been added as a swap device.

The final step in adding additional disk drives is to modify the /etc/fstab file to mount the new filesystems and add any new swap devices automatically at each future reboot. You should add a new line for each new filesystem and each new swap partition to the /etc/fstab file. This line will indicate which partitions need to be mounted and which need to be added as additional swap devices. In our example, the following lines would be appended to the /etc/fstab file for IDE disks:

```
/dev/wd1a       /usr/home1      ufs     rw      1 3
/dev/wd1h       /usr/var/news   ufs     rw      1 4
/dev/wd1b       swap            swap    sw      0 0
```

and for SCSI disks:

```
/dev/sd1a       /usr/home1      ufs     rw      1 3
/dev/sd1h       /usr/var/news   ufs     rw      1 4
/dev/sd1b       swap            swap    sw      0 0
```

where "1" is the second disk drive on the system, "2" would be the third, and so on.

For partitions being used as filesystems, the last field of the `/etc/fstab` line should be incremented from those of the previous “ufs” mount types. This indicates the order in which filesystem consistency checks are performed when the system reboots. See the `fstab`(5) and `fsck`(8) manual pages for more information.

Configuring PC Card (PCMCIA) Devices

BSD/OS 4.0 has basic support for PC Card (PCMCIA) devices, found on most notebooks and some desk top machines. This code was contributed to Berkeley Software Design, Inc. by the Wildboar Project and is covered by the following copyright notice:

```
Copyright(c) 1994, 1995, 1996, 1997
Yoichi Shinoda, Yoshitaka Tokugawa, WIDE Project,
Wildboar Project and Foretune.  All rights reserved.
```

Detection of PC Card devices is different than for other devices. If the system detects a supported slot controller during autoconfiguration (normally device `pcic0`), it then reports all of the configurated pcmcia devices. For example:

```
wdc1 at pcmcia0: PCCARD IDE disk controller
```

This does not mean that the devices were actually detected; it means that the system has set up for such a card so that it can be recognized if one is later inserted. These devices contain the word ‘PCCARD’ in the lines printed as the system configures.

Several types of slot controller chips are in use. The kernel on the standard installation floppy supports only Intel and compatible slot controllers (device type `pcic`):

- Intel i82365 and (almost) fully compatibles
- Cirrus Logic CL-PD6710 and CL-PD672x
- IBM KING chip
- Vadem VG468/VG469
- Ricoh RF5C296/396

BSD/OS will automatically recognize many PC Cards based on the description the cards themselves provide. However, some cards either do not behave in a standard way or do not adequately describe themselves (e.g., all Ethernet cards) and must be listed in the `/etc/pccard.conf` configuration file. The `pccard.conf` file maps cards to devices, along with optional per-card flags. This file is read into the kernel by the `csctl`(8) command, which is normally run upon startup via the `/etc/rc.hardware/1.cs.0` script.

BSD/OS normally recognizes a PC Card upon insertion. BSD/OS recognizes cards that were in place at the time that BSD/OS was booted by running the `csconfig`(8) command (as part of the `/etc/rc.hardware/1.cs.0` script). The `csconfig`(8) command can be used to power down and power up cards individually as well. The card in slot 0 is powered up by the command `csconfig 0 up` and powered down by the command `csconfig 0 down`.

When a card is inserted (or powered up) BSD/OS will make a high beeping noise. This will be followed by a low beeping noise if the card was not recognized, or by two descending beeps if the card was recognized.

Laptop computers and PC Cards vary widely, and there are numerous incompatibilities. Some configurations work only when device locations, interrupts, and other parameters are determined experimentally; see `boot`(8) for information on how to set these parameters. BSDI can provide only limited support for problems with these devices.

Special Notes

1. BSD/OS does not currently support any "combo" cards (e.g., combination Ethernet and modem cards).
2. Some ne2000 based Ethernet PC Cards require that they be connected to the Ethernet prior to insertion in order to be recognized.
3. Each time a SCSI adapter is inserted, it will find new devices. For example, if the first time it was inserted it found the disk drives `sd0` and `sd1`, the second time it is inserted it will find `sd2` and `sd3`. Currently there is no way to return to `sd0` and `sd1`, so it is important that added filesystems be unmounted prior to removing a PC Card SCSI adapter.
4. By default, BSD/OS only allows PC Cards to use the I/O address space between 0xd0000 and 0xdffff. This range can be adjusted by using the `-iomem` command in the `/etc/boot.default` file. The available I/O ports and IRQs can also be adjusted in the `/etc/boot.default` file via the `-ioport` and `-irq` commands. See `boot`(8) for more information on `/etc/boot.default.`

Configuring Power Management

BSD/OS 4.0 experimentally supports Advanced Power Management (APM) using the PCMCIA/Wildboar code (see *Configuring PC Cards (PCMCIA) Devices*). APM features include: ability to suspend operation (laptops), ability to learn and report the battery level, and ability to put devices (or even the CPU) into low-power or standby mode.

Of course, your hardware must support APM for any of its features to work. Some notebooks support some features but not others.

To configure APM, the entry `-apm` must be added to the `/etc/boot.default` file. See `boot`(8) for more information on `/etc/boot.default`.

When the system boots, the script `/etc/rc.hardware/0.apm.0` will run the command `apmstart`(8) to enable the Advanced Power Management features of BSD/OS.

The program `apm`(8) can control power management functions. This facility is considered experimental and should be used with care.

Power management BIOS and hardware implementations vary widely. Many laptop computers work properly; on others, some power management functions fail to work. BSDI can provide only limited support for problems with power management.

For machines with an APM bios and hardware which supports it, the `halt`(8) command may be called with the `-h` option to cause the machine to power down after the system is halted. If the BIOS or hardware does not support this feature then the `-h` option has no effect..

Administrator's Notes

Root's Home Directory

The home directory for the superuser, the user named "root", is `/root` instead of the historical `/`.

Console Terminal

The terminal type for the console terminal emulator is `ibmpc3`.

Virtual Consoles and Multiple Text Displays

For those who do not use the X11 Window System, there are several different ways to utilize multiple text displays on the console. The most straightforward is to enable some of the `ttyc*` devices in `/etc/ttys` (which enable "multiple consoles"). This will enable you to type <Alt-*Function-Key*> to switch among different screens (a feature not supported on older monochrome display adapters). Typing <Alt-Function-Key-1> shows you the primary console, <Alt-Function-key-2> gets you ttyc2, etc. The display switch is immediate.

BSD/OS also provides two programs that give you multiple text windows on any "glass-terminal" style display by using pseudo terminals:

- `screen`(1) – screen manager with VT100/ANSI terminal emulation.
- `window`(1) - Window implements a window environment on ASCII terminals.

The `screen` program is more like virtual consoles and `window` lets you see the output from two different sessions at the same time (you are really running a simple windowing system that happens to be text based). The `screen` program is particularly flexible in that you can "detach" a screen and later re-attach it to the same physical screen or to a different one.

You can choose whichever scheme fits your needs or mix and match (e.g., you can run a `screen` or `window` session on each virtual console).

See the `screen`(1), `window`(1), `cons`(4), `pccons`(4) and `ttys`(5) manual pages for more information.

Increasing the Number of Pseudo-Teletypes (ptys)

If you have a large number of network users, you might want to increase the number of pseudo-teletypes (ptys). To increase the number of available pseudo-teletypes from 64 (the default) to 128, execute the following commands (as root) to create the new devices (note, there is a limit of 256 pty devices in almost every system utility).

```
# cd /dev
# ./MAKEDEV pty4 pty5 pty6 pty7
```

The `/etc/ttys` file by default lists all 256 pty devices, so no changes to it are necessary.

See the `pty`(4) and `ttys`(5) manual pages for more information.

Shutting Down the System

Due to the nature of BSD/OS's high-speed, cached filesystems, powering off or rebooting the computer without flushing the cached data to disk can lead to filesystem data loss.

To shut down from multi-user mode, use the `halt`(8), `reboot`(8), or `shutdown`(8) commands. These commands must be run as the superuser. For example, to reboot the system execute:

```
# reboot
```

To halt the system execute:

```
# halt
```

Polite system administrators always use `shutdown -h` when other people are using the system so that the users are warned before the system is unavailable.

See the `halt`(8), `reboot`(8), and `shutdown`(8) manual pages for more information.

/etc/group

This list explains the purpose of the different groups found in the `/etc/group` file:

`bin`	Owns system binaries.
`bsdi`	No special meaning at user sites.
`daemon`	For system daemons and their files.
`dialer`	Outgoing modem access.
`games`	Security for games.
`kmem`	For processes who read /dev/kmem.
`mail`	Owner of mail spool directory.
`maxim`	People who can run MaxIM
`netdial`	Provides security for PPP & SLIP
`news`	For the news system
`nogroup`	Should never have write permission.
`operator`	For those who perform dumps.
`staff`	No special meaning at user sites.
`sys`	Read access to system sources.
`tty`	Programs that access serial lines.
`user`	Default group for users added by `adduser`.
`utmp`	For processes that write the utmp file.
`uucp`	For uucp and other processes that use serial lines.
`wheel`	Permission to su to super-user (root).
`www`	The owner of http server files.

Regularly Scheduled Maintenance

The default `/etc/crontab` file runs the `/etc/daily`, `/etc/weekly`, and `/etc/monthly` shell scripts on a daily, weekly, and monthly basis, respectively. The default scripts perform general maintenance, rotate logs, rotate accounting records, etc. They also invoke the files `/etc/daily.local`, `/etc/weekly.local`, and `/etc/monthly.local` for local procedures. As distributed, the local versions contain some general templates for operations many administrators like to perform, but they must be explicitly enabled for this to occur. The script output is saved in `/var/log/daily`, `/var/log/weekly`, and `/var/log/monthly` respectively, and is sent via email to "root" (which should be automatically forwarded to the system administrator).

Security Levels

BSD/OS 4.0 has the notion of a "current security level". By default, the system goes to a "secure" mode when multi-user. In that mode, `/dev/kmem` and `/dev/mem` cannot be written, raw disks can not be written, and immutable files cannot be written - even by the superuser. See `init`(8) for general information on the security levels. See `chflags`(1) for information on flags including the immutable flags.

The modes are:

- 0 – Insecure mode – immutable and append-only flags can be turned off. All devices can be read or written subject to their permissions.
- 1 – Secure mode – immutable and append-only flags can not be changed; disks for mounted filesystems, `/dev/mem`, and `/dev/kmem` cannot be written.
- 2 – Highly secure mode – same as secure mode, plus disks are always read-only whether mounted or not. This level precludes tampering with filesystems by unmounting them, but also inhibits running `newfs`(8) or writing floppies while the system is multi-user.

You can disable automatic activation of these security features by recompiling the kernel with the `INSECURE` flag set. You will still be able to change security levels manually using `sysctl`(8).

asyncd

The `asyncd`(8) daemon processes asynchronous page writes to mapped files in order to avoid deadlocks in the virtual memory system. It is normally started from `/etc/rc`, which runs two daemon processes by default.

The sh Shell

The POSIX 1003.2 shell specification requires that the Bourne shell `sh`(1) (like the historical `ksh` shell), read and interpret the file named by the ENV environment variable at startup. Many `ksh` users have an ENV variable already set – the BSD/OS `sh` might behave differently than before until the ENV environment variable is unset.

Support for Metamail in BSD/OS Mailers

The `metamail`(1) program displays MIME-format email. MIME provides a multimedia extension to email; MIME messages might contain non-English fonts, images, sounds and more, and they might logically divide a message into a sequence of sub-messages. `Metamail` uses a configurable mailcap file to parse MIME headers and to call appropriate multimedia display programs to display different parts of the message.

Many of the mailers included in BSD/OS 4.0 will automatically feed complex MIME messages to metamail for display. These mailers include `mail`(1), `mh`(1), `xmh`(1) and `elm`. To disable the automatic use of metamail, set the environment variable NOMETAMAIL. To do this using a Bourne-style shell, you would type:

```
  $ NOMETAMAIL=1; export NOMETAMAIL
```

To do this using a csh-style shell, you would type:

```
  % setenv NOMETAMAIL 1
```

`Metamail` is sensitive to several environment variables. The most common customizations are:

- To set the environment variable MM_NOASK to 1, causing `metamail` to not prompt before it executes a program to display part of a message.
- To set the environment variable METAMAIL_PAGER to the name of your favorite pager program (the default is `more`(1)).
- To set the environment variable MM_QUIET to 1, so that `metamail` will suppress messages about the programs it calls to display mail.
- To add a private `$HOME/.mailcap` file containing descriptions of headers that are not listed in the master `/etc/mailcap` file.
- To set the environment variable MM_CHARSET to iso-8859-1, so that the MH `show` command can tell whether it needs to treat ISO 8859-1 message bodies specially. Note that `more`(1) in BSD/OS 4.0 now supports ISO 8859-1 characters.

You can also run `metamail` standalone to display files containing MIME format messages. See `metamail`(1) for more information.

sysctl

The `sysctl`(8) program and the `sysctl`(3) C library interface support a number of new parameters in BSD/OS 4.0, in particular socket- and TCP-related parameters. See the manual pages for more information.

Swap Space

The system can hang if it runs out of swap space. The amount of swap space required depends on the amount of page-out activity, ranging from zero on systems with sufficient memory (RAM) for all needs, up to enough space to contain the data plus stack segments used by all running programs on systems that page heavily.

The BSD/OS 4.0 system will refuse to allow additional memory to be allocated via `sbrk`(2), `malloc`(3), `mmap`(2), or `fork`(2) if the system is low on memory and swap space, failing with the errno and message `ENOMEM: Cannot allocate memory`. Some older programs will dump core rather than printing the ENOMEM error when they run out of memory. This change reduces the likelihood that the system will run out of swap space and hang.

Swap Devices

The BSD/OS 4.0 kernel allows swap devices to be enabled even if they have not been pre-configured in the kernel. Allocations on dynamically-added devices are made only if the pre-configured devices cannot satisfy the request, and allocation on these devices is sequential rather than interleaved. The command `pstat -s` shows the status of each swap device that has been configured and/or enabled.

Bootstrap Changes

The bootstrap program (`/boot`) in BSD/OS 4.0 has several commands:

- The `-help` command lists available commands.
- The `-show` command shows the current values `/boot` will be sending to the kernel.
- Notebook support.
- Support for defining and pre-loading a ramdisk filesystem which can then be booted from.

See `boot`(8) for more details and descriptions of other new commands.

Booting From Floppy

If, for some reason, you need to boot from floppy (e.g., you forgot the root password, hard disk will not boot, filesystems are corrupted, or `/etc/fstab` is incorrect or missing), you can boot off the original installation floppy.

1. Insert Installation floppy
2. Power cycle or reset computer to boot
3. The automatic installation procedure will begin. Proceed far enough through this procedure to specify the location of a BSD/OS 4.0 Binary CD-ROM, either local or remote, or to configure the network if the CD-ROM is remote. Then type <Control-C> to exit the installation process.
4. You are now at a single user prompt. You can `fsck` and `mount` the hard disk filesystems and repair them.

Booting Single User

Sometimes you need to boot to single user mode (e.g., to test a new kernel or if you forgot the root password). To do this, abort the normal system boot by pressing a key when prompted, and then enter `/bsd` (or the device/pathname specification for another kernel binary) at the `Boot:` prompt. The system will then come up in single user mode. See `boot`(8) for additional details.

You should then run:

```
# fsck -p
# mount -a -t nonfs
```

After you have made whatever changes are needed (e.g., `passwd root`), you can come up multiuser by typing <Control-D> to exit the shell. You can also reboot using the `reboot` command.

My Boot Blocks Are Gone!

Problem: When I installed the DOS 5.0a update or DOS 6.0 it erased `bootany`!

No problem. Just boot from the BSD/OS boot floppy as described above, and run `disksetup -B wd0` (or `sd0` for a SCSI disk). This will reinstall the BSD/OS boot blocks.

Pinpointing System Performance Issues

The explosive growth of the Internet in recent years has created a heretofore unprecedented demand for system performance. In many cases, system administrators have been left scrambling, trying to understand the performance issues raised by the loads and new technologies they are facing.

In this chapter you will find suggested methods for analyzing system performance potential, prioritizing the search for bottlenecks, and identifying problem areas.

Introduction

This section discusses methods for tracking performance problems, specific parameters that can be tuned, and data analysis tools that are available.

Starting with a study of the overall architecture, this section covers configuration, kernel tuning, disk subsystem analysis, networks, memory usage, application performance, and network protocol behavior.

The Big Picture

Methodology

A bit of method goes a long way: it saves time; it gets to the source of the problem quickly; and it helps avoid missing the problem. When trying to improve performance, start where the biggest gains can be had. It makes no sense to speed a system by a factor of 1.01 when a speedup of 5x can be gained. In fact, given today's rules of hardware speeds ever-increasing (at a constant cost) over time, it makes little sense to spend time on 1% speedups. On the other hand, mis-tuned systems might run at only 10% of their potential speed. Concentrating on the factors that can gain back the 90% is obviously far more productive than concentrating on 1% gains. Of course, figuring out which part of a system gets back the other 90% is the problem.

Overall Architecture

When a system's overall design is wrong, performance can really suffer. The overall design must be right if a system is to achieve its potential. Be sure that your assumptions about the system's use and potential are clear.

Consider the model used by many Unix systems for network demons. The server forks a process to handle each network request it receives. This works reasonably well when the connections are infrequent or relatively long-lived (at least a handful of interactions). But, when connections are short-lived (a single interaction or two), the cost of the fork() can dominate the cost of handling the connection. This is too often the case for HTTP (the WWW protocol). Over the last few years, the connection rate for a 'busy' web server has gone from dozens of connections per hour to dozens of connections per second. A strategy of "pre-forking" HTTP servers or using a single process and multiplexing requests with select() can result in a 10x-100x improvement in performance [1].

Configuration Errors

In the big picture, most of the easy performance gains arise from system configuration improvements. Most systems come out of the box with a configuration that aims to be a "jack of all trades" and, as it turns out, master of none. This means that you can greatly improve performance by examining and improving items like:

- basic kernel tuning,
- disk layout,
- system memory sizing and allocation,
- hardware selection and configuration,
- configuration of system services (e.g., a local DNS cache), and
- computer center scheduling.

Best case improvements can approach the 100x range.

Application Tuning

Usually, it is difficult to increase application performance by 10x or more. Out of the box, an application is likely to perform acceptably, though it might not be a stellar performer under heavy load. The first rule for applications is to choose your applications carefully and with an eye toward performance.

In most cases, vendor's code is not modifiable or tunable to a great extent. Be sure to take the time to understand how an application uses system resources and to investigate the tuning options that are available. For example:

- Should the application's files be split among several disk drives?
- Can unused features be switched off?
- Is the application logging too much data?

Kernel Tuning

Kernel algorithms must be well-matched to the task at hand. Basic kernel tuning is a fundamental tool in performance improvement. Regrettably, beyond basic tuning, few gains can be had without technological innovation (usually at great cost unless amortized across a very large number of systems).

Developing Expectations For System Performance

It is hard to tune a system without a basic collection of performance reference points to help you analyze the data provided by your monitoring tools. For example: File transfers typically run at between 650 and 750 KB/second on an active Ethernet at your site. Is this good? Could it be better? Or, a mail server you manage is handling 100,000 messages a day and seems to be saturated. Should you expect more? Where do you start looking for problems?

Having an "intuitive" feel for the answers to questions like these makes the process of performance analysis considerably more productive and more fun. Run your performance monitoring tools and get a feel for the numbers you see under various kinds of loads. In many cases, you can generate useful synthetic loads with a few lines of C or a bit of perl code. This is the information that gives you a feel for what to expect from your machines. It will save you from having to individually calculate and analyze each problem you take on.

Take the time to think through some of your system's basic limits. For example, what kind of performance can you expect from TCP/IP running on a 10 Mbit Ethernet?

Starting from the raw speed of 10,000,000 bits per second (note that 10,000,000 is 10 x 1000 x 1000, not 10 x 1024 x 1024), you have to calculate the bandwidth that will actually be available for carrying data (payload). First, calculate the link layer and protocol overheads:

8 bytes	Ethernet preamble and start of frame delimiter
14 bytes	Ethernet header
20 bytes	IP header
20 bytes	TCP header (assuming no options)
4 bytes	Ethernet CRC
12 bytes	Ethernet interframe gap
76 bytes	Total per packet overhead

Then measure (or estimate) the average packet size on the network you are interested in. You can use `netstat`(8) to get a count of the number of packets and bytes sent and received by an interface.

Next, make an estimate of the acknowledgement (ACK) overhead. If the traffic is predominantly in one direction, there will be ACK traffic that will reduce throughput. In the worst case, you would see one ACK for every two packets. Each ACK will consume 76 bytes unless it can be 'piggybacked' with data.

Pulling this information together into a table provides a picture of the performance to expect from TCP/IP on Ethernet.

Average Packet Size (bytes)	Uni-directional Traffic (KB/sec)	Bi-directional Traffic (KB/sec)
64	439	558
128	646	766
256	845	941
512	998	1063
1024	1098	1136
1500	1134	1162

Similar tables could be produced for UDP or any other protocol of interest.

The above numbers are an upper bound. A system might get close to these numbers during bulk transfers with little competing traffic. In an environment with several busy machines on the same wire, throughput will be somewhat lower, perhaps 70-90% of these numbers.

With these calculations in hand, you can try some experiments to see what happens in your environment. The utility `rcp(1)` offers an easy way to do a quick experiment. Contrary to what many folks think, `ftp(1)` is usually not the best test of raw network capacity. Many ftp implementations use small buffers and are actually poor performers. Copy a big file (at least a megabyte) from one idle machine to another and keep track of the real-time spent. Do this two or three times to get an idea of your network's consistency. This experiment should reveal approximately the maximum throughput for your system as configured.

Try the experiment again on a loaded network. Note how badly (or not) the performance degrades in your environment.

It is worth doing this sort of analysis and experimentation for any subsystem you are interested in really understanding. Aaron Brown and Margo Seltzer presented a very interesting paper reporting their findings on the performance of Intel hardware at the 1997 USENIX conference [2].

Protocol Performance

Analysis and experimentation assist in understanding performance. Hard facts and knowledge about underlying algorithms can also contribute. Here are some interesting facts that contribute to overall performance of various network protocols:

- TCP/IP packets tend to be small, the average (according to one router vendor) is around 300 bytes.
- When a user's packets are being dropped (due to network load), network performance will be noticeably bad for that user (i.e., 20 second or more response time or 300 b/s FTP throughput). This is because TCP backs off exponentially when retransmitting lost packets. This is good for the network as a whole, but bad for the affected connection. Good performance requires very low packet loss rates.
- HTTP and SMTP connections tend toward a small number of interactions that move a small or medium amount of data. For example, consider a WWW page that sets up and tears down 14 connections to load 14 small pictures (e.g., bullets). This interaction pattern means that the TCP 'slow start algorithm' (which initially responds to requests a bit slower than possible to ensure that the network is not overwhelmed) will probably keep performance lower than is theoretically possible.
- New features often break things. While the design of TCP/IP supports adding new features without breaking existing implementations, it doesn't always work that way. It is worth keeping track of what's new on your network (and beyond), although the new code is not always where the bug lies. A common example of this phenomenon occurred a few years ago when TCP connections would fail when RFC1323 options were used. The problem was that some versions of Linux failed to handle the options correctly and corrupted packets. Fortunately, the folks implementing the support for RFC1323 had the forethought to implement an additional

mechanism for turning them off on the fly. Another example concerned some Microsoft PPP implementations that would negotiate larger packet sizes than they could handle.

Understanding the Application

To improve overall performance, including applications, you're going to need to dig into the application and understand how it works and what it needs from the system.

Sadly, the application's documentation is usually not where to start. Tools like `ktrace`(1), `ps`(1), `top`(1), `netstat`(8), and `tcpdump`(1) let you see how the application is interacting with your system and will yield real clues to the application's behavior.

Reference Points

As you build up a base of experience, you also build a set of reference points for what to expect from a system in real environments. These expectations can be really useful when trying to make initial "off-the-cuff" evaluations of system performance. For example, a good operating system can enable these levels of performance:

- A 90 MHz Pentium with 80 MB of RAM buffer cache can feed more than 3 million news articles (over 9GB) per day.
- A 150 MHz Pentium Pro can deliver over 1 million mail messages per day.
- A 133 MHz Pentium can serve static HTML from a cache at T3 speeds.
- A 150 MHz Pentium Pro can handle more than 3 million direct web hits per day, transferring 27 GB of content.
- A 133 MHz Pentium can only handle about 60 HTTP requests per second with a forking proxy.
- A 150 MHz Pentium running screend (user space packet filter) won't be able to handle a T1.
- A 266 MHz Pentium-II cannot quite route a 100 Mbit Ethernet running at full load (full routing table, 256 byte packets). It can handle a T3.

These reference points are not complete; they assume that you have the OS, hardware configuration, RAM, and other support (these reflect observations using BSD/OS, DEC or Intel Fast Ethernet Adapters, adequate memory and SCSI disks). Collect a set of reference points that are meaningful to you based on the systems that you are working with.

Collecting Data

Here are some hints for collecting performance data.

- **Get baseline data.** Without it you are lost, you won't be able to tell how much good (or bad) you have done.
- **Make sure you have enough.** For example, if you're trying to track down UDP packets "dropped due to no socket," make sure that your tcpdump has run for long enough that you expect to have collected several of these packets (`uptime`(1) and 'netstat -s' will help you pick an interval).
- **Make sure the data are representative.** Continuing with the above

example, you also need to consider the possibility that the traffic you are interested in is not evenly distributed, and make sure that the data you collected represents an interval during which you could reasonably expect interesting traffic. To do this, you could start using `cron`(8) and `netstat -s` to sample every hour to determine when the traffic you are interested in occurs.

- **Make sure you sample frequently enough.** When you're looking for things like loading trends, you need to sample often enough to actually catch the event you're interested in. As a rule of thumb, you want to sample at at least twice the frequency of the event you are trying to catch. If you don't know the frequency of the event, initially you may need to use a significantly higher sampling rate.
- **Automate your initial data reduction.** Tools like `tcpdump(1)` can generate huge amounts of data. Start by using tools like perl (and in the case of tcpdump, tcpdump's built-in filtering facility) to eliminate extraneous data. When you find no interesting data, don't forget to check your automated tools.
- **Check the log files.** Don't forget to check your system's log files and any logs files kept by the applications for clues.

Tuning the Subsystems

It is easier to look at the individual subsystems in relative isolation from each other.

The Disk Subsystem

Disk subsystem performance is often the single biggest performance problem on Unix systems. If your disks are busy at all (web, mail and news servers all qualify), this is the place to start.

Since the virtual memory system uses the disk when it is out of RAM, start by checking with `vmstat`(8) to ensure your system is not paging (the pages out 'po' column should be zero most of the time; it's OK to page in programs occasionally). If your system is paging, then skip the disk tuning for now and start by looking at RAM availability.

If your application has significant disk I/O requirements, the disk is almost certain to be your bottleneck. On a busy news or web server, correct disk configuration can easily improve performance by an order of magnitude or more.

Disk subsystem optimization has two main goals:

- minimize seek time, and
- match disk throughput to your system's demands.

When thinking about disk throughput, keep your system's load mix in mind. For some applications (e.g., streaming video), your system should maximize the rate at which data flows from the disk to the application. Rotational speed (the limiting factor on transfer rate) will likely dominate all other considerations. Other applications (e.g., mail and news servers) will see file creation and deletion operations dominate performance. File creation and deletion, which use synchronous operations to ensure data integrity, cause seek time to be the dominant factor in disk performance.

While modern SCSI disks are capable of transfer rates that approach or even exceed the speed of the SCSI bus (e.g., 10 MB/sec for Fast SCSI), the limiting factor for Unix systems is typically the number of file system operations that can be performed in a unit of time. For example, on a machine functioning as a mail relay, the limiting factor is going to be the speed of file creation and deletion. See Table 1 for a listing of SCSI speeds.

Table 1 - SCSI Bus Bandwidth

SCSI Version	MB/s
SCSI-I (async)	3.5
SCSI-I (sync)	5.0
Fast SCSI	10.0
Fast Wide SCSI	20.0
Ultra SCSI	20.0
Ultra Wide SCSI	40.0

In both cases, the only ways to increase performance (once you've hit the limit of the disk's physical characteristics) are:

- add disk spindles to increase parallelism,
- add hardware like PrestoServe to decrease seeks,
- identify and purchase a new file system technology that reduces seeks (test it carefully!)

In some cases it will be necessary to make changes to the application to make it aware of multiple directories or to use hardware that will transparently distribute the load across multiple spindles (e.g., RAID).

Disk Performance Factors

SCSI vs. IDE: For machines with heavy disk and/or CPU loads, SCSI is superior to IDE. A single system generally supports far larger numbers of SCSI disks than IDE; this can also be a consideration. With good host adapters, SCSI driver overhead is lower and 'disconnect' (the ability to issue a command to a drive and then 'disconnect' to issue a command to a different drive) is a big win. For machines that need only one or two disks and that have CPU cycles to spare, the lower cost of IDE is attractive.

Drive RPM: Rotational speed determines how fast the data moves under the heads, which places the upper bound on transfer speed. Today's really fast drives spin at 10,033 RPM (Seagate Cheetahs) and deliver about 15 MB/sec on the outside tracks. Last year's fast drives spun at 7200 RPM; inexpensive drives in use today spin in the 3600 to 5400 RPM range.

Average seek time: The average interval for the heads to move from one cylinder to another. Typically, this is quoted as about half of the maximum (inside track to outside track) seek time. Lower average seek times are very desirable. Currently, a 7-8 ms average seek time is really good, 8-10 ms is typical and much more than 10 ms starts to impact loaded system performance. Unix-like systems perform many seeks between the inode blocks and actual file data. To ensure file system integrity, writes of significant inode data are not cached, so these seeks are a significant part of the expense of operating a disk. In general, if a tradeoff must be made, a lower average seek time is better than a higher rotational speed.

The price of fast seek times and high RPMs is increased heat generation; take care to keep them cool or you will sacrifice performance for reliability. Use plenty of fans and make sure that there is good air flow through the box, or put the disks in their own enclosure(s).

On drive caches allow the drive to buffer data and therefore handle bursts of data in excess of media speed. Most modern drives have both read and write caches that can be enabled separately. The read cache speeds read operations by remembering the data passing under the heads before it is requested. A write cache, on the other hand, can cause problems if the drive claims that the data is on the disk before it is actually committed to the physical media (although some drives claim to be able to get the data to physical media even if power is lost while the data is still in the cache). Understand your hardware fully before enabling such a feature.

SCSI Tagged Command Queueing enables multiple commands to be issued serially to a single drive. Tagged command queueing increases disk drive performance in two ways: it enables the drive to optimize some mixes of commands and overlap command decoding with command processing.

Avoiding Seeks

Some of best ways to improve performance involve reducing the number of operations to be performed. Since seek operations are performance eaters and relatively common, avoiding seeks can improve performance. Here are some methods to reduce the impact of disk seeks.

Use Multiple Spindles

Systems that concurrently access multiple files can improve their performance by putting those files on separate disk drives. This avoids the expense of seeks between files. Any busy server that maintains logs would do well to separate the log files from the server's data.

If you must store several Unix-style partitions on a large disk, try to put the most frequently used partition (`swap` or `/usr`) at the middle of the disk and the less frequently used ones (like home directories) at the outside edges.

Turn Off Access Time Updating

If maintaining a file's most recent access time (i.e., the time someone read it – not the time someone wrote it) is not critical to your site and, additionally, your data is predominantly read-only, save many seeks by disabling access time updating on the file system holding the data. This particularly benefits news servers. To turn off access time updates on BSD/OS, set the "noaccesstime" flag in the `/etc/fstab` file for the file system in question:

```
/dev/sp0a /var/news/spool ufs rw,noaccesstime 0
```

The same functionality exists in other Unix variants, but the names have been changed to confuse the innocent.

Use RAM Disk For /tmp

Many programs use the `/tmp` directory for temporary files and can benefit greatly from a faster `/tmp` implementation. On systems with adequate RAM, using 'RAM disk' or other in-memory file system can dramatically increase throughput. BSD/OS and other 4.4BSD derivatives support MFS, a memory file system that uses the swap device as backing store for data stored in memory. The following command creates a 16 MB MFS file system for `/tmp`:

```
mount -t mfs -o rw -s=32768 /dev/sd0b /tmp
```

Or you could put this line in `/etc/fstab`:

```
/dev/sd0b /tmp mfs rw,-s=32768 0 0
```

The data stored on a MFS `/tmp` is lost if `/tmp` is unmounted or if the system crashes. Since `/tmp` is often cleared at reboot even when the file system is on disk, this does not seem to represent a significant problem.

Increasing Disk and SCSI Channel Throughput

Size The Buffer Cache Correctly

The best solution for disk bottleneck is to avoid disk operations altogether. The in-memory buffer cache tries to do this. On most 4.4BSD-derived systems, the buffer cache is allocated as a portion of the available RAM. By default, BSD/OS uses 10% of RAM. For some sites, this is not enough and should be increased. However, increasing the size of the buffer cache decreases the amount of memory available for user processes. If you increase the size of the buffer cache enough to force paging, all will be for naught.

Few systems support tools to report buffer cache utilization.

Use Multiple Disk Controllers

Adding extra disk controllers can increase throughput by allowing transfers to occur in parallel. Multiple disks on a single controller can cause bus contention, and thus lower performance, when both disks are ready to transfer data.

Striping (RAID 0)

RAID 0 distributes the contents of a file system among multiple drives. The result (when properly configured) is an increase in the bandwidth to the file system. The downside is that increasing the number of drives and controllers used to support a file system reduces the MTBF; RAID 0 does nothing to defend against hardware failures.

Higher Levels of RAID

RAID 1 (mirroring) provides the same write performance as RAID 0 at the expense of redundant disk drives (and, hence, higher cost per storage unit). Where performance is a major constraint and file system activity consists of many small writes (e-mail, news), RAID 1 may be the way to go. Of course, the redundancy can dramatically increase mean time to data loss.

Where read access dominates a file system's activity, RAID levels 3 (dedicated parity disk) and 5 (rotating parity disk) offer reliability and performance that is not significantly worse than RAID 0. RAID 3 and 5 offer reasonable write performance on large (relative to the stripe) files.

At the very high end, RAID hardware is implemented with a high degree of parallelism and sustained throughput can approach, or exceed, that of the system bus. Note, however, that write performance (particularly for smaller blocks) can be poor.

Monitoring and Tuning Disk Performance

The `iostat`(8) command can provide some help with optimizing disk performance. For each drive, it reports three statistics: sps (sectors transferred per second), tps (transfers per second), and msps (milliseconds per "seek," including rotational latency in addition to the time to position the heads). The accuracy of these numbers, especially msps, varies considerably among implementations.

The `tunefs`(8) command is used to modify the dynamic parameters of a file system. Disks whose contents are primarily read and that exploit read caching will often benefit from setting the rotational delay parameter to zero:

```
# tunefs -d 0 file system
```

When adjusting file system parameters with `tunefs`(8), remember that only new contents are affected by the change. Since the existing contents of the file system may constrain the allocation algorithms, it is best to use a scratch file system while experimenting with `tunefs`(8) options.

RAM

Two major performance factors related to RAM are:

- Avoiding paging (make sure that there is enough memory in the system to handle your load)
- Ensuring that enough memory has been allocated to the kernel.

Sizing Main Memory

If your system is short on main memory, it will end up swapping and the speed of your memory subsystem will degrade to the speed of your disk (usually about three orders of magnitude difference). The most useful tools for diagnosing paging problems are `vmstat`(8) and `pstat`(8). The 'po' column of `vmstat`(8) shows you 'page out' activity. Any value other than 0 on anything more than an occasional basis means that performance is suffering.

As you increase the amount of main memory, you are also increasing the likelihood of memory errors. You should plan on running Error Correcting Code (ECC) memory if your hardware supports it (and you should only be buying hardware that does!). Current ECC implementations on PC hardware cost about 5% in memory bandwidth when accessing main memory (and the CPU tends to go to main memory less frequently than one might expect).

When sizing a system's RAM, it is often useful to know the memory usage of a typical user or a particular process. On BSD derivatives, you can get some of this information with 'ps -avx.'

The RSS ('resident set size') column reports the number of kilobytes in RAM for a process. Since the shared libraries and the text are common to all of the instances of a program (and in the case of the shared libraries to all of the users of the library), you cannot use this information to determine the amount of memory that would be consumed by an additional instance of the program. It does give an upper bound for making estimates.

The TSZ ('text size') column reports the size of the program's text, but it does not tell you how much of the text is resident (i.e., how many pages are not in the swap area).

After observing these parameters for a while, you can get a pretty good idea of how much memory a particular process typically uses. It may also be worth watching the numbers while you subject a process to both typical and extraordinary loads. Using this information, you can then estimate how much memory you would need to support a particular mix of applications. Perl and cron can help to automate this task.

Cache Size

Currently, tools don't exist to directly monitor real life cache performance. Within reason, more cache is better. Run real-life benchmarks to see if different hardware can improve your site's performance.

Kernel Memory Tuning

Most modern kernels can be tuned for increased loads using only a "knob" or two, but for some applications you will need to do additional tuning to get peak performance.

On BSD/OS, the size of kernel memory has a default upper bound of 248 MB. Major components of kernel memory are the buffer cache (BUFMEM) and the mbufclusters (NMBCLUSTERS). Both are in addition to the memory allocated by KMEMSIZE. On most systems, these memory allocations are completely adequate, but if the size of the buffer cache is increased beyond 128 MB it may get tight.

The limit on kernel memory can be increased by modifying the include file `machine/vmlayout.h` (with suitable knowledge of the processor architecture in question) and then recompiling the entire kernel. In addition to the kernel, you'll also need to recompile `libkvm` (both the shared and static versions), `gdb`, `ps` and probably a few other programs.

maxusers

On BSD systems of recent vintage, the place to start kernel memory tuning is with the 'maxusers' configuration parameter. The maxusers parameter is the "knob" by which the kernel resources are scaled to differing loads. Don't think of this in terms of actual numbers of humans, but just as a knob that can request "more" or "less."

As of BSD/OS 4.0, setting `maxusers=0` causes it to adjust to the amount of memory actually installed on the system. Not specifying a value for `maxusers` will force a default value of zero to be used.

System default for maxusers

With maxusers set to a value of zero, the system automatically chooses a value based on how much RAM is installed:

- If you have 26MB or less of RAM, maxusers is set to 10.
- If you have more than 144MB of RAM, maxusers is set to 128.
- Otherwise, maxusers is set to (# MB - 16).

Or in other words, the number of MB's of RAM minus 16. If the result is less than 10, it will be set to 10. If greater than 128, it will be set to 128. (9 < maxusers < 129)

For example, a 64MB machine will effectively set maxusers to 48 (64 - 16). Unless you configure a special value, the maxusers parameter will change whenever you change the amount of installed RAM. If you upgrade from 64MB to 128MB, maxusers will then automatically change from 48 to 112.

Setting your own value

As a rough rule of thumb, you can start by setting maxusers to the size of main memory (e.g., for a system with 64 MB of main memory start by setting maxusers to 64).

maxbufmem

The kernel variable "maxbufmem" is used to size the buffer cache in systems without a unified user space/buffer cache. A zero value means "use the default amount (10% of RAM)." Otherwise, it is set to the number of bytes of memory to allocate to the cache. You can set this at compile time:

```
  options "BUFMEM=\(32*1024*1024\)"
```

Or you can patch the kernel image (with `bpatch(1)` or a similar tool):

```
  # bpatch -l maxbufmem 33554432
```

and reboot. It is not safe to change the size of the buffer cache in a running system.

A system running a very busy news or web server is an obvious candidate for increasing the size of the buffer cache.

When increasing the size of the buffer cache, the memory available for user processes is decreased. Ideally, a tool would report buffer cache utilization. Unfortunately, such a tool doesn't seem to exist, so tuning is sort of hit-or-miss – increase the size of the buffer cache until you start paging, then back off.

mbufs and NMBCLUSTERS

In the BSD networking implementation, network memory is allocated in mbufs and mbuf clusters. An mbuf is small (128 bytes). When an object requires more than a couple of mbufs, it is stored in an mbuf cluster referred to by the mbuf. The size of an mbuf cluster (MCLBYTES) can vary by processor architecture; on Intel processors, it is 2048 bytes.

The number of mbufs in the system is controlled by their type allocation limit (reported by `vmstat -m`). The configuration option NMBCLUSTERS is used to set the number of mbuf clusters allocated for the network. The default value for NMBCLUSTERS in the kernel is 256. If you have GATEWAY enabled, NMBCLUSTERS is increased to 512. Systems that are network I/O intensive, such as web servers, might want to increase this to 2048.

On recent BSD systems this parameter can be dynamically tuned with `sysctl(8)`:

```
  # sysctl -w net.socket.nmbclusters=512
```

The upper bound on the number of mbuf clusters is set by MAXMBCLUSTERS. Usually MAXMBCLUSTERS is set to 0 and the limit is calculated dynamically at boot time.

If a BSD kernel runs out of mbuf clusters, the kernel will log a message ("mb_map full, NMBCLUSTERS (%d) too small?") and resort to using mbufs. This can also be seen by an increase in the number of mbufs reported by vmstat -m.

KMEMSIZE

The size of kernel memory (aside from the buffer cache and mbuf clusters) is set by KMEMSIZE. Normally, it is scaled from "maxusers" and the amount of memory on the system, but it can also be set directly.

The default KMEMSIZE is 2 MB. At a maxusers value of 64 (or if there is more than 64 MB of memory on the system), KMEMSIZE will increase to 4MB. At a maxusers of 128, KMEMSIZE will increase to 8MB. Beyond that, use the KMEMSIZE option (in the kernel configuration file) to increase the kernel size.

Routing Tables

To run a full Internet routing table in the Spring of 1997, it is necessary to increase the amount of kernel memory to at least 16 MB. The BSD/OS config file for the GENERIC has notes on this:

```
# support for large routing tables,
# e.g., gated with full Internet
# routing:
# options "KMEMSIZE=\(16*1024*1024\)"
# options "DFLDSIZ=\(32*1024*1024\)"
# options "DFLSSIZ=\(4*1024*1024\)"
```

Since the gated process is also likely to get quite large, it also makes sense to increase the default process data and stack sizes.

The `vmstat`(8) can give an overwhelming amount of information about kernel memory resources. Some points worth noting: In the "Memory statistics by type" section, the "Type Limit" and "Kern Limit" columns report the number of times the kernel has hit a limit on memory consumption. Entries that are non-zero bear some investigation. These statistics are collected since system boot, so you'll probably want to look at the difference between two runs that bracket an interesting load. Many of these limits scale with maxusers, so a quick experiment can be done by recompiling the kernel with a higher maxusers value.

Network Performance

For network performance analysis, `netstat`(8) is the command you want. The raw output of a single `netstat`(8) run is not terribly useful, since its counters are initialized at boot time. Save the output in a file and use `diff(1)` or some fancy perl script to show you the changes over an interval.

For understanding why your network performance is sub-par, `tcpdump(1)` or some other sniffer/protocol analyzer is the tool of choice. Data collected with `tcpdump(1)` can be massaged with `perl` (or `sed` and `awk`) to make it easier to see what is going on.

Too Much Traffic

Take time to check that you've only got the traffic you expect on the network. Look for things like DNS traffic (if you have much, configure a local cache), miscellaneous daemons you don't need (not only are they costing you network traffic, they are probably costing you context switches), etc. If you're really trying to squeeze the last little bit out of the wire, consider using IP multicast for things like time synchronization. Use `tcpdump(1)` to see what kind of traffic there is. Watch for unexpected activity, or unexpected amounts of activity, on your hubs. Compare interface statistics with similar machines.

Too Little Bandwidth

With Ethernet, as the amount of traffic on the network increases, the instantaneous availability of the network decreases as the hardware experiences collisions. Use `netstat -is` to monitor the number of collisions as a percentage of the traffic on the wire. Ethernet switches can reduce the number of hardware collisions and increase apparent number of segments. Ethernet adapters that support full duplex when talking to an Ethernet switch can also provide a performance increase.

Errors

Significant error rates (anything more than a fraction of a percent) are often an indication of hardware problems (a bad cable, a failing interface, a missing terminator, etc.).

Use `netstat -is` to monitor the error rate.

Timeouts and Retransmissions

Don't forget to watch the activity lights on your interfaces, hubs, and switches. Look for unexpected traffic, pauses, etc. and then get busy with `tcpdump(1)` to see what is going on. This section only covers some very basic concepts; for more detail, see Richard Steven's excellent book, *TCP/IP Illustrated, Volume 1* [4].

Buffer Sizes

Network buffer sizes can have a significant impact on performance. With FDDI, for example, it is often necessary to increase the TCP send and receive space in order to get acceptable performance.

On BSD/OS and other 4.4 BSD derivatives this can be done with `sysctl`(8):

```
# sysctl -w net.inet.tcp.sendspace = 24576
# sysctl -w net.inet.tcp.recvspace = 24576
```

To a first approximation, think of net.inet.tcp.recvspace as controlling the size of the window advertised by TCP. The net.inet.tcp.sendspace variable controls the amount of data that the kernel will buffer for a user process.

As a general rule of thumb, start sizing the kernel buffers to provide room for five or six packets "in flight" (one packet for the sender to be working on, one packet on the wire, one packet being processed at the receiver – then multiply by two to account for the returning acknowledgements). Since Ethernet packets are 1500 bytes (or less), 8 KB of buffer space is about right. For FDDI, with 4 KB packets, 20 to 24 KB is likely to be necessary to get FDDI performance to live up to its potential.

Another way to think about this is in terms of a "bandwidth delay product:" multiply bandwidth by the round trip time (RTT) for a packet and specify buffering for that much data. If you want to be able to maintain performance in the face of occasional lost packets, figure the bandwidth delay product as:

buffer bytes = bandwidth bytes/second * (1 + N packets) * RTT seconds

Where N is the number of lost packets you want to be able to sustain without losing performance. This is a bit harder to calculate as you have to be able to measure RTT.

Increasing the size of the kernel send buffers can improve the performance of applications that write to the network, but it can also deplete kernel resources when the data is delivered slowly (since it is being buffered by the kernel instead of the application).

The size of application buffers should also be considered. If the buffers are too small, considerable additional system call overhead can be incurred.

Using netstat(8)

netstat -is

Using `netstat -is` yields a basic report on the amount of traffic and the number of errors and collisions seen by a network interface. Here are some of the statistics reported:

- **Input Errors** tells you that a bad packet was received – something is wrong somewhere between you and the sender.
- **Output Errors** means something is wrong with the local hardware (interface card, cables, etc.). If your hardware supports full duplex operation, another possibility is that one end is configured for full duplex while the other is not.
- **Collisions** tells you how busy the network segment you're attached to is. Lots of collisions (say more than 10%) on a regular basis tell you it's time to think about reworking your network architecture.

Another use of `netstat -is` is to track down lost packets. Suppose you are trying to ping a remote machine and are not receiving any replies. You can track down the location of the problem by observing the input and output packet counts of the machines along the way. Each machine that processes the packet will increment the appropriate counters. On the machine where the packet was "lost" you would expect to see an increasing error count, either on input or output.

netstat -isv

Specifying `netstat -isv` yields more detail on the traffic seen by the interface, including the number of dropped packets, the number of octets (bytes) sent and received (which, along with the packet count, enables you to compute average packet size) and data on multi-cast traffic.

netstat -s

With `netstat -s`, you get voluminous statistics on the whole networking subsystem. These statistics are maintained over the life of the system. To see current trends, look at a pair of reports and compare the numbers. A quick and dirty way to do this is to save the output into a pair of files and use `diff`. Of particular interest are the counts of:

- **Dropped packets** indicates that the local machine is receiving network traffic but does not have the resources to handle that traffic. The two most common causes are: running out of mbufs or running out of CPU power. Another possibility is that you've run out of mbuf clusters and the kernel is substituting mbufs, but the system is marginal on CPU power and the extra processing involved has pushed it over the edge.
- **Fast Retransmits** The TCP spec requires that an immediate ACK be sent when a packet is received out of order. This ACK will duplicate an ACK that has already been sent for the last in-order data that the receiver received. When the sender has seen three duplicate ACKs, it assumes that the next packet has been lost and it will retransmit the packet without waiting for a retransmit timeout. This is a "fast retransmit", and, if the receiver's window is big enough, it will prevent TCP from waiting for a timeout when a packet is lost.
- **Duplicates** Duplicate packets are packets that arrived twice. Usually this means that an acknowledgement was delayed and the sender retransmitted.
- **Retransmit Timeouts** The retransmit timer expired while we were waiting for an ACK.
- **IP Bad Header Checksums and TCP Discarded For Bad Checksums** Packets that were "damaged in transit" should be thrown out by the link layer checksum and not seen by any of the higher layers. This is a possible sign that an intermediate gateway is corrupting packets. If you are seeing significant numbers of these, it might be worth the trouble to try to track down the source. To do this, you will need to probe methodically, starting with the nearest gateways (routers) and work outward.
- **UDP with no checksum** These are interesting because running UDP without checksums is asking for trouble. If you are seeing many of these, it would be worth tracking down the source.
- **UDP with no socket** means that you're getting UDP traffic, but nobody is listening for it. At the very least, this is a waste of resources, and it may be indicative of a configuration error or perhaps even something malicious.

Track the source using if many packets show up here.

When the network is performing well, all of the above statistics should represent a small fraction of the traffic that the system sees (well under a fraction of a percent).

netstat -a

With `netstat -a`, you get information on all of the active network connections on the system. Things of interest include:

- **The number of connections** This gives you a feel for the amount of network traffic the system is seeing.
- **Connections with with data in the Send-Q or Recv-Q** You will see this occasionally, but significant numbers of connections with queued data would indicate that something is amiss.
- **Large numbers of connections in states other than ESTABLISHED** On a busy web server, many connections will be in the TIME_WAIT state. But, observing significant numbers of connections in SYN_SENT (indicative of a SYN-flood attack) or FIN_WAIT_2 (waiting for a FIN from the other side after we've sent our FIN, sometimes this indicates a kernel bug) bear investigation.

netstat -m

With `netstat -m`, you get a report on the memory usage of the networking subsystem. When investigating performance problems, the counts of "requests for memory denied/delayed" and of "calls to the protocol drain routines" are of particular interest. They are indications that the network is running low (or out) of memory.

Much of the `netstat` information can be summarized on one screen using the `-netstat` option to *systat*1. The `-netstat` display has been redesigned for BSD/OS 4.0 and contains information found in both the `netstat -s` and `netstat -is` displays. The amount of information may depend on the size of the terminal (or `xterm`) that `systat` is running in - the larger the better.

Using tcpdump(1)

The `tcpdump(1)` program is a powerful tool for analyzing network behavior and is available on most Unix-like systems. With `tcpdump(1)`, you can watch the network traffic between two machines. The packet contents are printed out in a semi-readable form (which is easy to massage with `perl` to help you see the info you're interested in). Each packet is printed, along with a time stamp, on a separate line.

One of the most useful features of tcpdump is the ability to filter traffic. For example, if you're curious about stray traffic on the network with your web servers, you could run something like this:

```
# tcpdump -i de0 not port http
```

If you had connected to the machine by telnet you might use:

```
# tcpdump -i de0 not port http \
    and not port telnet
```

The output from tcpdump would only show non-http and non-telnet traffic, thus enabling you to focus on the traffic that you're interested in.

You can save the data for later analysis by having tcpdump write it to a file:

```
# tcpdump -i de0 -w tcpdump.out
```

When you examine the data later, you'd use:

```
# tcpdump -r tcpdump.out not port http and not port telnet
```

Aside from making analysis of the data more flexible, this approach also consumes fewer machine resources, so you're less likely to drop packets or otherwise impact the system under test. By default, `tcpdump` only grabs the first 76 bytes of the packet (which is enough to get the headers, but you won't get much of the packet content). You can add `'-s 1600'` to save the whole packet.

One of the most common things that you'll be looking for is an explanation for why the network is occasionally pausing.

Here is a recipe for examining `tcpdump` output:

1. Collect the data, and post-process it if you want (for example, you may want to convert the time stamps into relative intervals).
2. Browse through the tcpdump output with your favorite editor, looking for places where the time-stamp jumps by more than the "usual" amount.
3. You're probably in the vicinity of a dropped packet, and the time-stamp is due to the timeout before the packet was retransmitted.
4. Take a quick look around and then start looking for the next time-stamp jump. What you're looking for is a pattern. If the network behavior was bad enough to get you looking at packet traces there is almost certain to be one.
5. After some looking, a pattern will emerge. If you don't know what the problem is by now, you'll need to find somebody who knows more about the protocol in question. But, armed with a description of the problem, the `tcpdump` output, and the patterns you found, it should be easier to get some help.
6. As you're looking at the jumps in the tcpdump output, an important part of the story is the size of the jumps. Are they the same? Are they increasing? There are people who know this stuff well enough to make a good guess as to the source of the problem just by the size of the delay.

In some cases, you will need to arrange for the machine running `tcpdump(1)` to be in the middle of the wire between the machines that are having problems. With some switches this is not possible, and you will either need to substitute a hub (you can also just add a hub at one of the switch ports and connect your monitor and one of the machines under test to the hub) or run tcpdump on the machines at both ends of the link and analyze both sets of data. The hub trick doesn't work if you're trying to look at a link that is running full duplex.

CPU

In performance analysis, you're usually up against one of two things: either CPU utilization is too high or else it's too low. Too high means that you've got to figure out a way to free up more cycles in order to get more work done. Too low means that you've got to figure out what you're waiting on, usually an I/O device.

Start CPU analysis with a tool that will tell you how your CPU is being utilized – `top`(1), `vmstat`(8), and `iostat`(8) will tell you how much of your CPU is being used and whether the cycles are being spent in user or kernel (system) code. If most of the CPU's time is being spent executing user code, `top(1)` and `ps(1)` will help you identify the processes of interest. If your CPU is close to 100% utilization, you need to figure out what it's doing. Start by identifying the process or processes that are using most of the user time.

Even if the CPU is spending most of its time executing in the kernel (high percentage of system time), these processes are likely to be the cause. Once the processes in question have been identified, `ktrace(1)` can be a very useful tool for figuring out what is going on. So are the 'in' (interrupts), 'sy' (system calls) and 'cs' (context switches) columns in the output of `vmstat`(8). You should be developing a rough idea of the performance bounds of the systems you support. This is the information you will need to read the output of commands like `vmstat`(8).

Very simple benchmarks (about 10 lines of code) can help you calibrate your expectations of the machine's performance. For example: A 200 MHz P6 can perform about 500 fork/exit/wait (i.e., null fork with two context switches) operations per second. It can also perform about 400,000 getpid() operations per second (getpid() is arguably the most trivial system call). From these benchmarks, you can make an educated guess that a process that is fork()ing tens of times per second should not be putting too much of a load on the system due to the forks, but a process that is doing hundreds of fork()s per second may well benefit from a redesign.

The `ktrace(1)` program can give insight into the behavior of the processes at the system call level. Since system calls are relatively expensive, this can be very useful. Often the information revealed by `ktrace` will be a surprise, revealing details of the program's operation that were neither documented nor intuitive.

For example, some (maybe all) versions of the Apache web server search from the root for a `.htaccess` file. If you're not using these files, turning off the check can get you a 5 to 10% increase in performance. If you are using them, you can decide if it would be worth the trouble to use a "shallower" directory hierarchy. This is also a great way to figure out error messages (like "file not found" with no file name or without a full path).

Other things to look for are frequent read() and write() calls or such calls with small byte counts, frequent system calls of any type (perhaps the application can cache the data), etc.

If all else fails, you may have to resort to profiling your application or your kernel. Typically this means that you have to either have source code or vendor cooperation. Enabling profiling usually means you need to recompile. You can get clues to the application architecture, and determine if the problem lies in the application or in the system libraries.

If the problem appears to be in the kernel, profiling can be a very useful tool. Even if you don't do kernel work, useful information can be gained. For example, if you discover that a significant amount of time is being spent in a device driver, you might want to try a different device. If you have a cooperative OS vendor (such as BSDI), being able to tell them the location of your kernel's hot spots might be just the incentive they need to work on that particular code.

The `time`(1) command can help you distinguish between CPU under-utilization due to outside delays (disk or network) and having an excess of CPU capacity. When time reports a wall clock (real) time that is significantly longer than the combined user and system times, you should suspect that you're seeing an I/O (or memory) problem. Small variances are to be expected due to other processes being given time to run.

Some types of loads will result in misleading numbers from `vmstat`(8) and its friends. For example, on a gateway, the load of handling the network interfaces will not be reported by `top`(1) or `vmstat`(8). In such cases a "cycle soaker" can be a useful thing. It will tell you how much CPU is actually available for computation.

Profiling User Code

If most of your CPU time is being spent in user mode and `ktrace`(1) doesn't give you enough information to improve the situation (or to place the blame and move on), then profiling the program in question might be the way to go. To profile the application you (or the vendor) will have to recompile the code. If this is not possible, you will need to choose another strategy.

The procedure for profiling user code is slightly different from that for profiling the kernel:

- Arrange to run the C compiler with the `-pg` flag. Probably the easiest way to do this is by adding it to the CFLAGS in the Makefile for the program. You may also need to arrange to link statically.
- Run the program. When it terminates, the file `gmon.out` will be produced in the directory where program was run. You won't get gmon.out until the program terminates.
- Display the profiling data with `gprof`(1). The profiling data will not include kernel time spent on the process's behalf. It may be necessary to piece together evidence from kernel and user profiling in order to get a full picture of where the cycles are going.

One of the challenges of profiling servers is to get the server to run in a mode where useful data can be collected. Servers that fork to handle requests will have one instance of the server process that will typically run for the life of the system, forking a new instance each time a request comes in. In this case, two different threads of execution will need to be profiled. The easy one to profile is the "master" instance that accepts the incoming requests and then forks to handle them. The processes created to handle the requests are harder to profile, and due to their short life may not produce statistically valid data. To get around this you may have to figure out a way to get the server to handle requests directly without forking.

Profiling the kernel

Typically, this is only interesting when you are out of CPU and want to figure out where the cycles went **and** you're really sure that the kernel is at fault. To do this, you'll need to build a profiling kernel and then reboot with that kernel. Profiling can be turned on and off, allowing you to collect data representative of the load you are interested in. Running a profiling kernel does add a bit of overhead to the system.

Here are the steps for doing this on BSD/OS:

- Configure and compile a profiling kernel:

```
# cd /usr/src/sys/i386/conf
# /usr/sbin/config -p MY_KERNEL
# cd ../../compile/MY_KERNEL.PROF
# make depend all
```

- Install the new kernel and reboot.

Figure 1 - Kernel profiling output

```
                                     called/total    parents
index  %time     self descendants  called+self     name           index
called/total     children
                                                 <spontaneous> [1]     53.0
0.19    85.75                           doreti [1]
                 0.19    80.64             97968/97968     _isa_intrswitch [2]
                 0.76    4.02              22237/43750     _ipintr [11]
                 0.00    0.14               2264/3158      _softclock [41]
                 0.00    0.00                 60/971       _trap [53]
                 0.00    0.00                  1/1         _arpintr [403]
-----------------------------------------------
                 0.19    80.64             97968/97968     doreti [1]
[2]     49.9     0.19    80.64             97968           _isa_intrswitch [2]
                 25.63   54.99             97893/97893     _efintr [3]
                 0.00    0.01                 75/75        _wdcintr [91]
-----------------------------------------------
                 25.63   54.99             97893/97893     _isa_intrswitch [2]
[3]     49.8     25.63   54.99             97893           _efintr [3]
                 53.68   0.00           12626901/12626907  _insw [5]
                 0.74    0.00              97893/97893     _ef_newm [24]
```

```
                0.56    0.01            97893/97893      _ether_input [27]
```

- When you are ready to collect profiling data, enable profiling with:

```
# kgmon -rb
```

- Run your load.
- When you're done collecting data, disable profiling with:

```
# kgmon -h
```

- Then dump the collected data with:

```
# kgmon -p
```

The data is left in a file called gmon.out in the current directory.

- The profiling data is displayed by `gprof`(1). `gprof` has almost as many options as `ls`(1), but the following should work pretty well for starters:

```
# gprof /bsd gmon.out
```

You may want to redirect output to a file so that you can use `perl` to help you make sense of the data.
For more details, take a look at the `config`(8), `kgmon`(8) and `gprof`(1) man pages.

Typically, what you're looking for in the profiling data is a rough idea of where the kernel is spending its time. Is it a device driver (which one)? The file system? The network, etc.? Often the answer is surprising. What you do next depends on where the data leads you and the amount of kernel hacking you want (or are allowed) to do.

For example, if the data points to the Ethernet driver (see *Kernel Profiling Example* below), you will either want to experiment with a different card or dig deeper into the profiling data to see if you can find some room for improvement in the driver (or suggest that your OS or Ethernet vendor do the same). On the other hand, if the data points you to `fork`, the place to look is at your load.

Kernel Profiling Example

See Figure 1 for sample kernel profiling output. The data is from a profiling run on a laptop with a heavy load of network traffic. The Ethernet interface is a 3com 3c589B which uses the `ef` driver.

Looking at the data, one of the first things to notice is that the `efintr` routine is right up near the top, so we can pretty reasonably focus our interest on the Ethernet interface. One of the next things to notice is that most of the time in `efintr` is attributed to `insw`. `insw` is part of the machine dependent assembler code in locore.s, doing signed 16-bit reads from the Ethernet interface. This is not surprising; since the 3c589B uses Programmed I/O (PIO), it is also a reasonable bet that `insw` was written with considerable concern for efficiency. In this case, it looks like the right thing to do is to try a different Ethernet card, looking for one that does DMA.

Code Availability

A collection of useful benchmarks (and pointers to more) can be found at **ftp://ftp.bsdi.com/bsdi/benchmarks**. Source code for tcpdump can be found on the contrib source CD in `contrib/tcpdump`.

References

1. Anawat Chankhunthod, Peter B. Danzig, Chuck Neerdaels, Michael F. Schwartz and Kurt J. Worrell. *A Hierarchical Internet Object Cache*. Technical Report 95-611, Computer Science Department, University of Southern California, Los Angeles, California, March 1995. Also Technical Report CU-CS-766-95, Department of Computer Science, University of Colorado, Boulder, Colorado: **http://harvest.transarc.com/afs/transarc.com/public/trg/Harvest/papers.html#cache**.
2. Aaron Brown and Margo Seltzer, "Operating System Benchmarking in the Wake of Lmbench: A Case Study of the Performance of NetBSD on the Intel x86 Architecture," *Invited Talk Notes of the 1997 USENIX Conference*, Anaheim, CA: **http://www.eecs.harvard.edu/vino/perf/hbench/sigmetrics/hbench.html**.
3. W. Richard Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison Wesley, 1994.

Programmer's Notes

This section is an overview of the BSD/OS programming environment, with an emphasis on changes that have occurred since BSD/OS 3.1. We won't be discussing bug fixes or minor improvements, but we will cover major new features and changes to programming interfaces. In most cases we will point you at the appropriate manual page or other documentation for more details.

ELF

The biggest new programming feature in BSD/OS 4.0 is **ELF**. ELF is an executable binary file format that Unix made standard and which other operating systems such as Linux have also adopted. ELF provides support for true dynamic linking, which is the main reason BSDI has switched to it. Almost all programs in the BSD/OS 4.0 distribution are ELF executable binary files. See `elf`(5) for details about the actual file format.

ELF programs from BSD/OS 4.0 will not run under earlier versions of BSD/OS. However, most *a.out* binary files from earlier versions of BSD/OS will run under BSD/OS 4.0. Exceptions include programs that examine certain kernel data structures directly via `/dev/kmem` or the `kvm`(3) library, or that rely on programs having the old a.out format.

We upgraded or replaced many utilities to add support for ELF. Here are some details about the new environment.

C and C++

BSD/OS 4.0 uses the GNU C compiler, version 2. GCC 1 has been retired. Most systems in the real world now have enough resources to run GCC 2, and GCC 1 didn't support ELF, so we finally replaced it. `cc`, `gcc` and `gcc2` all refer to GCC 2 now.

The version of GCC 2 that we currently ship is GCC 2.7.2.1. Libg++ is version 2.7.1.

The compilers generate ELF assembly code. It is unlikely that this code will assemble on older versions of BSD/OS.

Dynamic linking is now supported. With dynamic linking, the program arranges to postpone much of the mapping of names to addresses until the program runs. Dynamic linking has many advantages for maintaining shared libraries. Dynamic linking generally has a small but measurable overhead, both at start-up and while the program runs. If you link a program the ordinary way using `cc` or `g++`, you will create a dynamically linked program. To create an old-fashioned statically linked program, use `cc -static`. You can continue to use `shlicc`(1) to link programs with statically linked shared libraries. (Most utilities are still built with `shlicc`().)

Dynamically linked libraries use ‘position independent code’. To compile source files for dynamically linked libraries, use `cc -fPIC`. See `cc(1)` for more details.

GCC 2 under BSD/OS defines the macro ‘`__bsdi__`’. If you write code that needs to know which operating system it is being compiled on, use `__bsdi__` to identify BSD/OS. On Intel architectures, we provide the symbol `__i386__` to distinguish the hardware environment.

as

We have upgraded the assembler to GAS 2.8.1. See `as(1)` for details about options. The old assembler didn’t support ELF, and also didn’t support many modern (Pentium® and later) opcodes.

The new assembler generates only ELF object files, and accepts ELF assembly code as input. Fortunately, most pseudo-opcodes from older assembly code are also accepted. The two most important incompatibilities are:

- Assembly names now look exactly like C names. The compiler does not automatically prepend an underscore.
- The operand of the `.align` directive is a value measured in bytes, rather than power-of-two bytes.

Note that assembly code that is written for dynamically linked libraries must be position-independent. There is no automatic way to convert assembly code to PIC. Often it will require substantial, manual changes.

ld

We have upgraded the linker to GLD 2.8.1, for its ELF and dynamic linking support. To manually link a dynamically linked library or ‘shared object’, use `ld -shared`. Normally this work is done for you when you use the compiler `cc(1)` to link. See `ld(1)` for more details.

We configured the linker so that it can link ELF and a.out object files and libraries into ELF or a.out executables. There are a few restrictions on compatibility:

- If an a.out object or library appears in the link, you must link with `-static`.
- If you want to produce an a.out executable, you should link with ‘`-static -m i386bsdi -e _start`’. Note that `ld` cannot generate the old ZMAGIC format; the standard QMAGIC format is the default.
- You don’t need to include any a.out (respectively, ELF) objects in order to link an a.out (respectively, ELF) program. However, we only provide C start-up files for ELF.
- Older object files that use out-of-date and incompatible data structures may not work with newer object files, even if linking succeeds.

The old `-T` option to set the text address has been replaced by `-Ttext` (and if necessary, `-Tdata`).

Other binutils programs

We have replaced `ar`(1)/`ranlib`(1) and `gprof`(1) with versions from the GNU binutils 2.8.1 distribution. These new versions support ELF and (in the case of `ar`) long filenames using Unix format. We also supply the `objdump`(1) program, which provides annotated disassembly of binary files and other features.

gdb

We supply the GNU debugger, version 4.16. Gdb is a text-oriented source code debugger for user programs and the operating system kernel. See `gdb`(1) for details about this program.

The system now generates ELF core files, even for a.out programs. ELF core files contain all of the writable segments of the program that died, unlike traditional core files, which only contained program-specific data and stack. Gdb correctly handles ELF core files with a.out programs, although it may produce a warning message. Gdb still supports old core files, if you have any.

We configured `gdb` to be able to handle ELF, a.out and COFF binaries. The compilers use the GNU variant of the DBX '*stab*' debugging format by default, rather than DWARF.

The `attach` and `detach` commands now work. These commands permit you to debug a program that was already started.

Other dynamic linking information

The program `/shlib/ld-bsdi.so` is the dynamic linker. The system loads this program into the address space of dynamically linked programs at start-up, and runs it. The dynamic linker finds the shared libraries that the program needs in order to run, loads them, relocates them. and links the program against them. If the dynamic linker is missing, no dynamically linked program will run. See `ld.so`(8) for more details.

The `ldconfig` program compiles a list of current shared libraries. When you add a new dynamic shared library to the system, you must run `ldconfig` to tell the dynamic linker about it. Se `ldconfig`(8) for details.

The Unix-style dynamic linking routines (`dlopen`(3) and others) now use the ELF dynamic linker. A consequence of this change is that programs that use these routines must be dynamically linked. There is no non-dynamic version of `-ldl`. Libraries that call `dlopen`(3) must themselves be dynamic. The Tcl library is an example of this situation.

Other ELF information

We upgraded or replaced several non-binutils programs and library routines for ELF. These programs handle both a.out and ELF format files. The list of programs includes:

- `bpatch`,

- kvm_mkdb,
- make,
- nm,
- shlist,
- size,
- strip,
- strings

The shlist(1) program can now list dynamically linked shared libraries, if the application is dynamically linked.

We changed the nlist(3) functions to handle both ELF and a.out. Because of the difference in naming conventions, nlist(3) checks for symbol name matches with and without an initial underscore.

The gcore(1) program now uses the fcore(2) system call to create ELF core files from running programs. This eliminates the need to specify the executable file when running gcore.

ELF programs have a different layout in memory from a.out programs. The text segment normally starts at 0x08048000. The data segment is contiguous with the text segment in the binary file, but it is offset by a page in memory, so that it can have a different protection. The stack starts at the base of the text segment and grows down. This limits the maximum stack size to a bit over 128 MB.

BSD/OS programs are 'branded' using a PT_NOTE header, to distinguish them from ELF programs supplied by other vendors.

The standard BSD/OS makefile header /usr/share/mk/bsd.lib.mk now builds up to four kinds of libraries from library sources:

- a regular archive library
- a statically linked shared library (if it is specified in /etc/shlib.map)
- a dynamically linked shared library (compiled with -fPIC)
- a profiling archive library (compiled with -pg)

The seldom-used symorder(1) program has not been upgraded to support ELF and is not available in BSD/OS 4.0.

Library information

BSD/OS provides an implementation of the POSIX *pthread* routines, which can create multiple threads of control in a single process. See pthreads(1) for a starting point. The current implementation does not use so-called kernel threads; it is implemented entirely within the C library.

The C library contains the resolver routines from BIND 4.9.6. The −lresolv library has debugging versions of these routines. See gethostbyname(3) for more information.

The sysctl(3) interfaces continue to evolve. Check the manual page for new information.

In BSD/OS 4.0, we introduce a new tunable malloc(3) implementation from FreeBSD. See the manual page for details.

We have added a number of routines to libkvm for 4.0. These include kvm_boottime(3), kvm_cpustats(3), kvm_disks(3), kvm_iflist(3), kvm_kmem(3), kvm_ncache(3), kvm_ttys(3), kvm_vmmeter(3), and kvm_vnodes(3). See the manual pages for details.

In addition to the traditional BSD select(3) call, we also supply the new POSIX pselect(3) call. We provide functions for creating and manipulating large file descriptor sets. See the select(2) manual page for more information.

There are some new interfaces in the networking support. There is a sockaddr_in6 structure for IPv6, and a number of new macros and support code for IPv6. The traditional socket-related system calls can be used with these new interfaces to implement IPv6 services. We supply POSIX.1g / RFC 2133 routines such as getaddrinfo(3), getnameinfo(3) and if_indextoname(3), plus isfdtype(3) and sockatmark(3).

The interface to the BSD/OS authentication system is now documented. See authenticate(3) and auth_subr(3) for information.

BSD/OS 4.0 includes new math libraries. On Intel architectures, there are now separate -lmi386 and -lmstd libraries. These libraries have identical interfaces, but the -lmi386 library uses transcendental instructions and provides only IEEE 754 error semantics, while the -lmstd library can be used safely on systems with floating point emulation and can be switched between IEEE, SVID, X/OPEN and POSIX error semantics. A script named mathlink(8) is run at system start-up and creates links for -lm to the appropriate library; by default it checks for the presence of hardware floating point in dmesg(8) output and chooses -lmi386 if the system has hardware floating point, or -lmstd if it does not have it. The new libraries are adapted from the FreeBSD math library, which was in turn adapted from the Sunpro/Netlib fdlibm library that BSDI formerly shipped as -lnm. The new libraries have ‘float’ versions of most math library functions that take ‘float’ rather than ‘double’ arguments. On Intel architectures, floating point arithmetic is always performed to double precision accuracy, so the ‘float’ functions don’t usually run faster than the equivalent ‘double’ functions. See math(3) for more details about the math library.

C types

In BSD/OS 4.0, several routines in the mmap(2) family switched from using the **caddr_t** type to pass addresses to **void ***, following the developing POSIX standard. The defined value MAP_FAILED can now be used to test for failure of the mmap(2) call. Check mmap(2), munmap(2), mprotect(2), msync(2), mlock(2) and madvise(2) for more details.

Also in BSD/OS 4.0 we have made some cosmetic changes to parameter types and field types for various networking-related system calls, to keep up with POSIX standards development. Several functions now take socklen_t or socklen_t * parameters, rather than int or int * parameters. Read accept(2), bind(2), connect(2), getpeername(2), getsockname(2), recvfrom(2) and sendto(2) to see what's new. There is a new sa_family_t type for socket families; this affects various sockaddr structures. The types in_addr_t and in_port_t can be used to hold IPv4 addresses and IPv4 ports, respectively.

The long double type is (still) a second class citizen, but the situation isn't quite as awful as it was in earlier releases. The library and applications are both compiled with GCC 2, so they agree on the form of long double values. However the printf(3) function converts long double values to double values before printing them, which loses precision.

Because BSD/OS supports large files, the C type off_t is a 64-bit number, a quad_t (also known as *long long*). This type is used as a parameter to various system calls such as lseek(2), truncate(2), and mmap(2), and is also used to store a file size in the stat structure. Programs that use the appropriate header files (<unistd.h> and/or <sys/types.h>) will work correctly with one exception, printing such a number. An off_t should be printed with the format specification %lld (two letter L's), or, although it is less portable, %qd.

This is a common problem with porting software to the BSD/OS system. We recommend that if you experience random failures or output problems in ported software, you check the software with the -Wall option to cc(1). In addition, search for calls to lseek, truncate, or mmap that are being passed arguments that are explicitly cast to a long, instead of long long:

```
  lseek(fd, 0L, SEEK_SET);    WRONG!
```

First, ensure that a correct function prototype is in scope, and then the correct call is:

```
  lseek(fd, 0, SEEK_SET);
```

Program Debugging

This section gives an overview of what you can do when a program isn't working correctly. First we cover four ways to produce debugging information from a program, then we introduce gdb, the GNU debugger.

Program Tracing

If a program is doing something unexpected (for example, it's printing '`Permission denied`' or '`No such file or directory`' errors for no obvious reason), but it is not providing sufficient information to enable you to understand the real problem, you can often determine the cause by tracing the system calls that the program is executing. To do this, run the program using `ktrace`(1). For example:

```
% ktrace cat some_file
```

This example traces all the system calls that occur when running '`cat some_file`'.

The `ktrace` program causes the kernel to direct raw system call trace information into a file named `ktrace.out` in the current directory. When the program finishes running, you may invoke the `kdump`(1) program to display this information in a human readable form. `kdump` can display the name of each system call, the call's arguments (in hexadecimal) and its return value; each filename translation, printing the filename as a string; each I/O, dumping the contents of the operation; and each signal.

The `ktrace` tool works even if you have not compiled the program specifically for debugging. It works well with system utilities and your own applications. `ktrace` won't make a program run much slower, but `ktrace`('s) output files can quickly grow to huge sizes. There are several useful options to `ktrace` and `kdump`. See the manual pages for details.

Core Dumps

Sometimes the system will save the state of a program in a file, usually called a *'core dump'* or *'core file'*. (When an early computer encountered a terminal error condition, it would write the data in its magnetic core memory to tape; the term *'core dump'* is a salute to those early times.)

The system will automatically terminate a program and write its state to a core file when the program encounters certain kinds of errors. Common errors of this sort include references through invalid pointers, dividing by zero and calling the `abort`(3) function. Automatic core dumps are placed in the program's current directory under a name ending in '`.core`'. For example, if the `cat` program fails and dumps core, the core file will be named `cat.core`.

You may request a core file from a program by using the 'quit' character at the keyboard (usually control-backslash or control-. Typing this character will cause the program to terminate and the system will write a core file for it. A program may choose to disable this feature. There are other ways to ask a program to terminate and produce a core file; see the `sigaction`(2) manual page. Core files produced this way have the same name and contents as automatically generated core files.

You may request a core file from a program without killing the program by using the `gcore(1)` program. `gcore` takes a snapshot of the program's state and saves that state into a core file. These core files are identical in form to other core files. See the `gcore(1)` manual page for more details.

A core file is an ELF binary file. It contains copies of the writable memory areas of the program, including the initialized data, the `malloc`(3) data and the stack. The core file also contains values for the hardware registers and other hardware and system information. The `gdb(1)` program reads core files and displays information about the program; see the section *Gdb* below for more information.

Breakpoint Debugging

The third main way to debug a program is to manipulate the running program with `gdb(1)`. You can use gdb to run a program and make it stop and continue at various points, and print the values of variables and the source code that the program is executing. `gdb` uses the `ptrace`(2) system call to manipulate the program. See the section *Gdb* below for more details.

Malloc Debugging

The `malloc`(3) function is the root of many C programming errors, since programs often fail to allocate or free memory correctly. BSD/OS 4.0 provides two ways to debug problems with `malloc`.

The BSD/OS 4.0 `malloc` function (derived from FreeBSD) supports 'malloc options'. These options may be supplied at compile time using the global string variable malloc_options, or at run time using the environment variable MALLOC_OPTIONS. Options are a single letter; upper-case letters enable features, while lower-case letters disable features. The 'A' option causes warnings to be fatal, producing core dumps. The 'J' option writes garbage (0xd0) to new memory. The 'Z' option zeroes out new memory. Any use of a `malloc` option will enable extra warnings as well.

If you can build and link a program for debugging, there is a very useful library-level run-time debugging tool. BSD/OS 4.0 distributes Bruce Perens' library, *Electric Fence*. Electric Fence helps you detect two common programming bugs: software that overruns the boundaries of a `malloc`(3) memory allocation, and software that touches a memory allocation that has already been deallocated by `free`(3). Unlike other malloc debuggers, Electric Fence will detect read accesses as well as writes, and it will pinpoint the exact instruction that causes an error. See the `libefence`(3) manual page for additional information.

Gdb

The GNU debugger, `gdb(1)`, is a general tool for displaying information about a running or dead program. The amount of useful information depends on a number of factors, including the way that the program was built (see *Compiling for Gdb* below). With full debugging information, `gdb` can display source code lines as they are executed and can print or set the values of variables in the appropriate format.

If you wish to use breakpoint debugging with (for example) the `cat` program, you should use:

```
% gdb cat
```

If you have a core file from the `cat` program, you may run:

```
% gdb cat cat.core
```

to examine the core file. Note that the `cat` program must be in your current directory, or you must provide a pathname for it.

Here is a longer example:

```
% cc -static -g -o cat /usr/src/bin/cat/cat.c
% gdb cat
GDB is free software and you are welcome to distribute copies of it
 under certain conditions; type "show copying" to see the conditions.
There is absolutely no warranty for GDB; type "show warranty" for details.
GDB 4.16 (i386-unknown-bsdi4.0),
Copyright 1996 Free Software Foundation, Inc...
(gdb) run
Starting program: /usr/var/tmp/cat
^C
Program received signal SIGINT, Interrupt.
0x8051a33 in _syscall_sys_read ()
(gdb) where
#0  0x8051a33 in _syscall_sys_read ()
#1  0x80488e2 in raw_cat (rfd=0) at /usr/src/bin/cat/cat.c:256
#2  0x80487f4 in raw_args (argv=0x8047b0c) at /usr/src/bin/cat/cat.c:231
#3  0x8048369 in main (argc=1, argv=0x8047b0c) at /usr/src/bin/cat/cat.c:109
#4  0x80480fc in __start ()
(gdb) up
#1  0x80488e2 in raw_cat (rfd=0) at /usr/src/bin/cat/cat.c:256
256             while ((nr = read(rfd, buf, bsize)) > 0)
(gdb) list
251                             err(1, "%s", filename);
252                     bsize = MAX(sbuf.st_blksize, 1024);
253                     if ((buf = malloc((u_int)bsize)) == NULL)
254                             err(1, NULL);
255             }
256             while ((nr = read(rfd, buf, bsize)) > 0)
257                     for (off = 0; nr; nr -= nw, off += nw)
258                             if ((nw = write(wfd, buf + off, nr)) < 0)
259                                     err(1, "stdout");
```

```
260             if (nr < 0) {
(gdb) print bsize
$1 = 65536
(gdb)
```

As you can see, the control-C caused an interrupt signal (SIGINT) to be sent to the program. According to the stack backtrace, the signal was received while the program was at line 256 of the `raw_cat` function, executing a `read`(2) system call.

The most useful gdb commands are:

`run`	execute the program
`where`	get a stack backtrace
`break`	insert a breakpoint
`cont`	continue running from a breakpoint
`print`	print variables
`list`	list source code
`help`	describe commands

Use '`man gdb`', '`info gdb`' or gdb's on-line *help* feature to get more information about `gdb`.

Compiling for Gdb

Gdb will operate on any executable program, including system utilities, but it works best when you have built the program to include debugging information.

When you compile a program with the `-g` flag, as in the example above, the compiler inserts information about the program's source code into the binary file. This information helps `gdb` to find the source code lines that correspond to machine instructions in the binary file, and to determine the names, types and locations of variables.

If you compile without the `-g` flag, the compiler will still insert some information in the file that `gdb` can use. This information gives the names and locations of functions and variables, but not their types or their relationship to the source code. If you compile with the `-s` flag or run `strip`(1) on the binary file, then the program is *stripped* of all name information and debugging information, and `gdb` won't be able to display any names at all. In both cases, `gdb` cannot figure out how to refer to the source code, so you are usually reduced to examining the machine code or dumping raw data from specific locations.

`Gdb` works best when a program is not linked for shared libraries. To avoid shared libraries, compile with `cc`(1) or `g++`(1), not `shlicc`(1) or `shlicc++`(1), and use the `-static` flag, as in the example above. (`gdb` can display some information from shared libraries when it is tweaked appropriately, but this is not for the timid, and it won't be described here.)

gdb also works better when a program is not compiled with optimization. Optimization can make a program run much faster, but it also causes variables to be optimized away or combined with temporary values or other variables, and it can intermix the effects of source code lines so that the line number that gdb shows becomes misleading. Gdb can still get lots of information from an optimized program, and we recommend using optimization unless experience shows that gdb cannot display useful information about the optimized program.

Rebuilding the Kernel

The `roff` source file `/usr/share/doc/bsdi/config.n` describes kernel configuration and building in great detail. Use `make config.ps` from the `doc/bsdi` directory to make a PostScript file or view it with `man -M bsdi config`.

Loading the Source

Use `installsw`(8) to load the kernel source or object package (binary systems include the kernel object package; it's a limited source plus object distribution that includes the source files needed to re-configure). Once you have the kernel (`/sys`) hierarchy extracted, you can re-configure and build new kernels. The object kernel hierarchy is loaded automatically by Express Installation.

Debug objects for binary systems

For binary customers an additional kernel object package is available in `/diag/sysobj.debug.pax.gz`. This kernel tree has objects with debugging options enabled that are not available in the standard build supplied by default. To use the debug objects instead of the standard objects, unpack the pax archive on a filesystem with at least 50MB of free space; the instructions below apply equally well to building in this debug tree, however the DGENERIC configuration should be used in place of GENERIC. When building from full sources the debug object tree is not needed.

See the section *Kernel Debugging* for more information regarding the debug features.

Configuring the Kernel

The first step is to set up a configuration file. Configuration files are located in `/sys/i386/conf`. The `GENERIC` configuration file is used to build the kernel included on the distribution and can be used to boot from a floppy, or hard disk (SCSI or IDE/ESDI/etc.). The `GENERIC` configuration file includes sample configuration lines for all the supported devices, although some device configurations are commented out in the file. Lines beginning with # are comments and are ignored.

To create a configuration file for your local system, copy the `GENERIC` configuration file to a new name (e.g., `LOCAL`) and then edit it to suit your needs.

Note that if you have a source distribution and want to build from source the

```
  options SOURCE
```

line in your configuration file **`must`** be uncommented (the # removed). Failure to do this will cause the object files in `/sys/i386/OBJ` to be used instead of the source files.

Tuning

The single variable that has the greatest effect on kernel tuning is `maxusers`. Despite its name, this variable does not limit your system's capacity directly or relate to licensing arrangements. Increasing `maxusers` has the effect of increasing size of the kernel's data structures and, hence, its usage of RAM. If your system will have many processes (e.g., many WWW connections), consider doubling or even quadrupling `maxusers`.

Running config

Once you have a configuration file tailored for your machine, run the `config` program to build the appropriate `include` and `Makefiles`. For example, to set up a new configuration called `LOCAL` you would run the following commands:

```
# cd /sys/i386/conf
# cp GENERIC LOCAL
# edit the LOCAL configuration file as needed
# /usr/sbin/config LOCAL
```

The `config` program creates the directory `/sys/compile/LOCAL` and populates it with all the necessary header files and a `Makefile`. Regrettably, the `config` program is poor at reporting errors. Be sure to monitor your kernel build output carefully for any errors that might have occurred.

A kernel configuration that uses

```
config bsd swap generic
```

must have `GENERIC` defined using

```
options GENERIC
```

and must have all of the disk types configured (including `mcd`). If you leave the

```
config bsd swap generic
```

line in your local configuration file, you must also leave the

```
options GENERIC
```

line. If you wish to remove some of the disk devices and/or switch to one of the

```
config bsd root on ...
```

type lines (which specify the root and swap devices explicitly), you must remove the

```
options GENERIC
```

line.

If building from source, remember to uncomment the

```
options SOURCE
```

line in your configuration file!

The kernel **will not compile** if these rules are not followed!

The binary release requires the definition of QUOTA and MULTICAST in the configuration file.

Building Kernels

To build your new kernel, change directories to the subdirectory with the same name as your configuration file in the directory /sys/compile/, and run make:

```
# cd /sys/compile/LOCAL
# make clean
# make depend
# make
```

The last make command should create an executable kernel called "bsd" in the current directory. It takes less than 5 minutes on Pentium 90 processors.

To install this kernel from the directory in which it was built, save the current kernel under a different name, copy the new one to /bsd, and reboot:

```
# cp /bsd /bsd.save
# cp bsd /bsd
# reboot
```

We recommend that you leave the original distribution kernel in /bsd.generic so you always have a usable kernel to fall back on. See boot(8) for additional details on booting. To boot a kernel other than the default /bsd, you must interrupt the normal boot sequence by typing <Enter> when prompted to

```
Press any key to interrupt boot sequence.
```

The boot program should then respond with a Boot: prompt. At the Boot: prompt, you can load whatever kernel you wish:

```
Boot: bios(1,0,0,0)/bsd.save
```

This will load the "bsd.save" kernel from the filesystem on the boot drive.

When you boot an alternate kernel using this method, the system will come up to single-user mode rather than multi-user mode and the root filesystem will be left read-only. To put the bsd.save kernel back into place and reboot, run the commands:

```
# fsck /dev/rwd0a              /dev/rsd0a for SCSI
# mount -u /                   enable writing on root
# cp /bsd.save /bsd
# reboot
```

Unsupported Kernel Modules

As a convenience to our customers, BSDI includes some software for which BSDI offers no support at all. Users are not encouraged to use this software. The software might well be broken and cause your system to break in all the worst possible ways (e.g., you to lose your data).

For BSD/OS 4.0, this list includes:

- the fdesc, kernfs, LFS, portal and procfs filesystems
- the ISO network protocols
- the CCITT/X.25 network protocol (no hardware drivers are available)

None of these is configured into the standard BSDI kernel. Configure and use them at your own risk!

Kernel Debugging

In the event your system is hanging or crashing (unexpectedly rebooting), please use one of the checklists below and report the results to support@bsdi.com (or your vendor's support address). You might also print the information gathered and fax it to us if you do not have email access. Of course, if you can think of any other facts or data that might be relevant, please include them.

System Hang Checklist

If your system is crashing (unexpectedly rebooting), skip this section.

1. Do the keyboard LEDs blink rapidly when the system first hangs? If the BSD/OS kernel crashes, it blinks the keyboard LEDs sequentially while attempting to save a core dump on the swap device. All three LEDs are turned on if the dump is successful; all are turned off if the dump fails. This is intended as a visual indication of a crash when X is running and the panic message is not visible. A memory dump can take a while for a system with a lot of memory, so the system can appear to be hung. If the system is actually crashing, proceed as described in the crash checklist.
2. If the system is on a network: can you ping it while it is hung?
3. If you can ping the system, how far do you get trying to telnet to it? Please record the progress made when attempting to log in via a telnet session exactly.
4. If using a text mode console: do characters echo when typed on the console?
5. If running X: can you make the mouse pointer move?
6. If not running X: do the Num Lock or Caps Lock keys cause the keyboard LEDs to change?
7. Is the disk drive busy after the system hangs? (Look at the disk activity LED or listen for disk activity.)
8. Run `/bin/sh` or `/bin/csh` on the console and execute:

   ```
   stty status ^T
   ```

   Then cause the hang and try typing <Control-T>. If you get output like:

   ```
   load: 0.16  cmd: cat 5935 [ttyin] 0.00u 0.02s 0% 0k
   ```

   please include this information. If you get no output, let us know.
9. If you are not running X on the console, does <Control-alt-delete> cause a prompt asking whether to reboot? (If the console is in the wrong mode you might not see the "reboot?" prompt. Go ahead and type "y" to start the reboot.)
10. If you can reboot with <Control-alt-delete>, it might be possible to force a crash dump after the hang occurs. There are two ways to arrange for this. You can add

    ```
    options DEBUG
    ```

    to your kernel configuration file before building and installing a new kernel.

Or, you can modify the running system to enable crash dumps by entering the following command as superuser (root):

```
# sysctl -w machdep.cntlaltdelcore=1
```

This command can be placed in `/etc/rc.local` to enable core dumps after future system reboots.

When the system hangs, you will be able to use <Control-alt-delete> to generate a crash dump. The system will ask you if you want to reboot, and you should type "***y***". Then the system will ask you if you want to create a crash dump. Type "***y***" again. Once you have a dump, go on to the next section.

If the hang happens infrequently you might wish to use the deadman timer facility described below.

If the above information isn't sufficient to explain to us what is happening on your system, we might send you additional instructions or scripts to execute. If the hang cannot be explained by problems with user mode programs or daemons, we will almost always need a crash dump to analyze the problem further.

System Crash Checklist

If your system is crashing, please follow these steps. Look at the tips section below for information on how dumps are written and how to obtain a kernel with debugging symbol tables.

1. Make sure you are getting a crash dump and it is being saved in `/var/crash`. You might have to create `/var/crash` as a directory (using `mkdir`(1)) or create a link to another directory (using the `-s` option to `ln`(1)). The chosen directory must be on a filesystem with enough free space to hold a full copy of system RAM plus a copy of `/bsd`. If the system reboots itself unexpectedly but does not write a dump (the keyboard LEDs do not flash) try to catch the last message (the one containing the string '`panic`') on the system console before the system reboots.
2. Find the `bsd.gdb` file that matches the running kernel. The `bsd.gdb` file can be very important to identifying the cause of the crash under `kanal` (see below). If you loaded the kernel object or source software package, and you have not built a custom kernel (see the section *Rebuilding the Kernel*), the debugging kernel will be named:

   ```
   /sys/compile/GENERIC/bsd.gdb
   ```

   An express installation will load the kernel object package by default. If you have built and booted a custom kernel named (say) *LOCAL*, then your debugging kernel file will be named:

   ```
   /sys/compile/LOCAL/bsd.gdb
   ```

   Substitute the name of your kernel for 'LOCAL'. If you did not load the kernel software package, you can still find a debugging kernel on the CD-ROM. If the CD-ROM is mounted as `/cdrom`, then the debugging kernel will be named:

```
/cdrom/sys/compile/GENERIC/bsd.gdb
```

3. It is often much easier to debug a problem if we have access to more than one crash dump. If possible, please obtain two or more crash dumps before contacting us. To save space in `/var/crash`, you can delete identical `bsd.#` files. Use the `cmp`(1) command if you are unsure which files are identical. Compressing the `bsdcore.#` files (with `gzip`(1)) is another way to save space when you run low.

Once you have created the above environment, you are ready to gather detailed debugging information.

To gather debugging information change directories to `/var/crash` (or wherever you have the `bsdcore.N` and `bsd.N` files saved) and run the command '`kanal` *N*', where *N* is the number of the crash dump you need analyzed (the N in `bsdcore.N`). The `bsdcore.N` file must NOT be compressed for this step.

The `kanal` program will ask various questions about the crash and system environment. `kanal` will then run various analysis steps on the core dump and write a (lengthy) summary to a shar file called `info.N` in the current directory. During analysis, you might be asked to supply additional information, such as the path to the `bsd.gdb` file that matches the kernel you were running at the time of the crash.

Once the analysis is complete the `info.N` file should be transferred to BSDI's ftp server; log in as user `anonymous` and put the `info.N` file into the `/bsdi/support/incoming` directory with a descriptive name that includes your customer number or email address (for example: `info.2.n1234` or `info.2.ewv`). Note that `/bsdi/support/incoming` is different from `/incoming`, which BSDI uses for non-support-related FTPs.

After uploading the crash information, contact support through your normal support channel (telephone, email, or fax), send a brief description of the crash circumstances, and make sure to include the name of the file you uploaded.

If you can not upload the information to BSDI over the Internet, please contact support for further instructions.

Notes and Tips

How to Write and Save a Crash Dump

When BSD/OS crashes, the system prints one or more messages on the console, and, in some cases, it copies the contents of memory into a 'crash dump' on disk. While this image is being written, BSD/OS is frozen (not allowed to run). By analyzing this snapshot of the system with `gdb` it is often possible to understand what caused the system to crash (or hang).

While the system is writing the dump, a countdown is displayed on the console indicating the number of megabytes of memory remaining to be written; additionally, the keyboard LEDs are flashed rapidly. If X windows is running, the console messages will not be visible but the keyboard LEDs will still blink.

To recover the information from the dump on reboot (an operation performed as part of the standard booting procedure), the directory `/var/crash/` must exist (or be a symbolic link to a directory where you have sufficient free space). When the `/var/crash/` directory is present, the command `savecore /var/crash` is executed automatically when the system reboots. The `savecore` program saves information about the crash dump in `/var/crash/`: the file `/var/crash/bsdcore.#` will contain the core dump and the file `/var/crash/bsd.#` will contain a copy of the running kernel (where # is 0, 1, 2 and so on). You will need at least as much free space in `/var/crash` as you have main system memory plus room for a copy of the kernel.

Kernel Debugging Symbol Tables

Debugging a crash dump is much simpler with a kernel containing debug symbol tables; the semi-automated analysis provided by the `kanal` program is only possible if debug symbol tables are available. When a kernel is built with debugging enabled, the output of the build process is two files: `bsd` and `bsd.gdb`. The code and data in these two files are identical and the system can be booted from either one; however, `bsd.gdb` contains the debugging symbol tables needed to debug the core dump using `gdb` or `kanal`. Typically, the version without symbol tables is installed in `/bsd` due to the large size of `bsd.gdb`. During the core dump, `savecore`(8) copies `/bsd` into `/var/crash/bsd.#`. As this is usually the kernel image without the debugging symbol tables, it is good practice to keep a copy of the `bsd.gdb` matching the running system around (be especially careful to save a copy of the `bsd.gdb` matching `/bsd` when building a new kernel).

Hardware Tips

Many of the problems we see are due to broken hardware. Many times marginal hardware works fine with one release of BSD/OS and then starts crashing on a different (configuration or release) kernel.

Look for some of the following signs that a crash is hardware related:

- Message on the console (or `dmesg` output) containing the string `NMI`.
- The value of `eip` listed in the panic message changes for each panic.
- Corrupted data in files.
- Apparently random creation of core files from user processes.

In many cases, changing various configuration parameters or swapping parts around can pinpoint where a problem is. This isn't intended to be a complete list, just some items we've seen trouble with in the past:

- Be sure main memory parity checking is enabled. We do not recommend using machines with 8 bit (non-parity) RAM or those unable to support parity checking (such as those based on the Intel Triton chipset).

- Disable external cache via the BIOS.
- Slow down cache cycle times.
- Slow down main memory (add wait states).
- Remove 1/2 of main memory.
- Replace bottom half of main memory with the SIMMs from the top half.
- Remove or replace the IDE controller. If this is not possible, try removing the second (slave) device from the IDE cable (a broken IDE peripheral can cause very strange system behavior).
- Disable internal (CPU) cache.
- On Adaptec AHA-15xx SCSI adapters, slow down DMA speed (to 3.3 MB/s).
- Disable fast SCSI.
- Disable synchronous SCSI.
- Swap the motherboard.
- Turn off all BIOS shadowing options.
- Check for overheating due to broken or clogged CPU fan.
- Replace CPU.

Remember, you must run the EISA configuration utility that came with your motherboard any time you change EISA hardware or add more memory on an EISA machine.

Memory errors

BSDI does not recommend running systems without some form of memory error detection or correction hardware (simple parity or ECC/SECDED). On systems with memory error protection it is sometimes desirable to ignore parity/SECDED errors instead of allowing them to crash the system (the default behavior); typically this situation arises when either the parity logic is broken (rare) or the SECDED/ECC logic is correcting an infrequent single bit error. This can be accomplished by running:

```
# sysctl -w machdep.parityerrorpanic=0
```

Note that this leaves the system open to undetected data corruption (as does running a system without parity memory). This should generally only be used as a stop-gap measure while waiting to replace a (slowly) failing SIMM or DIMM module.

Due to the nature of PC-class hardware, BSD/OS does not have a detailed indication of what SIMM/DIMM is failing when the hardware reports a memory error to the kernel – swapping memory modules around (as described above) or running memory diagnostics are the two best methods to track down a failing memory module.

Deadman Timer

Situations have arisen in the past where the kernel is running more-or-less normally but user processes are not getting scheduled for execution (this sometimes happens when a disk stops responding for example). It is not always possible or convenient to have an operator request a system dump from the console under these circumstances; since many minutes might have passed between the time the system hung and the dump was taken it might be difficult or impossible to determine the reason the system hung from the dump image.

To aid in debugging this class of problem a deadman timer facility is available. A deadman timer is a timer built into the kernel that must be reset at regular intervals by user code; if the system is hung the user mode program will not clear the timer and the system will panic and write a dump (if it is properly configured for dumping as discussed above).

The deadman timer is started by running:

```
# sysctl -w kern.deadmantimer=N
```

where N is the number of system timer ticks before the timer will next expire. The default clock tick rate is 100 ticks per second (10ms/tick), so setting the deadman timer to 100 will result in a panic 1 second after the command is issued unless the timer is reset again. Setting the timer to 0 disables it.

In normal usage a window is opened on the console and a small shell-script loop is started to periodically rearm the timer:

```
# while (1)
? sysctl -w kern.deadmantimer=6000
? sleep 30
? end
```

Note that in the above example the timer is set for one minute every thirty seconds – this leaves a margin for error if the system gets busy but does not hang. Depending on system load (most critically on the oversubscription of physical memory) the safety margin might need to be increased to prevent spurious panics.

An alternate way to use the deadman timer is to use a much larger value (over 1 minute) and reset the timer via a `crontab` entry – this method has the advantage of automatically rearming the timer when the system reboots; the disadvantage is that much larger timer values must be used since the granularity of cron is much coarser than 1 second.

Sending a Dump to BSDI for analysis

If we cannot get enough information from the above steps, we might ask you to run some additional commands against the dump files. Occasionally, we might ask you make the dump and `bsd.gdb` file available to us for further analysis. In the past, we have used the following methods:

- You can set up an account on your system that has read access to the files in `/var/crash` (including the `bsd.gdb` file and the sources used to build the kernel if built from source). We can either dial in via modem or access your system via the internet. Looking at the dump on your machine is the preferred method since it does not involve moving large blocks of data around.
- You can build a compressed archive file consisting of the `bsd.gdb` and `bsdcore.#` files, then transfer it to us via `ftp`(1), tape, or CDR. The

command to create this archive file (from the `/var/crash` directory) is:

```
  # pax -w -v bsd.gdb bsdcore.0 | \
          gzip > kernel.archive.pax.gz
```

PLEASE do NOT upload a full dump to our `ftp` server unless we request it.

Advanced debugging topics

If you are a developer and are attempting to debug kernel problems, additional tools are available to assist you.

Kernel gdb

The debugger (`gdb`) can be run against a system dump to look at the state of the system after a crash or panic. To start the debugger on a core dump, the `-k` flag is used:

```
# gdb -k bsd.gdb bsdcore.0
```

Once in the debugger one can debug the kernel just as one would debug a user mode core dump. One additional facility that is useful in this mode is the `paddr` command, when passed the address of a proc structure in memory paddr switches gdb's view of virtual memory to that of the selected process. This allows the kernel stacks and user memory of multiple processes to be examined easily. The `ps` command (found in `/sys/scripts/bsdi.ps`) can be installed with the command:

```
gdb> source /sys/scripts/bsdi.ps
```

This gdb script will print all the procs on the system and their proc structure addresses (for use with `paddr`).

Kernel GDB on a live system

It is possible to run `gdb` on a live system by specifying the kernel memory device instead of a dump file:

```
# gdb -k bsd.gdb /dev/mem
```

Since kernel memory will be changing while the debugger runs results may be inconsistent with this method.

Cross system GDB

A live system may be debugged via `gdb` running on a seperate system. The target system (the one to be debugged) is connected to the system running gdb with a null modem cable. With the target kernel configured for serial GDB the command:

```
gdb> target remote /dev/ttyxx
```

will halt the target processor and put it under control of `gdb` running on the debugging system. The `paddr` command does not work in this mode; only the current processes virtual memory and stack may be examined.

With cross system gdb it is possible to set breakpoints just as with a user program (the `cont` command resumes the target machine until the next breakpoint is hit).

A generic kernel with cross system debug support enabled is available in `/diag/bsd.debug`; for source customers the following options must be present in the target kernel to use cross system gdb:

```
options KGDB
options "KGDBRATE=9600"
options "KGDBDEV=0x800001"
```

The rate may be higher if the debugging machine uses the command:

```
gdb> set sl-baudrate xxxx
```

to match. The device is a raw major/minor number (in this case `/dev/tty01`).

A getty should not be run on the debugging serial port.

Kdebug

An in-kernel command line debugger is also available for cases where cross system debugging is not practical. It is included in the debug kernel (`/diag/bsd.debug`), source customers may build a kernel with:

```
options KDEBUG
```

to include it. The `kdebug`(4) man page contains substantial details on the use of kdebug; the most useful commands after a crash are `trace` (dump an entry in the trace buffer) and `bt` (current stack backtrace).

Kdebug is not a source level debugger, one must have a copy of the kernel symbol table available to get the most use out of kdebug. When debugging problems where memory gets semi-randomly corrupted, kdebug provides the means to use the CPU's internal debugging registers to watch for reads or writes to a particular location. It is also possible to set code breakpoints, but not to single step instructions.

Kdebug can run on a VGA console (that is not running X) or on a serial port (independent of the system console settings, although by default kdebug runs on the system console). Please refer to the `kdebug`(4) man page for details on configuration and usage.

When kdebug is included in a kernel, it is entered after a panic automatically -- the system will not automatically reboot in this mode (although the `cont` command may be issued to continue the panic sequence, write a dump, and reboot).

Tracing support

A small (8K) wraparound trace buffer facility is available, even in the normal non-debug GENERIC kernel. Trace points are scattered around the kernel in various critical locations; each trace point simply allocates the next slot in the trace buffer and writes the CPU cycle counter (as a timestamp), a pointer to an ASCII string, and up to 5 integer arguments. When the trace buffer is displayed (with `kdebug`(4) or `tdump`(8)) the strings are passed to a function that formats them in a manner similar to printf() (passing the formatter the 5 arguments in the trace slot entry).

The buffer wraps around on itself, so the contents of the buffer are the most recent 8K worth of trace entries (each trace entry is 32 bytes). The overhead at a given trace point is very low (a small handful of instructions) so this facility can be used to debug timing sensitive problems.

Trace points can be enabled or disabled at run time by bits in the global variable *ktr_mask*, each trace point is a member of a class and each bit in ktr_mask determines which trace classes are enabled. The cost of a runtime disabled trace is a memory read and a taken branch.

Trace points can be left out of the kernel at compile time by defining the *KTR_COMPILE* option, it is a mask of class bits (similar to ktr_mask), classes that are not enabled in KTR_COMPILE are not included in the generated kernel (0 overhead).

Please refer to the `ktr`(4), `tdump`(8), and `kdebug`(4) man pages for details on the use of this facility.

Hung systems

If you are comfortable switching kernels and poking around inside your machine an additional mechanism is available to help debug system hangs. When the SERR signal on the ISA bus is grounded most motherboards generate an NMI (non maskable interrupt) to the CPU. If the kernel debugger (KDEBUG) is installed in the kernel, it will intercept the NMI and start the kernel debugger.

ISA cards with a pushbutton for the SERR signal are available, in a pinch a paperclip or small screwdriver may be used to short the two rearmost (toward the back of the machine, the end that has the openings for the ISA slots) ISA pins together; these are the last two pins that are directly across from each other at the back of the ISA/EISA slot. Be *sure* you understand which pins to short and are comfortable with the procedure; if you short the wrong pins you may damage your motherboard.

Not all motherboards route or enable NMI from the SERR signal on the ISA bus, when debugging a problem it is worth testing this feature by attempting to enter the debugger after booting to single user mode.

Once in the kernel debugger the trace buffer and kernel stack may be dumped to provide additional information useful in debugging a hang.

A generic kernel with kdebug installed is available in `/diag/bsd.debug` on the release CD; an up to date kernel with debugging enabled can be found on the BSDI web site.

Other Tools

The CD-ROM contains a few informal tools for testing systems. If you call in a support request, we may ask you to run one or more of these tests. Most of the tests are run standalone or in single-user mode, and some of them may cause hangs or other problems on sensitive hardware. If your CD-ROM is mounted on `/cdrom`, then you can find the diagnostics as follows:

`/cdrom/diag/bin/checkio`: Scans the I/O memory space looking for BIOS ROMs, display memory and other memory areas. If your system suffers from I/O resource conflicts, this program can identify cards in your system that BSD/OS failed to detect. This program runs in single-user mode.

`/cdrom/diag/bin/memtest`: Tests physical memory. This is a standalone program; you run it by copying an image to a floppy and booting the floppy.

`/cdrom/diag/bin/mtest`: Tests virtual memory. This test looks for problems in your virtual memory subsystem, including physical memory and paging I/O.

See the manual pages in `/cdrom/diag/man` for more details.

Source Code

Systems with source code have access to wide range of debugging tools that are not available with object-only distributions. These tools are normally used only by experienced system programmers and are oriented toward kernel developers.

The DEBUG option turns on extensive (and somewhat expensive) kernel consistency checking. To enable it, simply uncomment or add the '`options DEBUG`' line in your kernel config file, then do '***`make clean`***' in your build directory before building a new kernel. Most of the DEBUG tests will cause your system to crash with a 'panic' message when a consistency test fails; these messages, along with the crash context, can help to isolate many kinds of software and hardware problems. You can add '`#ifdef DEBUG`' directives to your own kernel code to provide your own optional consistency checks.

The `kdebug`(4) facility provides a very limited kernel debugger on the console. The primary advantage of this debugger is that it can use the CPU's hardware debug breakpoint registers to watch for writes to particular addresses or I/O locations, without substantial performance loss. It may also be useful for quick-and-dirty debugging when a second machine isn't available for debugging with remote `gdb`. See the `kdebug`(4) man page for details on this facility.

The `gdb`(1) debugger in BSD/OS may be used to perform source code debugging of kernels. You enable the kernel debugging feature with the `-k` flag. You may debug local kernels, crash dumps or remote kernels using serial ports (COM ports) for communication. Breakpoint debugging only works with remote kernels.

```
Local kernels:  # gdb -q -k bsd.gdb /dev/mem
Crash dumps:    # gdb -q -k bsd.gdb /var/crash/bsdcore.N
Remote kernels: # gdb -q -k bsd.gdb /dev/ttyNN

    (gdb) target remote /dev/ttyNN
    Remote debugging using /dev/tty1a3
    kgdb_connect (verbose=0) at ../../i386/i386/kgdb_stub.c:295
    295 if (kgdb_dev >= 0 && kgdb_getc != NULL && kgdb_debug_panic) {
    (gdb) [...]
```

You must configure a kernel specifically for remote debugging with `gdb`; see the comments in `/sys/i386/conf/GENERIC` for details.

The directory `/sys/scripts` contains a number of useful gdb scripts for displaying kernel data structures, especially the `bsdi.tricks` file. Read the comments in the files to see what is available.

See the BSD/OS configuration document for more details on kernel configuration. You may view it with '***man -M bsdi config***'.

SCSI Notes

Device flags

The flags field in the kernel configuration file is no longer used by BSDI supplied SCSI HBA drivers. While HBA drives almost always operate acceptably with the default parameters, the possible combinations of parameters exceeds that which can be specified in 32 bits. Parameters are now passed in with the -parm command to `boot`. This is available from both the command line, and from `/etc/boot.default`.

Disconnect

In prior releases of the OS some disk type targets had problems with disconnect. The work around was to either set a "broken disconnect bit" in the HBA, or disable disconnect. The vast majority of these problems were caused by targets finishing a write operation in a way which prevented the HBA from determining if all the data had been transfered. In general, residual counts have no meaning for block devices, and are now ignored by the upper layer disk driver. This should allow most disk devices which had problems with disconnect to now work without special configuration flags.

Raid

Raid devices which appear as a SCSI target are supported much better. The number of simultaneous operations has been increased in almost all the HBAs. Also the disk driver can now be told that a particular unit is a raid device and to send down more operations in parallel. See the manual page for `sd`.

Tagged queueing

In BSD/OS 4.0, it is possible to read and write large files at full disk-transfer speed through the SCSI subsystem. The tagged queueing feature of SCSI 2 is utilized to achieve this. The firmware in the drive to implement tagged queueing is difficult and it is unfortunately not uncommon to find it broken. To make matters worse, drives with incorrect firmware will typically appear to operate correctly, but will silently read and write incorrect data. For this reason tagged queueing is **not** enabled by default.

There is no way to determine if the tagged queueing feature in a particular drive's firmware is flawed or not, other than by running it and seeing if data corruption occurs. This test is not something that should be done on a system with live data, as it is possible that all data on the system could be lost! In most cases, upgraded firmware for the drive will fix the problems. If you are unsure as to the correctness of the firmware in the drives you are using, contact your drive vendor.

In order to use tagged queueing two changes must be made. First, tagged queueing must be enabled in the HBA. For example, to enable tagged queueing to all drives on the first NCR controller, the following line would be added to `/etc/boot.default`:

```
  -parm ncr0 tags=all
```

See the manual page for each specific host bus adapter for adapter specific details as well as general information.

The second change is to configure the file system with zero as its "`rotdelay`" value. Do this by specifying "`-d 0`" as an argument when initializing the file system with `newfs`(8), or when changing the file system parameters with `tunefs`(8).

Writing SCSI Tapes

Streaming cartridge tape devices have some restrictions on where writes might be initiated. A write can only start at the beginning of the tape (BOT) or after the last data block or file mark (end of data, EOD). Old data following the last written block or filemark is inaccessible; it is impossible to edit a tape, although appending is OK. The SCSI tape driver will automatically space to EOD if the tape is opened for writing and the tape head is not at BOT.

DAT and Exabyte SCSI Drives

Versions of BSD/OS before 2.0 used fixed-length 1024-byte records when reading or writing Exabyte 8mm cartridge tapes. BSD/OS 4.0 defaults to variable-length records, to be compatible with Sun Exabyte tapes. You can read fixed-style tapes in variable-length mode by providing a blocksize of 1024, but this can be very slow; `/dev/rst0_fixed` provides the old fixed-length, record interface and is usually faster with such tapes.

Drives that are detected as DAT (either DDS1 or DDS2) or Exabyte will default to variable-blocking. Several special files are available in `/dev` to force the use of a specific format and/or fixed-blocking on DAT or Exabyte drives. The fixed-blocking entry for the first SCSI Exabyte or DAT tape is `/dev/rst0_fixed`. There are also entries to force DDS or DDS2 selection, `/dev/rst0_dds` and `/dev/rst0_ddsII` for drive 0. These devices default to variable-blocking. The fixed-blocking names to force DDS or DDS2 are `/dev/rst0_fdds` and `/dev/rst0_fddsII`. Each of these names is also available in a non-rewinding version with a leading n, e.g., `/dev/nrst0_ddsII`.

Some drives have to be operated at a fixed record size other than 1024 bytes (the default). The length of the fixed records can be controlled by using the `mt`(1) utility. See the `mt` manual page for details.

Fixed-record SCSI Tapes

The `pax`(1) (or `tar`(1)) program will not read data correctly from some fixed-record SCSI tapes. QIC tapes work fine, as do Exabyte or DAT tapes written with the default (variable-record) selection. 4mm DAT and Exabyte tapes written with fixed 1024-byte records fail. At least some drives will not allow doing a variable length read of 512 bytes when the tape was written with fixed 1024-byte records. In this case the data must be re-blocked to allow reading, for example:

```
  % dd bs=1k if=/dev/rst0 | pax -r
```

Splicing Disk Devices with BSD/OS 4.0

A splice is a term used for multiple physical disk partitions joined into a single unit, which then behaves as if it were a single disk. The partitions to be joined need not be the same size.

Below are the steps to take to set up a splice of two or more disk partitions to act as one device.

1. Make disk devices (if needed) This is in case if you are using say more than 4-SCSI disk drives

   ```
   # cd /dev
   # ./MAKEDEV sd3 sd4 sd5      ---basic stuff
   # ./MAKEDEV sp0              ---Makes Splice device one
   ```

   Note:Depending upon how many splices you want to use, create sp0,sp1,sp2 etc.
2. Edit the exisiting kernel configuration file, uncomment the splice option which is near the end of the file.

   ```
   # pseudo device sp 10           #"splice" stiped
   ```

   Also, if you are using more than two SCSI controller cards [eg. three Buslogics] you can make those changes right here..
3. Rebuild and install the kernel. Reboot the system.
4. Run disksetup on each of the drives you need to use for the splice
5. Now, run the splice command to create the new device.

   ```
   # splice -i 2048 /dev/sd#x /dev/sd#x
   ```

   where # represents the number of the disk and "x" is the name of the partition.

   An example of the splice command would be something like following:

   ```
   # splice -i 2048 /dev/sd1h /dev/sd3h /dev/sd2h /dev/sd4h
   ```
6. Run disksetup on the new splice device.

   ```
   # disksetup -i sp0
   ```

   Tell `disksetup` it is a SCSI disk and use the internal geometry. Define the needed partitions. No boot blocks needed as you cannot boot from the splice device.
7. Run newfs on the partition you created to make a filesystem.

   ```
   # newfs /dev/rsp0a
   ```

   If you like to use different inodes or block sizes, you can add that here just like a normal newfs command.

   At this time do some basic testing and if it all seems to work okay, proceed to the next step.
8. Edit the `/etc/rc.first` file and add the splice command as mentioned in

step 5 above. This will cause the system to recreate the splice device at boot up time before ‘fsck’ is run on the filesystems.

```
splice -i 2048 /dev/sd1h /dev/sd3h /dev/sd2h /dev/sd4h
```

9. Edit your `/etc/fstab` file and create an entry for your new splice device.

```
/dev/sp0a /var/news/spool ufs rw 0 3
```

Multiprocessor Support

BSD/OS 4.0 adds the capability to run systems with more than one CPU. The system supports many Intel-standard MP systems. For most MP-capable systems all that need be done is to add an additional CPU, jumper the motherboard appropriately, create the configuration file and reboot the system. However, some vendor hardware will make the process more challenging. Please refer to the BSDI web page:

```
http://www.bsdi.com/products/internet/mpmb
```

for up to date information on tested systems, CPUs, and motherboards.

As a new feature in BSD/OS, there are some caveats to the support of MP. First, in the current version, only one CPU can run system calls at a time. This means that workloads with substantial user-load processing in more than one process can benefit from additional CPUs. But, workloads with a large percentage of system (kernel) time may not benefit at all. Second, hardware vendors seem to have taken many liberties with the Intel MP spec. Getting some of these systems running has taken some manual tweaking. Although this facility is relatively new and testing on many new MP systems has required such tweaks, once running, most systems have been quite stable and reliable. This feature was updated several times during the BSD/OS 4.0 beta test, and will continue to change over the following months. Finally, systems with more than two CPUs have not yet been tested successfully.

When the system first boots, the BIOS selects the bootstrap CPU. The autoconfiguration process only reports information on that CPU, e.g.:

```
Cpu-0 = Pentium II (300 MHz) ...
```

(The boot processor is often Cpu-0, but this is hardware dependent.) The program `cpu`(8) is then used to extract information about the hardware saved during the bootstrap process, to determine how to run the system in MP mode, and to enable additional CPUs. Normally, as the system comes up in multiuser mode, the command `cpu mp` is run to configure and enable any additional processors. Creating the file `/etc/mp.nostart` will prevent the system from attempting multiprocessor operation.

All you need to do to run in multiprocessor mode is to plug in the new CPU and create the file `/etc/mp.config` (either as an empty file or containomg configuration override directives for the `cpu`(8) program).

For the most part this facility is transparent to the system administrator and its users, the main difference being that more CPU time is available for runnable processes. The percentages shown by `top`(1) and similar utilities are based on a single processor, so in a two processor system a single processor running 100% of the time in user mode will be reported as a 50% user bound system. The administrative interface to the multiprocessor aspects of the system is the `cpu`(8) command, this is used to configure the I/O system, start additional processors, and debug configuration problems.

Definitions

The following definitions may help understanding the following sections.

- **UP**
  Uniprocessor mode. Refers to the state of a system after being booted (as it enters single user mode). In this state the system is in PIC mode with only the boot processor running. A system with only one CPU stays in UP mode through multi-user startup.
- **SMP**
  Symmetric multiprocessing mode. The state of the system after the I/O system has been switched into Symmetric I/O mode and additional CPU's have been started.

  Also sometimes used to refer to a motherboard or machine that is capable of running in SMP mode.
- **PIC mode**
  The mode in which the I/O system runs as 'AT compatible'; PIC refers to the 8259 programmable interrupt controller which handles the prioritization and masking of interrupts.
- **Symmetric I/O mode**
  An I/O mode which dynamically routes interrupts across the APIC bus to the CPUs that are most likely to be able to service them. This mode is a prerequisite to starting CPU's other than the boot processor. This term is interchangeable with 'APIC mode'.
- **APIC**
  Advanced programmable interrupt controller. This device is the bridge from a CPU or I/O device onto the APIC bus.
- **Local APIC**
  The APIC built into a CPU that allows it to receive interrupts from the APIC bus and to send interprocessor interrupts. Each CPU in an SMP system must have a local APIC.
- **I/O APIC**
  An external peripheral IC that is used to route interrupt signals from peripherals onto the APIC bus for delivery to CPU local APICs. There must be at least one I/O APIC in an SMP system.
- **APIC bus**
  A serial bus (usually built into the motherboard) which interconnects all CPU local APIC's and the I/O APIC's. There can be 14 devices on an APIC bus (any combination of local and I/O APIC's is supported).
- **Intel MPS**
  Intel Multiprocessor Spec (version 1.4). This is a standard that defines how SMP motherboard vendors must build systems using a combination of local APIC's, I/O APIC's and special BIOS API's. It also defines the ways in which interrupt signals may be interconnected and how to transition a system from PIC mode to Symmetric I/O mode. For the most part a system built to the Intel MPS 1.4 will be compatible with BSD/OS. As with any standard, it is possible to meet the letter of the spec and still produce an unusable or

difficult to configure system.

Configuration

BSD/OS will work on an SMP capable system without any modifications to the GENERIC configuration. For custom configurations, the cpu pseudo device must be defined to use SMP features:

```
pseudo-device   cpu
```

The CPU driver may be removed for UP only systems, this will reduce the kernel size by approximately 3 pages (12k).

Hardware Requirements

To use BSD/OS on a multiple CPU system, the following is required:

- A motherboard capable of multiprocessor operation (more than 1 CPU socket)
- Motherboard hardware and BIOS must comply with Intel Multiprocessor specification (Intel MPS), version 1.1 or 1.4. If given an option, choose version 1.4.
- CPU's must be Intel Pentium, Pentium Pro, or Pentium II
- All CPU's must run at the same clock speed
- For Dual Processor Pentium (and Pentium MMX) systems all processors must be at the same stepping level. It is suggested that same stepping CPU's be used in Pentium Pro, Pentium II, and Pentium Xeon systems, however this is not a hard requirement.
- Each CPU must have a local APIC, this is optional on P54 CPU's - do not assume that a given CPU has an APIC (and always order them with the APIC, usually sold as 'MP capable').

To assist in determining if a given CPU is capable of SMP operation, the `cpu`(8) program may be run with the 'check' option, this will check if the CPU on which it is running is capable of SMP operation. The '***`cpu -v check`***' command will display detailed information on the CPU, including the stepping information. The '**`cpu check`**' command can be run on a system without SMP support in the kernel, and without root access. There is no way for '**`cpu check`**' to check the second processor in a system prior to that processor being started. To check a CPU it must be the boot processor.

Some UP motherboards may erroneously report a CPU as non-MP capable if they are designed with a particular signal to the CPU left inactive. It is best to test CPU's for MP functionality on an MP capable motherboard with all jumpers set for MP operation (any other configuration may produce a false negative).

I/O Modes

The I/O system can run in two modes: PIC mode, and symmetric I/O mode.

PIC mode is the standard AT compatible mode that all motherboards boot with; BSD/OS runs in PIC mode until a command is issued to switch to symmetric I/O mode. In PIC mode the two 8259 compatible interrupt controllers are attached to the 16 ISA IRQs, PCI and EISA interrupts are transparently routed by the BIOS to ISA IRQs.

Symmetric I/O mode is defined by the Intel MPS. In this mode the 8259 compatible PICs are disabled and interrupts are instead routed to one or more I/O APIC's on the motherboard. The I/O APIC's then determine which CPU to route the interrupt to based on the state the CPU is in. BSD/OS programs the hardware to route interrupts to the CPU's that are holding the lock needed to handle interrupts. A system may run in symmetric I/O mode with only one processor, this is the first step in debugging problems with an SMP system (switch to symmetric I/O mode, but do not start additional processors).

In symmetric I/O mode up to 30 blockable interrupt sources are supported by the kernel. Many chipsets route the PCI interrupts to different interrupt pins than the ISA IRQs they were mapped to in PIC mode. At present this does not allow one to use more IRQs than the usual ISA complement since the system still runs autoconfiguration from PIC mode and all interrupts must be assigned then; in the future loadable device drivers will make it possible to boot in PIC mode with a bare bones configuration, then load PCI device drivers (which will not require ISA IRQ's on some motherboards) after switching to symmetric I/O mode.

Another mode (that is really part of symmetric I/O mode) is 'mixed mode'. Some Intel chipsets do not route all interrupt signals to the I/O APIC after switching to symmetric I/O mode, in this case the system must run in both PIC mode and symmetric I/O mode simultaneously. This is less than optimal since any interrupts delivered via PIC mode (usually the clock interrupt, IRQ0) are routed to the boot processor only, regardless of what locks it is holding (this may require the interrupt to be queued and serviced by another processor, adding to overhead).

Basic Usage

The `cpu`(8) command is used to bring the system into symmetric I/O mode and start additional CPU's. For an Intel MPS compliant system, the `cpu`(8) command will automatically configure the system and start additional CPU's as needed by simply issuing the command:

```
cpu mp
```

This is normally run by the system startup scripts if the file `/etc/mp.config` is present (it may be empty).

The '`cpu stat`' command may be used at any time to determine the state of the system (and display some interrupt related statistics):

For example, a uniprocessor system running in PIC mode might report:

```
System running in 8259 PIC mode (AT compatible)
1 cpus are available for process scheduling
Cpu-0: Running
```

```
Interrupt counts (only non-zero shown)
Count        APIC Pin IRQ Type             Description
============ ==== === === ================ ===========
     8648944  0    0   0  PIC              IRQ0 (clock)
        1324  0    0   4  PIC              IRQ4 (com0)
       23986  0    0   a  PIC              IRQ10
       14899  0    0   e  PIC              IRQ14
```

or an SMP system with 2 running CPU's might report:

```
System running in APIC (symmetric I/O) mode
2 cpus are available for process scheduling
Cpu-0: Running
Cpu-1: Running

Interrupt counts (only non-zero shown)
Count      APIC Pin IRQ Type           Description
========== ==== === === ============== ===========
       769  2    2   0  APIC Clock     IRQ0 (ISA/EISA source)
        87  2    4   4  APIC Edge      IRQ4 (ISA/EISA source)
         3  2    a   a  APIC Level     IRQ10 (ISA/EISA source)
        43  2    e   e  APIC Edge      IRQ14 (ISA/EISA source)
```

Advanced Usage

The `cpu`(8) command can be used to incrementally bring the system into SMP mode. Instead of using the '`cpu mp`' command, a more selective approach can be used:

- Switch the system into symmetric I/O mode (cpu siomode)
- Start specific processors as desired (cpu start N)

The '`cpu stat`' command can be used to determine how interrupts are configured before and after switching to symmetric I/O mode.

Debugging

The Intel MPS leaves some room for interpretation, and some vendors seem to go to great lengths to make things difficult on the operating system.

Initial problems usually crop up when trying to switch to symmetric I/O mode. If possible, test symmetric I/O mode on a given motherboard or system before committing to using it as a multiprocessor system; symmetric I/O mode can be entered with '`cpu siomode`', even on a single processor system.

The two main problems that occur going to symmetric I/O mode are:

1. The BIOS does not provide an accurate description of the SMP hardware facilities, usually this is in the area of the mapping of PCI interrupt sources (A/B/C/D) to PCI agents (some motherboards simply declare that 4 PCI interrupts exist and do not specify which devices are on which interrupts).

   Solution: The cpu command can be told how to route particular interrupts via `/etc/mp.config` (or command line options). See below on some tips for `cpu`(8) usage.

2. The hardware does not route one or more interrupts to the I/O APIC. A good example of this is the Intel PIIX3 and PIIX4 chips (used on many motherboards). The clock interrupt is not available outside of the PIIX3 chip, and thus cannot be connected to the I/O APIC; some motherboard vendors, in a quest to retain bug compatibility, do not connect the available IRQ0 signal on the PIIX4 chip to the I/O APIC. There are a number of ways to deal with this problem; unfortunately the Intel MPS does not allow the BIOS to provide enough information to automatically configure around this problem in some cases.

Generally, if a system will not go into symmetric I/O mode then the SMP compatibility web page should be checked for parameters to put into `/etc/mp.config`. Failing that, support should be contacted to help debug the problem. The general method used to debug such a configuration problem is:

1) Inspect the MPS configuration records from the BIOS (cpu bootparam)
2) Run '`cpu -d siomode`' (debugging mode) to determine what configuration decisions are being made based on the BIOS data given.
3) Determine which interrupts are not being configured correctly and create configuration overrides (for `/etc/mp.config`) to fix them.
4) For systems with mapping problems on the PCI bus (where the BIOS does not tell which interrupts are connected to which devices) it may be necessary to use trial and error to determine which of the 4 PCI interrupt inputs map to particular IRQs.
5) For systems which must run in mixed mode the interrupts that are not actually connected to the I/O APIC must be determined and overrides developed that describe their routing through the PIC. The configuration may also need to be modified to describe how the PIC chains to the I/O APIC.

When working with support the first thing that will be asked for is the output from the '`cpu save` command on the system being debugged. This is uuencoded data gathered from the target system that can be used with `cpu` on a remote machine to do most of the above steps without access to the machine.

Once symmetric I/O mode is working, the problem areas tend to be either in starting additional CPU's or in running reliably after additional CPUs have been started.

Problems starting additional processors are usually due to the BIOS not describing the addresses of the additional CPUs correctly. As with symmetric I/O mode these problems can be overcome by supplying overrides in `/etc/mp.config`. Before getting into overrides, specific CPUs can be started with the '`cpu start`' command (instead of letting '`cpu mp`' try to figure out which CPUs are present from the BIOS data).

An additional problem is instability with multiple CPUs running. Some motherboards may be more sensitive to marginal RAM or cache modules with 2 or more CPUs running. Hangs or data corruption should be prosecuted as for a UP system. The most common problems seen to date have been due to mismatched CPU steppings and marginal DRAM (always use parity or ECC if possible).

Internals

The rest of this chapter is primarily of interest to kernel developers.

Interrupt Handling

The low level interrupt handling code has been changed quite significantly to support SMP systems. There are now two modes of I/O, PIC mode and symmetric I/O mode (or APIC mode).

PIC Mode

PIC mode is the familiar AT compatible system where there are two 8259 (compatible) programmable interrupt controllers (PIC's) that are cascaded in a master/slave relationship. The master is programmed with an interrupt vector base (0x20) and issues interrupt requests to the CPU via a hardware pin, the CPU responds with an INTA cycle and the 8259 then sends the vector number of the interrupt to the processor which interrupts execution and starts executing code as described by the IDT table entry corresponding to the vector number.

In Intel MPS systems PIC mode can be implemented in various software transparent ways, some of which use the APIC hardware. These setups must be strictly AT compatible. When first booted, BSD/OS assumes it is running on standard AT hardware and configures the system for PIC mode.

Earlier releases of BSD/OS had hardwired knowledge of which IDT vectors would be used for which IRQs and the method used to mask specific interrupts. At the hardware level, interrupts can be masked two ways: by blocking all interrupts at the CPU (cli/sti), or by configuring the PICs to mask certain of their input pins. Previous implementations of `splXXX`() functions performed the latter. They would read the current mask word (16 bits, 8 from each PIC), logical-or in the new interrupts to be masked, then rewrite the mask word to the PIC. The `splx`() function would simply write the saved mask state to the PIC mask registers.

On older PC hardware the operation of read and writing the mask bits to the PICs was relatively cheap, on faster machines this operation can take quite long (in terms of clock cycles). This mechanism also introduced a race condition when blocking interrupts on the slave PIC that caused some extra overhead handling 'stray' interrupts.

BSD/OS 4.0 handles PIC mode interrupts in a slightly different manner. The `splXXX`() functions update a word in memory (cpl) that indicates which interrupt sources are logically blocked. When any interrupt occurs, the low level assembly code that first gets control checks if the interrupt is logically blocked. If it is, then the interrupt is queued (in the ipending variable), it is blocked at the PIC, and the interrupt handler returns immediately. When the interrupt is unblocked (either due to an `splx`() or return from an interrupt handler) it is delivered to the interrupt handler as before.

This new scheme has the advantage of making `splXXX`() and `splx`() calls quite inexpensive (although `splx`() now has to check for queued interrupts, which is some additional work). `splx`() will not reprogram the PIC mask registers unless it detects that the interrupt was physically blocked at the PIC. So, the typical `splXXX`()->`splx`() sequence involves only a few operations on main memory (which are cached and quite fast).

In PIC mode there is a limitation of 16 interrupt sources (IRQ0 through IRQ15). The additional interrupt lines coming from the PCI slots (usually 4 lines, marked A through D) are typically routed by the BIOS and motherboard to override certain ISA IRQs.

APIC Mode

There is no analog to APIC (or symmetric I/O) mode on earlier releases of BSD/OS. In APIC mode, an interrupt line from a peripheral device is connected to an I/O APIC. The I/O APIC is in turn connected to a shared APIC bus which connects all the CPU's and I/O APIC's together. Each device on the APIC bus has a unique ID (from 0 through 14). This ID (in hex) is used as the CPU identifier when describing CPU's (ex: Cpu-c is the CPU who's local APIC is at ID 12, or 0xc).

When an interrupt is signalled at the I/O APIC, it determines which CPU is the best candidate to receive it (using priority information supplied by each CPU) and sends a message to that CPU's local APIC containing the vector number programmed for the interrupt pin. In BSD/OS, the CPU priorities are programmed, based on which locks are being held by each CPU. Priority is given to the CPU holding the lock required to process interrupts, with the next highest priority being an idle CPU, and the lowest priority being a CPU running user code.

When the CPU receives an interrupt message on the APIC bus it uses the vector provided by the I/O APIC (an 8 bit quantity) to index into the IDT table (the same one used in PIC mode) and deliver the interrupt to the appropriate handler.

Unlike PIC mode, interrupt priorities are determined based on the interrupt vector number, the higher the vector the higher the priority. While the interrupt is 'in service' no interrupts of a lower priority are accepted by that CPU. Another complication is that the implementation of the local APIC in the CPU requires that no more than 2 interrupts be delivered to a priority band at a time (a priority band is a group of IDT vectors that all have the same upper 4 bits, 0x20 through 0x2f, for example).

To handle the more complex environment of symmetric I/O mode BSD/OS switches to an IDT of 256 entries (the maximum size of an IDT) and assigns interrupts sparsely to priority bands based on the relative priority of the class of the device (clock tick is highest, disk is lowest). The algorithm used attempts to assign no more than 2 interrupt sources to a priority band, in most cases this is not a problem.

In symmetric I/O mode some motherboards break the association between PCI interrupt sources and ISA IRQs that were set up at boot time by the BIOS. For these motherboards there are more than 16 interrupt sources (typically there end up being 20 sources, 16 ISA plus 4 PCI). BSD/OS supports up to 30 unique interrupt sources (this limitation is based on the size of the word returned by the `splXXX()` functions). For interrupt sources that do not need to be blocked by the spl family of functions (such as interprocessor interrupts) there is no limit other than the 256 entry size of the IDT.

In APIC mode interrupt blocking is not fully supported by the hardware. Each interrupt pin of an I/O APIC may be disabled, however a disabled pin will not latch an edge triggered interrupt. Furthermore, several bus transactions are required to disable or enable each pin, there is no central mask word that can be written to block groups of pins.

The designers of the I/O APIC clearly intended that the priority system be used to block delivery of lower priority interrupts while handling higher priority interrupts or tasks, however this requires a strict hierarchical interrupt priority model which BSD/OS does not use. (We group interrupts by device class and block the classes individually.)

To work around the limitations of the APIC hardware the notion of interrupts being logically blocked (as described above in PIC mode) is used. This allows `splXXX()`/`splx()` calls to be made without incurring the high penalty of blocking/unblocking multiple interrupt pins. Since edge triggered interrupts will generally not fire again until the device has been serviced, the APIC edge mode interrupt handlers do not even bother to disable the interrupt when it is received. The logical blocking mechanism prevents recursion in the infrequent case where this does happen.

Two additional complications are worth mentioning: EOI's and level triggered interrupts. When the local APIC delivers an interrupt to its CPU it raises the CPU's priority to that of the interrupt vector's priority band (the upper 4 bits of the interrupt vector number). The interrupt handler must acknowledge the interrupt by issuing an EOI command (end of interrupt) when it completes. Since the priority mechanism isn't being used, BSD/OS issues an EOI command as soon as possible in the interrupt handler entry code, this allows 'lower priority' interrupts to be serviced while the interrupt handler is running. (In this case, lower priority is from the APIC's point of view - an interrupt at a lower IDT vector may actually be unblocked, and thus higher priority than the interrupt being serviced.)

This use of EOI, which returns the processor to its 'natural' priority (based on the locks held and user/idle mode) as soon as possible, works fine except in the case of level triggered interrupts. A level triggered interrupt will re-fire immediately after the EOI command has been issued, not giving the interrupt handler a chance to run the interrupt service routine of the device driver and clear the interrupt. To work around this problem, level triggered interrupts are disabled in the hardware when they are received (before issuing the EOI command). This does not result in lost interrupts since a level triggered interrupt will fire when re-enabled.

Mixed Mode

Mixed mode is APIC mode with some interrupts still serviced in PIC mode. This is needed to deal with less than ideal chipsets that do not actually connect all the ISA interrupts to the I/O APIC when switching to symmetric I/O mode (such as the Intel PIIX3). PIC interrupts delivered in mixed mode are delivered via a special delivery mode on the I/O APIC called 'ExtINT' mode. The master PIC interrupt request line is connected to a pin configured for ExtINT mode on the I/O apic. When the PIC indicates an interrupt the I/O APIC pretends to be a CPU and issues the interrupt acknowledge handshake with the PICs. It then takes the vector given to it by the PIC's and transmits it to one of the CPU's on the APIC bus (fixed address, the boot processor in BSD/OS).

Interrupts delivered through PIC mode do not participate in the interrupt priority system and are EOI'ed at the 8259 PIC instead of the local APIC.

Interrupt Handler Stubs

When running in APIC mode each IDT entry is pointed at a short stub routine that is dynamically allocated for use by that IDT entry alone. The stub saves the general purpose registers on the kernel stack and then loads identifying information into %esi and %edi and jumps to the main interrupt handler.

There several interrupt handlers that are used depending on the needs of the interrupt source:

- APIC edge triggered source
- APIC level triggered source
- Mixed mode PIC edge interrupt
- APIC clock interrupt (IRQ0)
- Clock interrupt via mixed mode PIC source
- Unknown interrupts (unexpected on vector)
- PIC mode interrupt

Configuration

To deal with the numerous configuration possibilities, interrupt routing is primarily table driven at the kernel level. Each interrupt source has an entry in a global kernel table named 'inin' (interrupt information). The indices of the first 30 entries in this table correspond to bits in the cpl and ipending words (and thus can be blocked by `splXXX()` functions). The entries in the inin table contain information used to deliver an interrupt to interrupt service routes (such as the IRQ number), interrupt counters, debugging information, and masks determining which other interrupts must be blocked while handling a particular interrupt.

When the system boots into UP mode, the inin table is set up with a mapping that approximates the old interrupt blocking mask behavior. Entry 0 corresponds to bit 0 in cpl, and describes IRQ0. When symmetric I/O mode is entered the entries in the inin table are reprogrammed by the `cpu`(8) program to describe the routing of each interrupt source (remember there may be more than 16 of them now) to a particular IRQ number (and ultimately an interrupt handler). Entry 3 in the inin table may no longer be related to IRQ3 after entry to symmetric I/O mode. This extra level of indirection is needed to handle cases such as when all 4 PCI interrupt sources were mapped by the BIOS to a single IRQ, after symmetric I/O mode is entered there are now 4 independent interrupt pins that all map to the same IRQ yet are blocked and unblocked independently (since they are PCI interrupts and level triggered, they must be blocked when received).

When the system is taken into symmetric I/O mode, the entries in the inin table are set up by the `cpu`(8) program (based on Intel MPS BIOS information and overrides from `/etc/mp.config`). The information in these entries is used by the kernel to allocate the stubs, connect IDT entries to them, and route them to the appropriate handlers. The handlers in turn use information in the inin table to determine how the interrupt should be handled and in some cases how to block the interrupt.

Interprocessor Interrupts

Interprocessor interrupts are originated in a given CPU's local APIC, transmitted across the APIC bus to the target CPU APIC, and then delivered by it to its local CPU core. Interprocessor interrupts carry a vector number (supplied by the sending CPU's software) with them, this vector is allocated from the same vector space as I/O interrupts.

The interprocessor interrupts are assigned to the highest priority band in the IDT vector table (0xf7 through 0xfd). The `splXXX`() functions do not block interprocessor interrupts. However, the CPU interrupt flag (cleared by cli) does. As with I/O interrupts, the first thing done by the various IPI handlers is to issue an EOI to the local APIC to allow other IPI's or I/O interrupts to be delivered.

The IPI's used are:

`IDT_IPI_HALT`
This is used during shutdown and panic situations to forcibly halt its target (quickly). Contrast to IDT_IPI_STOP which is used to gracefully shut down a CPU.

`IDT_IPI_STOP`
If the target is connected to a process, it puts that process back on the run queue and then halts itself.

`IDT_IPI_AST`
This causes a trip through the `trap`() entry point. It is used to force the target to consider delivering a signal to the process it is currently running or to force a processor to switch context to a higher priority process.

`IDT_IPI_PCLOCK`
This is a broadcast IPI that delivers the 10ms clock tick to the processors other than the one the received the actual clock tick from the PIT. It gives the target a chance to update statistics and perform profiling as well as considering switching context to a higher priority process on the run queue.

`IDT_IPI_CLKSYNC`
This is broadcast IPI that is used (in debugging situations only) to synchronize the cycle counters on all the CPU's in a machine.

`IDT_IPI_TLBSHOOTPG`
This IPI is used to invalidate specific entries in a target TLB.

`IDT_IPI_TLBSHOOT`
This causes the translation lookaside buffer of the target to be flushed immediately.

Locks

Numerous locks are used to control access to kernel facilities by multiple CPU's. The current lock is a fairly simple (low overhead) mutex with an owner and reference count to allow recursive acquisition by the same processor. The in-kernel debugger has a command that shows the state of the known locks (d lock) and what code last acquired each of them.

The locking primitives are in `i386/include/mutex.h`. There are assembly and C callable versions. The C versions allow for spin waits as well as trying for the lock and returning failure if we didn't get it.

Briefly, the following locks are used by the kernel to serialize access to critical resources:

- **klock**- Main kernel lock, protects the bulk of the kernel. Most entry points to the kernel (system calls, interrupts, etc.) require this lock.
- **slock**- Switch lock. When a user process makes a system call but the kernel lock is not available, it will attempt to run another process that may be on the run queue and is ready to return to user mode. This lock is used to protect parts of the run queue that do not require the kernel lock.
- **ipend_lock**- Interrupt pending queue lock, protects write access to interrupt pending word, the hwcpl word, and the I/O APIC hardware.
- **kdebug_lock**- Kernel debugger console lock. Used to serialize access to its console when running on multiple CPUs at the same time.

Debugging Facilities

Two facilities have been added to aid in debugging multiprocessor systems at the kernel level. There is greater detail on these facilities in the chapter *Kernel Debugging*.

The first is a wrap around trace buffer that is kept per processor. This trace buffer can show what circumstances occurred prior to a problem that caused a panic or hang, the traces are very low overhead and many of them are in the default GENERIC kernel. See `ktr`(4) and `tdump`(8) for details.

A built in command line driver debugger is also provided, its primary purpose is to allow poking around on an otherwise hung system and decoding the wraparound trace buffers. It also allows use of the CPU provided hardware debug registers to breakpoint on memory, I/O, or instruction accesses. See `kdebug`(4) for details.

Clock Issues

Security features introduced in BSD/OS 3.0, or the need to run DOS/Windows on the same machine may be making it difficult for you to keep your clock set properly. If you are having problems with the date and/or time on your system, this chapter should answer your questions.

Unable to Set Time Backwards

There is a new security feature in BSD/OS 4.0 that makes it impossible to set the time backwards when running in secure mode (see `init`(8) for further information on running in secure mode). This feature prevents malicious outsiders who have broken system security from covering their tracks by tricking other machines into believing that outdated security credentials are still valid.

The `date`(1) program will report an error if you try to set the time backwards and the kernel security level is greater than zero. It is still possible to set the time forward using `date`(1), of course. Daylight savings changes are not affected by this security feature, as the system's clock does not actually change for daylight savings; the system merely displays a different value for the time of day.

If you use NTP (see *Configuring NTP*), BSD/OS will automatically fix your clock at boot time, before the security level is raised during the switch to multi-user. The `/etc/rc` script will read a list of servers and peers from the `/etc/ntp.conf` file and will use `ntpdate`(1) to contact them and set the time correctly. After setting the time, the `xntpd`(8) program will keep your clock synchronized over the network. The `xntpd` program controls the speed of the system's clock, slowing it down or speeding it up to account for inaccuracies.

If you do not use NTP, you must set the clock backwards while you are in single-user mode.

DOS time is incorrect

BSD/OS normally stores the computer's time in GMT in the CMOS (hardware) clock, which means that DOS time could show up several hours different from your local timezone. You can change this by reconfiguring the kernel. In the kernel config file change the "`timezone` ***HH***" line ("`timezone 0`" by default) to specify the number of hours west of Greenwich (GMT). If you are not using DOS (or don't care about having DOS show GMT time) then you do not need to change this.

Alternatively, use

```
  # bpatch tz 480
```

(or other appropriate number of minutes) to set the difference between the BSDI clock and the hardware clock (without reconfiguring the kernel). You must reboot for changes made via `bpatch` to take effect.

DOS Time Is Set To 12:00

You need to update your boot blocks to those in BSD/OS 4.0. Make sure you have the latest boot blocks in `/usr/bootstraps` and execute

```
 # disksetup -B wd0    sd0 for a SCSI disk
```

Distribution Media Format

Installation CD-ROM Media Format

The CD-ROMs shipped with BSD/OS are mastered as ISO 9660 images and use the Rock Ridge extensions for encoding long filenames and other POSIX semantics The CD-ROMs also include `.MAP` files which can be used to reconstruct the Rock Ridge information.

The CD-ROMs shipped with BSD/OS contain the BSD/OS software in both a compressed archive format as well the extracted images. The compressed archives are found in the directory `/cdrom/PACKAGES`. The `/cdrom/PACKAGES/PACKAGES` file on the CD-ROM contains the manifest. The file is flat-text, and has comments at the top to describe the fields.

The commands for extracting a normal archive are:

```
% pax -rvp e < file
```

The commands for extracting a compressed archive (type "z" in the manifest) are:

```
% gzcat file | pax -rvp e
```

Installation Floppy Filesystem Format

The BSD/OS 4.0 INSTALL floppy has a compressed filesystem with most of the files needed to load BSD/OS. These files are decompressed and loaded into "RAM Disk" (memory) which is then used as the root device while installing BSD/OS. Using the RAM Disk provides four benefits:

1 A significant increase in the amount of items that can be placed on the root filesystem (for instance, the shared C library is now compressed by virtue of the entire filesystem being compressed).
2 Once the RAM Disk is loaded, is is much much faster than a floppy based filesystem.
3 The RAM Disk is mounted read/write and so new files can be created.
4 There are no self-extracting binaries as the entire filesystem is compressed.

The files found on the INSTALL floppy are as follows:

File	Description
/boot	The bootstrap loader that loads the kernel and the RAM Disk.
/bsd.gz	A compressed image of the generic BSD/OS 4.0 kernel.
/filesys.gz	A compressed image of the RAM Disk based root filesystem.
/etc/boot.default	Instructions on how to load the kernel and RAM Disk based filesystem.
/scripts/install	The upgrade procedure (initiated by the installfloppy command)

The important commands found in the /etc/boot.default file are as follows:

Command	Description
-ramdisk 2560K	Reserve a 2.5 MB RAM Disk
-ramdiskimage filesys.gz	Preload the RAM Disk with filesys.gz
-rootdev rd(0)	Use the RAM Disk as the root filesystem
-load bsd.gz -	Load the kernel (and RAM Disk)
-start -	Start the kernel just loaded

To view the contents of the root filesystem under BSD/OS 4.0, use the following commands:

```
# mount -r /dev/fd0a /a
# gzcat /a/filesys.gz > filesys
# umount /a
# vndconfig filesys
# mount /dev/vnd0a /mnt
```

The filesystem is now mounted on /mnt. If you make any changes under /mnt, those changes will be reflected in the filesys file. Use the following commands to unmount the filesystem:

```
# umount /mnt
# vndconfig -f -c
```

To place the modified filesystem on an INSTALL like floppy, use the commands:

```
# mount /dev/fd0a /a
# gzip --best < filesys > /a/filesys.gz
# umount /a
```

If you modify the contents of the floppy, you might leave the “free” blocks dirty and the compressed image might be larger than needed (and hence not fit on the floppy disk). To alleviate this problem you can execute the following commands prior to unmounting the filesystem:

```
# dd if=/dev/zero of=/mnt/zero
[ dd should eventually fail due to lack of space ]
# rm /mnt/zero
```

This will fill all the free blocks of the filesystem with zeros and allow more efficient compression of the filesystem.

When we build the INSTALL floppy, we always start with a fresh image by using the following commands:

```
# dd if=/dev/zero of=filesys count=5120
# vndconfig filesys
# disksetup -R /dev/rvnd0c filesys.label
# newfs -c 10 /dev/rvnd0a
# mount -o async /dev/vnd0a /mnt
...
```

To produce a copy of the filesys.label file, execute the following command:

```
# disksetup -E ./filesys > filesys.label
```

Note, `filesys` is the uncompressed version of `filesys.gz` from the INSTALL floppy.

Since the BSD/OS 4.0 INSTALL floppy actually runs from a RAM Disk, there is no need to leave the INSTALL floppy in the floppy drive once the system has booted. This allows the mounting of a second floppy with more commands:

```
# mount -r /dev/fd0a /a
```

or the CD-ROM itself, if it is local. This can be a useful method of disaster recovery.

Disaster Recovery

Catastrophic Error Recovery

In the event that your system has a catastrophic error and you are unable to boot, you should reboot from the INSTALL floppy and then attempt repairs. If you can then mount the distribution CD-ROM, you will have access to the standard BSD/OS commands.

To prepare for a situation where you might not have access to a CD-ROM (e.g., you're recovering a system without CD-ROM hardware) you should copy the floppy image `/cdrom/FLOPPIES/RECOVERY.img` from your installation CD-ROM to a blank floppy disk and label the disk `BSD/OS 4.0 RECOVERY Floppy`. (See the file `/cdrom/FLOPPIES/README` for copying instructions.) To use this floppy, first boot from the INSTALL floppy. When asked for your license, enter "none". When asked:

```
  Use Express or Custom Installation? [express]
```

enter "recover". You will then be prompted to insert the RECOVERY floppy and press <ENTER>. The setup program will then run the following commands:

```
  mount -r /dev/fd0a /floppy
  /floppy/setup
```

and then exit, leaving you at a single user shell prompt. This will provide more tools to help.

You can customize the RECOVERY floppy by mounting it and removing and/or adding commands to its `/bin` directory. You can also make a custom INSTALL floppy with a copy of your site's customized kernel instead of the generic kernel (either to add support for a device not in the generic kernel, or to make it smaller and faster to load). To change the kernel, you need to create a compressed copy of your kernel and copy it onto a **copy** of the INSTALL floppy. The following command will do this, (presuming that your new floppy is writeable and mounted on `/a`):

```
  # gzip < bsd > /a/bsd.gz
```

Manual Pages

The BSD/OS online manual contains hundreds of pages of documentation, known as manual or “man” pages. Man page references are usually written as follows: `name(1)`, where “name” is the name of the manual page, and the following number is the section of the manual in which this manual page appears. The section need not follow the name of the man page.

Use the `whatis(1)` and `apropos(1)` commands to locate more information about the topic in which you are interested. The `whatis(1)` command prints the header from the manual pages for any given command:

```
% whatis whatis
whatis (1) - describe what a command is
```

The `apropos` command performs a pattern search on the titles of the manual pages for its arguments. For example, to find the floppy formatting program, enter the following command:

```
% apropos floppy
fd (4) - floppy disk driver
fdformat (1) - format floppy disk
installfloppy (8) - install packages from a floppy
```

Once you know the name of the manual page, use the `man(1)` command to display its contents as follows:

```
% man installfloppy
INSTALLFLOPPY(8)  BSD System Manager's Manual  INSTALLFLOPPY(8)

NAME
     installfloppy - install packages from a floppy

SYNOPSIS
     installfloppy [-d dir]

DESCRIPTION
     The installfloppy program mounts a package format floppy ...

[... lots more output ...]
```

If the command name is not unique, you can specify a section number to retrieve the specific manual page in which you are interested. For example, there are two `chmod` manual pages: `chmod(1)` and `chmod(2)`. To display the manual page describing the `chmod` user-level utility, use the command:

```
% man -s 1 chmod
```

or

```
% man chmod
```

Note that `man` will use the first appropriate man page that if finds (in this case from Section 1) when the section number is omitted. The order that sections are searched is described later in this chapter. To display the manual page describing the `chmod` system call, use the command:

```
% man -s 2 chmod
```

The standard section numbers are as follows:

1. The shell level commands useful to all users.
2. System calls (Programming level).
3. C library functions (Programming level).
4. System I/O devices and configuration.
5. System configuration file formats and the like.
6. Games.
7. Miscellaneous information, including troff macros and various handy reference tables.
8. The commands intended for use by system administrators.

Section numbers are specified by using the -s flag to `man`.

Man pages are broken into many "trees" of man pages. Major software subsystems sometimes have their own tree. If no tree is specified, `man` will search all of the default trees. A specific tree may be specified by using the -M flag to `man`.

Subsystems with their own man page trees include:

- **X11**: The X Window System, version 11 manual pages.
- **contrib**: The manual pages for most contributed software.
- **elm**: The manual pages for the Elm mail program.
- **isode**: The manual pages for the ISODE software.
- **local**: The manual pages for locally installed software.
- **mh**: The manual pages for the MH mail system.
- **old**: The manual pages for deprecated software packages.
- **tex**: The manual pages for the TeX text processing package.
- **iv**: The manual pages for the Interviews software.
- **tcl**: The manual pages for the Tk/Tcl software.

Unless a section number (or section name) is specified, the `man` utility will search all of the known locations for manual pages until it finds one that matches the specified name. The command manual pages are checked first (sections 1 and 8), then the system calls and C library functions (sections 2 and 3), followed by the remaining sections (4, 5, 6, and 7).

Each manual page has several standard sections (see the manual pages in the back of this Installation Guide for examples). The major sections include:

- **TITLE**: The command, file format, or other system object described by the manual page.
- **NAME**: The name of the command and summary of its purpose.
- **SYNOPSIS**: The command(s) or function(s) described by the manual page, and their options or calling conventions.
- **DESCRIPTION**: A complete explication of the command or function.
- **SEE ALSO**: References to related material.
- **BUGS**: A list of ways that the command or function might behave in an unexpected or incorrect manner.

The manual pages that you might want as part of BSD/OS installation are included at the end of this Installation Guide.

Other Online Documentation

A tremendous amount of other system documentation is available online. The `/usr/share/doc/bsdi` directory contains BSD/OS documentation. The file `config.n` describes kernel building and configuration in detail. The file `driver.n` gives hints on writing device drivers. These files are also available via the `man` command:

```
% man -M bsdi config
% man -M bsdi driver
```

In addition:

`Sendmail`(8) documentation can be found in the directory `/usr/share/doc/sendmail`.

Programmer's Supplementary Documentation, can be found in the directory `/usr/share/doc/psd`, including *The Berkeley Software Architecture Manual (4.4BSD Edition)*, *The Introductory 4.4BSD IPC Tutorial*, *The Advanced 4.4BSD IPC Tutorial* and documentation on `gdb`(1), `make`(1), `rcs`(1), `m4`(1), `gprof`(1), and `curses`(3).

User's Supplementary Documentation, can be found in the directory `/usr/share/doc/usd`, including documentation on `csh`(1), `ex/vi`(1), `mail`(1), `rogue`(6), `me`(7), `ms`(7) and `trek`(7).

System Manager's Supplementary Documentation, can be found in the directory `/usr/share/doc/smm`, including *The Berkeley Fast Filesystem*, *The 4.4BSD NFS Implementation*, *The Networking Implementation Notes (4.4BSD Edition)*, and documentation on `perl`(1), `quota`(1), `fsck`(8), `lpd`(8), `timed`(8),

The `info`(1) command examines GNU info documents that live in `/usr/contrib/info/*`. Running `info` with no arguments starts the info system interface at the index.

Further Documentation

Filesystem Hierarchy

See the manual page `hier`(7) for a description of the filesystem hierarchy.

Building Kernels on BSD/OS

See `/usr/share/doc/bsdi/config.n` for a detailed guide to rebuilding and reconfiguring your BSD/OS 4.0 kernel. You can also access this document by `man -M bsdi config`.

Device Drivers and Autoconfiguration in BSD/OS

Found in `/usr/share/doc/bsdi/driver.n`, a detailed guide to writing device drivers for your BSD/OS 4.0 kernel. You can also access this document by `man -M bsdi driver`.

Books

Many publishers offer books that are applicable to BSD/OS. In particular, many of the books in the *Nutshell* series by O'Reilly & Associates are useful. For further information, contact them in the U.S. at 800-338-6887 (from overseas at +1-707-829-0515), via email at nuts@ora.com, or visit their Web site at `http://www.ora.com/catalog/`.

Many other books are available that can help you get the most out of your BSD/OS system. Here are some of our favorites!

Advanced Programming in the UNIX Environment

Stevens, Addison-Wesley. A splendid explanation and rationale for the UNIX programming interface, with lots and lots of examples.

DNS and BIND

Albiz & Liu, O'Reilly & Associates. A detailed reference to DNS and the Berkeley Internet Name Daemon (`named`(8)).

Managing Internet Information Services

Liu et al., O'Reilly & Associates. Covers `ftp`, `telnet`, WWW, Gopher, mailing lists (MajorDomo) and more. A must for anybody thinking of setting up an Internet server. This one is fantastic, and is an excellent follow-on to "TCP/IP Network Administration".

Managing NFS and NIS

Stern, O'Reilly & Associates. While not BSD/OS-specific, a detailed reference to the Network File System and the Network Information System.

Using & Managing UUCP

Ravin, et al., O'Reilly & Associates. The best reference on UUCP installation and configuration.

Sendmail

Costales, et al., O'Reilly & Associates. A tutorial introduction combined with a wealth of detail about modern version of `sendmail` including the Version 8 `sendmail` that is part of BSD/OS 4.0. This book takes the fear and black magic out of `sendmail`, and is an excellent follow-on to "TCP/IP Network Administration".

System Performance Tuning

Loukides, O'Reilly & Associates. While not BSD/OS specific, this book offers useful information on the tools used to monitor system performance and the steps to take when performance is slow.

TCP/IP Network Administration

Hunt, O'Reilly & Associates. A good introduction to the TCP/IP networking suite. Specific sections on troubleshooting, security, DNS, `gated` and `sendmail`.

TCP/IP Illustrated

Stevens, et al., Addison-Wesley. A complete and detailed, multi-volume guide to the TCP/IP protocol suite.

The Design and Implementation of the 4.4BSD Operating System

McKusick, et al., Addison-Wesley. A complete and detailed guide to the design and implementation of the 4.4BSD operating system. The best introduction to BSD/OS internals for the kernel programmer.

The 4.4BSD Manuals

A five volume set from O'Reilly & Associates (currently out of print). These are the 4.4BSD manuals from the University of California, Berkeley. While not as up-to-date as the online manuals, they are the best source for users wanting hardcopy versions of the manual pages and supplementary documents.

UNIX for the Impatient

Abrahams & Larson, Addison-Wesley. A good introduction to Unix-like systems, although not BSD/OS specific.

UNIX System Administration Handbook

Nemeth, et al., Prentice Hall. Arguably the best book ever written about UNIX system administration. This reference includes lots of BSD/OS-specific information.

FAQs & RFCs

Use your favorite WWW search engine to find frequently asked question lists (FAQs) and network RFCs (requests for comments – documentation about every network protocol and operation). Try a search on *www.yahoo.com* for '`sendmail faq`' to get an idea of the availibility of such documentation.

More Online Documentation

The directory `/var/www/docs` has all sorts of documents including hints for apache, ntpd, gated, and hylafax. Read these docs using your WWW browser via URLs like `file://var/www/docs/xntp` (for the xntp directory).

Problem Reporting Procedures

When reporting problems to BSDI, be sure to include your BSDI Customer ID (if you have one) with the support request. It is on your original invoice and is a letter, followed by 4 numbers (possibly followed by an "F" if you are not in the US or Canada); it generally looks like B9999, G9999, or N9999.

If you do not know your Customer ID, then include other distinguishing information (e.g., company name, postal address, telephone number, etc.) so we can look it up for you.

Of course, you should always check the administrative notes, manual pages, and `/usr/share/doc` for information relevant to your problem first.

If you want to submit just a bug report into the bug queue and not a request for support, please use the `sendbug` program which will email to `problem@bsdi.com`. Requests for support should go to `support@bsdi.com` or your vendor's support request address.

When sending bug reports or requests for support:

- Include the OS version you are running and information about any patches you have applied (BSDI official patches and ad hoc patches). If you do not have all of the current BSDI official patches installed, please be sure to check the current patch set carefully to see if something might be relevant to your problem (see *Getting Patches* below). Also, tell us if you are running a source or binary system.
- If multiple systems are involved, please include details about each system. For example, if you are trying to dial into a SUN/OS 4.1.3 system from a BSD/OS system using `uucico`, then it is useful to provide all that information, rather than just saying that `uucico` does not work.
- Explain in detail what you have done to try to solve the problem so far. This will give us important background information about the problem.
- Please send all configuration files that might be related (see below for some tips on that for various kinds of problems).
- Sometimes failures cascade; make sure you catch whatever originally failed and include that in your report as well.
- Include the exact error text as reported by the system; sometimes errors are very similar and without the exact text it can be hard to locate the failure (or worse, it can be misleading).
- Try to avoid making assumptions about the problem. Include all of the basic facts ("I typed this, the system displayed that, here are the configuration files, etc.") and resist the temptation to summarize. The more information we get, the better!
- Include detailed hardware configuration information (`/var/db/dmesg.boot` is a good start but not always complete, so please review it). Many problems we see are IO port or interrupt conflicts.

Common Frustrations

- For network related problems, include the output from `netstat -m`, `netstat -rn`, `netstat -isn`, `netstat -s`, `vmstat -m`, and `ifconfig -a`. In many cases, `tcpdump` can be used to gather helpful information about failing network traffic. The `ping` and `traceroute` programs can also be most helpful.
  Include a description of the network load, e.g., how much PPP, SLIP, etc., traffic, and the names of the devices that are processing the load. Also, include any ideas you might have on correlation of system activity at the time the problem begins can be helpful (it always happens during an `ftp` of a certain file, etc.).
- Routing Configuration Questions: Please be sure to include details about your entire network, e.g., which IP addresses are on which networks using what netmasks? Send the output of `ifconfig -a` on each machine involved (for systems that do not support `ifconfig -a`, send the output of `ifconfig -a` for each interface) and include the output from `netstat -rn` from each system involved. Be sure to say which systems are running what software (including version numbers).

Unsupported Software

BSD/OS 4.0 includes a few pieces of software for which BSDI offers no support, and many software packages written and maintained by other organizations and individuals. While BSDI does not encourage customers to use unsupported software, we include this software in the hope that it will be useful to customers in the rare circumstances that require it.

While we always want to hear about any problems that you might experience in using any BSDI supplied software, we might occasionally choose not to provide support for a piece of software or to refer technical questions and problems to the software's maintainers. For further information and specific notes about software that is less likely to be supported by BSDI, see the *Contributed Software* chapter and the *Unsupported Kernel Software* section in the *Rebuilding the Kernel* chapter.

Getting Patches

BSDI makes patches available when a security problem arises, when a significant problem affecting the utility of a piece of software is discovered, and when specific system enhancements are made available between releases, e.g., the TCP/IP performance enhancements patch. Patches that provide system enhancements are only available to customers with support contracts. Security and bug fix patches are available to all customers.

The email support contact for each site can obtain the current patch information by sending email to `patches@bsdi.com`. To get further instructions, send email with the single line

```
  send help
```

as the message body. To get the current list of patches, send email with the single line

```
send index
```

as the message body.

You must have registered your email address with BSDI before this will work. If the system doesn't recognize you by your email address, you can specify your customer ID by sending a message with your customer ID, e.g.:

```
cust G9999
send index
```

However, except in the case of "help", the reply will still be sent to the email address registered with BSDI; see the full help instructions for more information.

If you have questions, or have forgotten who is the support contact for your site, or need to register your email address with the email patch server, contact your vendor's support address or BSDI support at `support@bsdi.com`.

Installing Mods

Each mod is self-explanatory and is a self-installing `perl5` script. To install a mod, in this case, the mod `M400-001`, run it as a `perl5` script:

```
# perl5 M400-001 apply
```

The script will check for prerequisites, back up previous versions of the files, and install source and/or binary versions of the patches.

There are several options that can be specified to each mod, run the mod without the "apply" argument for more information on what options are available.

A log of installed mods may be views by running the `modlog` command.

Acknowledgments

BSD/OS includes software distributed by the Computer Systems Research Group at the University of California, Berkeley. BSDI gratefully acknowledges their contribution.

Organizations that funded the CSRG with grants, gifts, personnel, and/or hardware.

Center for Advanced Aviation System Development, The MITRE Corp.
Compaq Computer Corporation
Cray Research Inc.
Department of Defense Advance Research Projects Agency (DARPA)
Digital Equipment Corporation
The Hewlett-Packard Company
NASA Ames Research Center
The National Science Foundation
The Open Software Foundation
UUNET Technologies Inc.

The following people and organizations provided a large subsystem for the BSD releases. (Their contribution might not be part of the final 4.4BSD release.)

ANSI C library	Chris Torek
ANSI C prototypes	Donn Seeley and John Kohl
Autoconfiguration	Robert Elz
C library documentation	American National Standards Committee X3
CCI 6/32 support	Computer Consoles Inc.
DEC 3100/5000 support	Ralph Campbell
Disklabels	Symmetric Computer Systems
Documentation	Cynthia Livingston and The USENIX Association
Franz Lisp	Richard Fateman, John Foderaro, Keith Sklower, Kevin Layer
GCC, GDB	The Free Software Foundation
Groff	James Clark (The FSF)
HP300 support	Jeff Forys, Mike Hibler, Jay Lepreau, Donn Seeley and the Systems Programming Group; University of Utah Computer Science Department
ISODE	Marshall Rose
Ingres	Mike Stonebraker, Gene Wong, and the Berkeley Ingres Research Group
Intel 386/486 support	Bill Jolitz and TeleMuse
Job control	Jim Kulp
Kerberos	Project Athena and MIT
Kernel support	Bill Shannon and Sun Microsystems Inc.
LFS	Margo Seltzer, Mendel Rosenblum, Carl Staelin
MIPS support	Trent Hein
Math library	K. C. Ng, Zhishun Alex Liu, S. McDonald, P. Tang and W. Kahan
NFS	Rick Macklem
NFS automounter	Jan-Simon Pendry
Network device drivers	Micom-Interlan and Excelan

Omron Luna support	Akito Fujita and Shigeto Mochida
Quotas	Robert Elz
RPC support	Sun Microsystems Inc.
Shared library support	Rob Gingell and Sun Microsystems Inc.
Sony News 3400 support	Kazumasa Utashiro
Sparc I/II support	Computer Systems Engineering Group, Lawrence Berkeley Laboratory
Stackable file systems	John Heidemann
Stdio	Chris Torek
System documentation	The Institute of Electrical and Electronics Engineers, Inc.
TCP/IP	Rob Gurwitz and Bolt Beranek and Newman Inc.
Timezone support	Arthur David Olson
Transport/Network OSI layers	IBM Corporation and the University of Wisconsin
Kernel XNS assistance	William Nesheim, J. Q. Johnson, Chris Torek, and James O'Toole
User level XNS	Cornell University
VAX 3000 support	Mt. Xinu and Tom Ferrin
VAX BI support	Chris Torek
VAX device support	Digital Equipment Corporation and Helge Skrivervik
VAX virtual memory implementation	Ozalp Babaoglu
Versatec printer/plotter support	University of Toronto
Virtual memory implementation	Avadis Tevanian, Jr., Michael Wayne Young, and the Carnegie Mellon University Mach project
X25	University of British Columbia

The following people and organizations provided a specific item, program, library routine or program maintenance for the BSD system. (Their contribution might not be part of the final 4.4BSD release.)

386 device drivers	Carnegie Mellon University Mach project Don Ahn, Sean Fagan and Tim Tucker
HCX device drivers	Harris Corporation
Kernel enhancements	Robert Elz, Peter Ivanov, Ian Johnstone, Piers Lauder, John Lions, Tim Long, Chris Maltby, Greg Rose and John Wainwright
ISO-9660 filesystem	Pace Willisson, Atsushi Murai
System enhancements	Charles M. Hannum
bha reporting enhancements	Terry Kennedy
bha wide support	Terry Kennedy
bha error reporting	Terry Kennedy
NFS interoperability fixes	Terry Kennedy
ports of the *stat utilities	Terry Kennedy

adventure(6)	Don Woods
adventure(6)	Jim Gillogly
adventure(6)	Will Crowther
apply(1)	Rob Pike
apply(1)	Jan-Simon Pendry
ar(1)	Hugh A. Smith
arithmetic(6)	Eamonn McManus
arp(8)	Sun Microsystems Inc.
at(1)	Steve Wall
atc(6)	Ed James
awk(1)	Arnold Robbins
awk(1)	David Trueman
ba'kgam'n(6)	Alan Char
banner(1)	Mark Horton
battlestar(6)	David Riggle
bcd(6)	Steve Hayman
bdes(1)	Matt Bishop
berknet(1)	Eric Schmidt
bib(1)	Dain Samples
bib(1)	Gary M. Levin
bib(1)	Timothy A. Budd
bind(8)	Kevin Dunlap

bind(8)	Douglas Terry
bitstring(3)	Paul Vixie
bpf(4)	Steven McCanne
btree(3)	Mike Olson
byte-range locking	Scooter Morris
caesar(6)	John Eldridge
caesar(6)	Stan King
cal(1)	Kim Letkeman
cat(1)	Kevin Fall
chess(6)	Stuart Cracraft (The FSF)
ching(6)	Guy Harris
cksum(1)	James W. Williams
clri(8)	Rich $alz
col(1)	Michael Rendell
comm(1)	Case Larsen
compact(1)	Colin L. McMaster
compress(1)	James A. Woods
compress(1)	Joseph Orost
compress(1)	Spencer Thomas
courier(1)	Eric Cooper
cp(1)	David Hitz
cpio(1)	AT&T
crypt(3)	Tom Truscott
csh(1)	Christos Zoulas
csh(1)	Len Shar
curses(3)	Elan Amir
curses(3)	Ken Arnold
cut(1)	Adam S. Moskowitz
cut(1)	Marciano Pitargue
dbx(1)	Mark Linton
dd(1)	Keith Muller
dd(1)	Lance Visser
des(1)	Jim Gillogly
des(1)	Phil Karn
des(1)	Richard Outerbridge
dipress(1)	Xerox Corporation
disklabel(8)	Symmetric Computer Systems
du(1)	Chris Newcomb
dungeon(6)	R.M. Supnik
ed(1)	Rodney Ruddock
emacs(1)	Richard Stallman
erf(3)	Peter McIlroy, K.C. Ng
error(1)	Robert R. Henry
ex(1)	Mark Horton
factor(6)	Landon Curt Noll
file(1)	Ian Darwin
find(1)	Cimarron Taylor
finger(1)	Tony Nardo
fish(6)	Muffy Barkocy
fmt(1)	Kurt Schoens
fnmatch(3)	Guido van Rossum
fold(1)	Kevin Ruddy
fortune(6)	Ken Arnold
fpr(1)	Robert Corbett
fsdb(8)	Computer Consoles Inc.
fsplit(1)	Asa Romberger
fsplit(1)	Jerry Berkman
gcc/groff integration	UUNET Technologies, Inc.
gcore(1)	Eric Cooper
getcap(3)	Casey Leedom
glob(3)	Guido van Rossum
gprof(1)	Peter Kessler
gprof(1)	Robert R. Henry
hack(6)	Andries Brouwer (and a cast of thousands)
hangman(6)	Ken Arnold
hash(3)	Margo Seltzer
heapsort(3)	Elmer Yglesias
heapsort(3)	Kevin Lew
heapsort(3)	Ronnie Kon
hunt(6)	Conrad Huang
hunt(6)	Greg Couch
icon(1)	Bill Mitchell
icon(1)	Ralph Griswold
indent(1)	David Willcox
indent(1)	Eric Schmidt
indent(1)	James Gosling
indent(1)	Sun Microsystems
init(1)	Donn Seeley
j0(3)	Sun Microsystems, Inc.
j1(3)	Sun Microsystems, Inc.
jn(3)	Sun Microsystems, Inc.
join(1)	David Goodenough
join(1)	Michiro Hikida
join(1)	Steve Hayman
jot(1)	John Kunze
jove(1)	Jonathon Payne
kermit(1)	Columbia University
kvm(3)	Peter Shipley
kvm(3)	Steven McCanne
lam(1)	John Kunze
larn(6)	Noah Morgan
lastcomm(1)	Len Edmondson
lex(1)	Vern Paxson
libm(3)	Peter McIlroy
libm(3)	UUNET Technologies, Inc.
locate(1)	James A. Woods
lock(1)	Bob Toxen
log(3)	Peter McIlroy
look(1)	David Hitz
ls(1)	Elan Amir
ls(1)	Michael Fischbein
lsearch(3)	Roger L. Snyder
m4(1)	Ozan Yigit
mail(1)	Kurt Schoens
make(1)	Adam de Boor
me(7)	Eric Allman
mergesort(3)	Peter McIlroy
mh(1)	Marshall Rose
mh(1)	The Rand Corporation
mknod(8)	Kevin Fall

monop(6)	Ken Arnold
more(1)	Eric Shienbrood
more(1)	Mark Nudleman
mountd(8)	Herb Hasler
mprof(1)	Ben Zorn
msgs(1)	David Wasley
multicast	Stephen Deering
mv(1)	Ken Smith
named(8)	Douglas Terry
named(8)	Kevin Dunlap
news(1)	Rick Adams (and a cast of thousands)
nm(1)	Hans Huebner
pascal(1)	Kirk McKusick
pascal(1)	Peter Kessler
paste(1)	Adam S. Moskowitz
patch(1)	Larry Wall
pax(1)	Keith Muller
phantasia(6)	C. Robertson
phantasia(6)	Edward A. Estes
ping(8)	Mike Muuss
pom(6)	Keith E. Brandt
pr(1)	Keith Muller
primes(6)	Landon Curt Noll
qsort(3)	Doug McIlroy
qsort(3)	Earl Cohen
qsort(3)	Jon Bentley
quad(3)	Chris Torek
quiz(6)	Jim R. Oldroyd
quiz(6)	Keith Gabryelski
radixsort(3)	Dan Bernstein
radixsort(3)	Peter McIlroy
rain(6)	Eric P. Scott
ranlib(1)	Hugh A. Smith
rcs(1)	Walter F. Tichy
rdist(1)	Michael Cooper
regex(3)	Henry Spencer
robots(6)	Ken Arnold
rogue(6)	Timothy C. Stoehr
rs(1)	John Kunze
rune support	Paul Borman
sail(6)	David Riggle
sail(6)	Edward Wang
sccs(1)	Eric Allman
scsiformat(1)	Lawrence Berkeley Laboratory
sdb(1)	Howard Katseff
sed(1)	Diomidis Spinellis
sendmail(8)	Eric Allman
setmode(3)	Dave Borman
sh(1)	Kenneth Almquist
slattach(8)	Rick Adams
slip(8)	Rick Adams
spms(1)	Peter J. Nicklin
strtod(3)	David M. Gay
swab(3)	Jeffrey Mogul
sysconf(3)	Sean Eric Fagan
sysline(1)	J.K. Foderaro
syslog(3)	Eric Allman
systat(1)	Bill Reeves
systat(1)	Robert Elz
sysv_shm	Adam Glass & Charles Hannum
tail(1)	Edward Sze-Tyan Wang
talk(1)	Clem Cole
talk(1)	Kipp Hickman
talk(1)	Peter Moore
telnet(1)	Dave Borman
telnet(1)	Paul Borman
termcap(5)	John A. Kunze
termcap(5)	Mark Horton
test(1)	Kenneth Almquist
tetris(6)	Chris Torek
tetris(6)	Darren F. Provine
timed(8)	Riccardo Gusella
timed(8)	Stefano Zatti
tn3270(1)	Gregory Minshall
tr(1)	Igor Belchinskiy
traceroute(8)	Van Jacobson
trek(6)	Eric Allman
tset(1)	Eric Allman
tsort(1)	Michael Rendell
unifdef(1)	Dave Yost
uniq(1)	Case Larsen
uucpd(8)	Rick Adams
uudecode(1)	Mark Horton
uuencode(1)	Mark Horton
uupoll(1)	Brett Wynkoop
uuq(1)	Lou Salkind
uuq(1)	Rick Adams
uusnap(8)	Randy King
uusnap(8)	Rick Adams
vacation(1)	Eric Allman
vi(1)	Steve Kirkendall
which(1)	Peter Kessler
who(1)	Michael Fischbein
window(1)	Edward Wang
worm(6)	Michael Toy
worms(6)	Eric P. Scott
write(1)	Craig Leres
write(1)	Jef Poskanzer
wump(6)	Dave Taylor
X25/Eth'net	Univ. of Erlangen-Nuremberg
X25/LLC2	Dirk Husemann
xargs(1)	John B. Roll Jr.
xneko(6)	Masayuki Koba
XNSrout'd(1)	Bill Nesheim
xroach(6)	J.T. Anderson
yacc(1)	Robert Paul Corbett

The following people and organizations have provided a specific item, program, library routine, or program maintenance used in the BSD/OS system. (Their contribution might not be part of the current BSD/OS 4.0 release.)

Allied Telesis RE2000/AT-1700 series Ethernet driver	Sei Kigoshi
Belgian keyboard map	Lode Vande Sande
Booting larger kernels	Yuval Yarom
Bulgarian keyboard map	Radka Kalcheva
Chase serial port cards	Chase
DPT drivers	DPT
Equinox serial port cards	Equinox
HP EtherTwist PC LAN Adapter	Kevin Fall
Keyboard mapping support	Dirk Husemann
Maxpeed driver	Doug Urner
NCSA Mosaic for the X Window System	National Center for Supercomputing Applications
Network performance enhancements	Steve Nuchia, South Coast Computing Services
PCMCIA and APM	Yoichi Shinoda, Yoshitaka Tokugawa, The WIDE Project, The Wildboar Project and Foretune Co., Ltd.
PS/2 mouse driver	Jan-Simon Pendry
Stallion serial port cards	Stallion
TNIC-1500 driver	Steve Nuchia, South Coast Computing Services
VoxWare sound package	Hannu Savolainen

Apache

BSD/OS 4.0 includes software developed by the Apache Group for use in the Apache HTTP server project (http://www.apache.org/).

Squid

Squid is the result of efforts by numerous individuals from the Internet community. Development is led by Duane Wessels of the National Laboratory for Applied Network Research and funded by the National Science Foundation. Squid is derived from the "cached" software from the ARPA-funded Harvest research project. The Harvest home page is http://harvest.cs.colorado.edu/.

Xv

BSDI thanks John Bradley (bradley@cis.upenn.edu) for permission to ship xv(1). Xv is shareware, not freeware, so if you find it useful a donation to the author is recommended. See the "License" selection on the XV control panel for more information.

Rich Salz

BSD/OS 4.0 includes software developed by Rich Salz.

RADIUS

BSD/OS 4.0 includes the Basic Merit AAA Server for the RADIUS protocol. It is provided by Merit Network, Inc. and The Regents of the University of Michigan More information about this RADIUS server, along with information about the Enhanced Merit AAA Server is available at their website: http://www.merit.edu/aaa

The following copyright notices acknowledge others' work that is included in BSD/OS 4.0.

University of California Berkeley

All of the documentation and software included in the 4.4BSD and 4.4BSD-Lite Releases is copyrighted by The Regents of the University of California.

Copyright 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994 The Regents of the University of California. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:
   This product includes software developed by the University of California, Berkeley and its contributors.
4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The Institute of Electrical and Electronics Engineers and the American National Standards Committee X3, on Information Processing Systems 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 the second BSD Networking Software Release, from IEEE Std 1003.1-1988, IEEE Standard Portable Operating System Interface for Computer Environments (POSIX), copyright C 1988 by the Institute of Electrical and Electronics Engineers, Inc. In the event of any discrepancy between these versions and the original IEEE Standard, the original IEEE Standard is the referee document.

In the following statement, the phrase "This material" refers to portions of the system documentation.

This material is reproduced with permission from American National Standards Committee X3, on Information Processing Systems. Computer and Business Equipment Manufacturers Association (CBEMA), 311 First St., NW, Suite 500, Washington, DC 20001-2178. The developmental work of Programming Language C was completed by the X3J11 Technical Committee.

The views and conclusions contained in the software and documentation are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the Regents of the University of California.

PS/2 Mouse Driver

Copyright © Computer Newspaper Services Limited 1993 All rights reserved.

License to use, copy, modify, and distribute this work and its documentation for any purpose and without fee is hereby granted, provided that you also ensure modified files carry prominent notices stating that you changed the files and the date of any change, ensure that the above copyright notice appear in all copies, that both the copyright notice and this permission notice appear in supporting documentation, and that the name of Computer Newspaper Services not be used in advertising or publicity pertaining to distribution or use of the work without specific, written prior permission from Computer Newspaper Services.

Xntp Program

The following copyright notice applies to all files collectively called the Network Time Protocol Version 4 Distribution. Unless specifically declared otherwise in an individual file, this notice applies as if the text was explicitly included in the file.

Copyright © David L. Mills 1992, 1993, 1994, 1995, 1996 Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appears in all copies and that both the copyright notice and this permission notice appear in supporting documentation, and that the name University of Delaware not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. The University of Delaware makes no representations about the suitability this software for any purpose. It is provided "as is" without express or implied warranty.

Wild Boar

Portions or all of this software are Copyright © 1994, 1995, 1996 Yoichi Shinoda, Yoshitaka Tokugawa, WIDE Project, Wildboar Project and Foretune. All rights reserved.

This code has been contributed to Berkeley Software Design, Inc. by the Wildboar Project and its contributors.

The Berkeley Software Design Inc. software License Agreement specifies the terms and conditions for redistribution.

THIS SOFTWARE IS PROVIDED BY THE WILDBOAR PROJECT AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE WILDBOAR PROJECT OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Adam Glass and Charles Hannum - Shared Memory

Copyright © 1994 Adam Glass and Charles Hannum. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by Adam Glass and Charles Hannum.
4. The names of the authors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The Apache Group

Copyright © 1995-1997 The Apache Group. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes software developed by the Apache Group for use in the Apache HTTP server project (http://www.apache.org/)."
4. The names "Apache Server" and "Apache Group" must not be used to endorse or promote products derived from this software without prior written permission.
5. Redistributions of any form whatsoever must retain the following acknowledgment:

   "This product includes software developed by the Apache Group for use in the Apache HTTP server project (http://www.apache.org/)."

THIS SOFTWARE IS PROVIDED BY THE APACHE GROUP "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE GROUP OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Group and was originally based on public domain software written at the National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign. For more information on the Apache Group and the Apache HTTP server project, please see <http://www.apache.org/>.

Sendmail, Inc.

Copyright © 1998

THIS SOFTWARE IS PROVIDED BY SENDMAIL, INC. AND CONTRIBUTORS "AS I" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL SENDMAIL, INC., THE REGENTS OF THE UNIVERSITY OF CALIFORNIA OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

US Naval Research Laboratory

Copyright © 1.1 (NRL) 17 January 1995

COPYRIGHT NOTICE

All of the documentation and software included in this software distribution from the US Naval Research Laboratory (NRL) are copyrighted by their respective developers.

Portions of the software are derived from the Net/2 and 4.4 Berkeley Software Distributions (BSD) of the University of California at Berkeley and those portions are copyright by The Regents of the University of California. All Rights Reserved. The UC Berkeley Copyright and License agreement is binding on those portions of the software. In all cases, the NRL developers have retained the original UC Berkeley copyright and license notices in the respective files in accordance with the UC Berkeley copyrights and license.

Portions of this software and documentation were developed at NRL by various people. Those developers have each copyrighted the portions that they developed at NRL and have assigned All Rights for those portions to NRL. Outside the USA, NRL has copyright on some of the software developed at NRL. The affected files all contain specific copyright notices and those notices must be retained in any derived work.

NRL LICENSE

NRL grants permission for redistribution and use in source and binary forms, with or without modification, of the software and documentation created at NRL provided that the following conditions are met:

1. All terms of the UC Berkeley copyright and license must be followed.
2. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
3. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
4. All advertising materials mentioning features or use of this software must display the following acknowledgements:

   This product includes software developed by the University of California, Berkeley and its contributors.

   This product includes software developed at the Information Technology Division, US Naval Research Laboratory.
5. Neither the name of the NRL nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THE SOFTWARE PROVIDED BY NRL IS PROVIDED BY NRL AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL NRL OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The views and conclusions contained in the software and documentation are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the US Naval Research Laboratory (NRL).

Other Acknowledgments

The following individuals contributed in part to the Network Time Protocol Distribution Version 4 and are acknowledged as authors of this work.

Mark Andrews <marka@syd.dms.csiro.au>: Leitch atomic clock controller.

Viraj Bais <vbais@mailman1.intel.com>: and Clayton Kirkwood <kirkwood@striderfm.intel.com> port to WindowsNT 3.5.

Karl Berry <karl@owl.HQ.ileaf.com>: syslog to file option.

Piete Brooks <Piete.Brooks@cl.cam.ac.uk>: MSF clock driver, Trimble PARSE support.

Steve Clift <clift@ml.csiro.au>: OMEGA clock driver.

Torsten Duwe <duwe@immd4.informatik.uni-erlangen.de>: Linux Port.

John A. Dundas III <dundas@salt.jpl.nasa.gov>: Apple A/UX port.

Dennis Ferguson <dennis@mrbill.canet.ca>: foundation code for NTP Version 2 as specified in RFC-1119.

Glenn Hollinger <glenn@herald.usask.ca>: GOES clock driver.

Mike Iglesias <iglesias@uci.edu>: DEC Alpha port

Jim Jagielski <jim@jagubox.gsfc.nasa.gov>: A/UX port.

Jeff Johnson <jbj@chatham.usdesign.com>: massive prototyping overhaul.

William L. Jones <jones@hermes.chpc.utexas.edu>: RS/6000 AIX modifications, HPUX modifications.

Dave Katz <dkatz@cisco.com>: RS/6000 AIX port.

Craig Leres <leres@ee.lbl.gov>: 4.4BSD port, ppsclock, Maganavox GPS clock driver.

George Lindholm <lindholm@ucs.ubc.ca>: SunOS 5.1 port.

Louis A. Mamakos <louie@ni.umd.edu>: MD5-based authentication.

Lars H. Mathiesen <thorinn@diku.dk>: adaptation of foundation code for Version 3 as specified in RFC-1305.

David L. Mills <mills@udel.edu>: Spectractom WWVB, Austron GPS, Heath, ATOM, ACTS, KSI/Odetics IRIG-B clock drivers; PPS support.

Wolfgang Moeller <moeller@gwdgv1.dnet.gwdg.de>: VMS port.

Jeffrey Mogul <mogul@pa.dec.com>: ntptrace utility.

Tom Moore <tmoore@fievel.daytonoh.ncr.com>: i386 svr4 port.

Rainer Pruy <Rainer.Pruy@informatik.uni-erlangen.de>: monitoring/trap scripts, statistics file handling.

Nick Sayer <mrapple@quack.kfu.com>: SunOS streams modules.

Frank Kardel <Frank.Kardel@informatik.uni-erlangen.de>: PARSE (GENERIC) driver, STREAMS modules for PARSE, support scripts, reference clock configuration scripts, Makefile cleanup, syslog cleanup.

Ray Schnitzler <schnitz@unipress.com>: Unixware1 port.

Michael Shields <shields@tembel.org>: USNO clock driver.

Jeff Steinman <jss@pebbles.jpl.nasa.gov>: Datum PTS clock driver.

Kenneth Stone <ken@sdd.hp.com>: HP-UX port.

Ajit Thyagarajan <ajit@ee.udel.edu>: IP multicast support.

Tomoaki TSURUOKA <tsuruoka@nc.fukuoka-u.ac.jp>: TRAK clock driver.

Paul A Vixie <vixie@vix.com>: TrueTime GPS driver, generic TrueTime clock driver.

Appendix A

Xmetro Video Cards

ATI 3D RAGE (3D RAGE)
ATI 3D RAGE II (3D RAGE II)
ATI 3D RAGE II + DVD
ATI 3D RAGE II
ATI Graphics Pro Turbo (Mach64)
ATI Graphics Ultra (Mach8)
ATI Graphics Xpression (Mach64)
ATI Mach32 (Mach32)
ATI Mach64 (Mach64)
ATI VGA STEREO-F/X (ATI 28800)
ATI Winturbo PCI (Mach64)
ATI XPERT@Play (3D RAGE PRO)
ATI XPERT@Play AGP (3D RAGE PRO)
ATI XPERT@Work (3D RAGE PRO)
ATI XPERT@Work AGP (3D RAGE PRO)
Actix Graphics Engine 32
BTC 11VO (Oak Technology 087)
Cardex ET4000 (ET4000)
Diamond Fire GL 3000 (GLINT 500TX)
Diamond SpeedStar 24X (Western Digital 90C31)
Diamond SpeedStar Pro SE (Cirrus 5430)
Diamond Stealth 24 (S3 801)
Diamond Stealth 32 (ET4000/W32p)
Diamond Stealth 3D 2000 (S3 ViRGE)
Diamond Stealth 64 (86c968-P, IBM RGB526CF22)
Diamond Stealth 64 (S3 964, Bt485KPJ135)
Diamond Stealth 64 DRAM (S3 Trio64)
Diamond Stealth 64 DRAM (SDAC) (S3 864, S3 SDAC)
Diamond Stealth 64 Graphics 2000 Series (S3 864, S3 SDAC)
Diamond Stealth 64 Graphics 2200 (S3 Trio64)
Diamond Stealth 64 VRAM (S3 968, IBM RGB526CF22)
Diamond Stealth 64 Video 3000 Series (S3 968, TI 3026-175)
Diamond Stealth 64 Video VRAM (S3 968, TI 3026-175)
Diamond Stealth 64 Video VRAM
Diamond Stealth Video (SDAC) (S3 868, S3 SDAC)
Diamond Stealth Video 2000 Series (S3 868, S3 SDAC)
Diamond Stealth64 VRAM (86c968-P, IBM RGB526CF22)
Diamond Stealth64 Video 3000 Series
Diamond Viper (110 MHz RAMDAC) (P9000)
Diamond Viper (135 MHz RAMDAC) (P9000)
Diamond Viper Pro (P9100)
Diamond Viper SE (P9100)

ELSA GLoria Synergy (PERMEDIA 2)
ELSA GLoria-L (GLINT 500TX)
ELSA GLoria-L/MX (GLINT 500TX)
ELSA GLoria-XL (GLINT MX)
ELSA Gloria-L (Glint 500TX)
ELSA Gloria-L (S3 ViRGE)
ELSA Gloria-L/MX (Glint MX)
ELSA Gloria-L/MX (S3 ViRGE)
ELSA Gloria-XL (Glint MX)
ELSA Gloria-XL (Glint MX, IBM RGB640/250Hz)
ELSA Gloria-XL (S3 Trio64 V2/DX, IBM RGB640/250Hz)
ELSA Victory 3D (S3 ViRGE)
ELSA WINNER 2000 Office (PERMEDIA 2)
ELSA Winner 1000 TRIO/V (S3 Trio64V+)
ELSA Winner 2000 AVI (S3 968, TI 3026-175)
ELSA Winner 2000 AVI
ELSA Winner 2000 PRO/X
ELSA Winner 2000 PRO/X-2, -4 (S3 968, TI 3026-220)
ELSA Winner 2000 PRO/X-8 (S3 968, IBM RGB528CF25)
ELSA Winner 2000 PRO/X-8
EPS Apex L-200 (C&T 65550)
Generic ATI 28800
Generic Alliance ProMotion
Generic Ark 2000
Generic Avance Logic 22xx/23xx/24xx
Generic Chips & Technologies
Generic Cirrus 5420
Generic Cirrus 5422
Generic Cirrus 5422/5424
Generic Cirrus 5424
Generic Cirrus 5426
Generic Cirrus 5426/5428
Generic Cirrus 5428
Generic Cirrus 5429
Generic Cirrus 5430
Generic Cirrus 5434
Generic Cirrus 5436
Generic Cirrus 5446
Generic Cirrus 5462
Generic Cirrus 5462/5465
Generic Cirrus 5480
Generic Cirrus 62x5
Generic Cirrus 6410/6412/6420/6440
Generic Cirrus 7541
Generic Cirrus 7542
Generic Cirrus 7543

Generic Cirrus 754x
Generic ET3000
Generic ET4000/W32P
Generic ET4000AX
Generic ET6000
Generic GLINT
Generic IBM 8514 compatible
Generic Mach64
Generic Oak Technology 067/077/087
Generic P9000
Generic P9100
Generic PERMEDIA 2
Generic S3 864/868/924/928/964
Generic S3 Trio64
Generic S3 Trio64V+
Generic S3 ViRGE
Generic S3 ViRGE
Generic SiS 86c201/86c202/86c205
Generic Trident
Generic VGA
Generic Video 7 SVGA
Generic Western Digital SVGA
Genoa WindowsVGA 8500VL (Cirrus 5426)
Genoa WindowsVGA(8500VL)
Genoa 8500
Genoa Phantom 64 (S3 Trio64V+)
Headland GC208 (Video 7)
Hercules Dynamite 128/Video (ET6000)
Hercules Dynamite 3D/GL (PERMEDIA 2)
Hercules Dynamite 3D/GL AGP (PERMEDIA 2)
Hercules Stingray (Avance Logic 2301)
Hercules Stingray 128/3D (Alliance ProMotion AT3D)
Hercules Stingray 64 (Ark 2000)
Hercules Terminator 3D (S3 ViRGE/DX)
IBM VGA
Matrox Marvel (ET4000)
Matrox Marvel II (ET4000)
Matrox Millennium (MGA Storm)
Matrox Millennium II (MGA 2164, TVP3026-220PCE)
Matrox Millennium II (MGA Storm)
Matrox Millennium II AGP (MGA 2164, TVP3026-250CPCE)
Matrox Mystique (MGA 1064)
Matrox Mystique 220 (MGA 1164)
Metheus Premier 928
Number Nine GXE (S3 928)
Number Nine GXE64 (S3 864)

Number Nine GXE64 Pro
Number Nine Imagine 128 (Imagine 128)
Number Nine Imagine 128 Series 2 (Imagine 128 Series 2)
Number Nine Imagine 128
Number Nine Motion 531 (S3 868)
Number Nine Motion 771 (S3 968)
Number Nine Revolution 3D (Ticket to Ride)
Number Nine Vision 330 (S3 Trio64)
Orchid Fahrenheit VA (S3 801)
Orchid Kelvin 64 (Cirrus 5434)
SPEA Mirage Video (S3 Trio64V+)
STB Glyder MAX-2 (Permedia)
STB NITRO 3D (S3 ViRGE/GX)
STB S3 ViRGE Reference Board
STB VELOCITY 3D (S3 ViRGE XV)
STB/Symmetric GLyder MAX-2 (PERMEDIA 2)
STB/Symmetric GLyder TX GOLD (GLINT 500TX)
Sigma Designs VGA Legend (ET4000)
Tech Source Raptor (Imagine 128 Series 2)
Trident 8900 (Trident 8900)
V PCI-53 (Cirrus 5434)

Xmetro Monitors

AT&T PRO 21
AT&T 3298-0290-8090
CAPTRONICS OEM TRINITRON 20
DELL GPD-16C
DELL GPD-19C
DELL Hewitt
DELL Super VGA
DELL Super VGA Colour
DELL Super VGA DL 1428 I/L
DELL Super VGA Jostens
DELL UGA DL 1460 NI
DELL UltraScan 14C
DELL UltraScan 14C-E
DELL UltraScan 14C-EN
DELL UltraScan 14ES
DELL UltraScan 14LR
DELL UltraScan 14XE
DELL UltraScan 15ES/15ES-P
DELL UltraScan 15FS-N/15FS-EN
DELL UltraScan 15FS/15FS-E
DELL UltraScan 15LR
DELL UltraScan 15TE
DELL UltraScan 17ES

DELL UltraScan 17FS-ELR
DELL UltraScan 17FS-EN
DELL UltraScan 17FS-LR
DELL UltraScan 17FS-N
DELL UltraScan 17HS
DELL UltraScan 21FS
DELL UltraScan 21TE
DELL UltraScan V17X
DELL V15X
DELL V17X
DELL VC15 Colour
DELL VGA 800
DELL VGA Color/Color Plus
DELL VGA Monochrome
DELL VS14
DELL VS15
DELL VS17
DELL VS17X
DELL Vi14X
DELL 1024i
DELL 1024i-P/1024i-Color
GENERIC 640x 480 @ 60Hz 13 inch
GENERIC 640x 480 @ 60Hz 14 inch
GENERIC 640x 480 @ 60Hz 15 inch
GENERIC 800x 600 @ 60Hz 14 inch
GENERIC 800x 600 @ 60Hz 15 inch
GENERIC 800x 600 @ 60Hz 16 inch
GENERIC 1024x 768 @ 60Hz 14 inch
GENERIC 1024x 768 @ 60Hz 15 inch
GENERIC 1024x 768 @ 60Hz 16 inch
GENERIC 1024x 768 @ 60Hz 17 inch
GENERIC 1024x 768 @ 60Hz 19 inch
GENERIC 1024x 768 @ 60Hz 20 inch
GENERIC 1024x 768 @ 60Hz 21 inch
GENERIC 1024x 768 @ 75Hz 14 inch
GENERIC 1024x 768 @ 75Hz 15 inch
GENERIC 1024x 768 @ 75Hz 16 inch
GENERIC 1024x 768 @ 75Hz 17 inch
GENERIC 1024x 768 @ 75Hz 19 inch
GENERIC 1024x 768 @ 75Hz 20 inch
GENERIC 1024x 768 @ 75Hz 21 inch
GENERIC 1024x 768 @ 85Hz 14 inch
GENERIC 1024x 768 @ 85Hz 15 inch
GENERIC 1024x 768 @ 85Hz 16 inch
GENERIC 1024x 768 @ 85Hz 17 inch
GENERIC 1024x 768 @ 85Hz 19 inch

GENERIC 1024x 768 @ 85Hz 20 inch
GENERIC 1024x 768 @ 85Hz 21 inch
GENERIC 1024x 768 @ 87Hz interlaced 14 inch
GENERIC 1024x 768 @ 87Hz interlaced 15 inch
GENERIC 1024x 768 @ 87Hz interlaced 16 inch
GENERIC 1024x 768 @ 87Hz interlaced 17 inch
GENERIC 1280x1024 @ 60Hz 16 inch
GENERIC 1280x1024 @ 60Hz 17 inch
GENERIC 1280x1024 @ 60Hz 19 inch
GENERIC 1280x1024 @ 60Hz 20 inch
GENERIC 1280x1024 @ 60Hz 21 inch
GENERIC 1280x1024 @ 75Hz 16 inch
GENERIC 1280x1024 @ 75Hz 17 inch
GENERIC 1280x1024 @ 75Hz 19 inch
GENERIC 1280x1024 @ 75Hz 20 inch
GENERIC 1280x1024 @ 75Hz 21 inch
GENERIC 1280x1024 @ 85Hz 16 inch
GENERIC 1280x1024 @ 85Hz 17 inch
GENERIC 1280x1024 @ 85Hz 19 inch
GENERIC 1280x1024 @ 85Hz 20 inch
GENERIC 1280x1024 @ 85Hz 21 inch
GENERIC 1600x1200 @ 60Hz 19 inch
GENERIC 1600x1200 @ 60Hz 20 inch
GENERIC 1600x1200 @ 60Hz 21 inch
GENERIC 1600x1200 @ 65Hz 19 inch
GENERIC 1600x1200 @ 65Hz 20 inch
GENERIC 1600x1200 @ 65Hz 21 inch
GENERIC 1600x1200 @ 75Hz 19 inch
GENERIC 1600x1200 @ 75Hz 20 inch
GENERIC 1600x1200 @ 75Hz 21 inch
GENERIC 1600x1200 @ 85Hz 19 inch
GENERIC 1600x1200 @ 85Hz 20 inch
GENERIC 1600x1200 @ 85Hz 21 inch
HITACHI HM4921D
MAG INNOVISION DX1495
MAG INNOVISION DX1595
MAG INNOVISION DX15T
MAG INNOVISION DX1795
MAG INNOVISION DX17T
MAG INNOVISION MXP17F
MITSUBISHI HL7955K
MITSUBISHI THZ8155SKTK
NEC MultiSync 3FGe
NEC MultiSync 3FGx
NEC MultiSync 3V
NEC MultiSync 4FG

NEC MultiSync 4FGe
NEC MultiSync 5FG
NEC MultiSync 5FGe
NEC MultiSync 5FGp
NEC MultiSync 6FG
NEC MultiSync 6FGp
NEC MultiSync XE15
NEC MultiSync XE17
NEC MultiSync XE21
NEC MultiSync XP15
NEC MultiSync XP17
NEC MultiSync XP21
NEC MultiSync XV14
NEC MultiSync XV15
NEC MultiSync XV17
NOKIA 44BS
NOKIA 449E
SONY CPD-1302
SONY CPD-1304
SONY CPD-1304S
SONY CPD-1425
SONY CPD-1430
SONY CPD-15SF1
SONY CPD-15SFII
SONY CPD-15SX
SONY CPD-1604S
SONY CPD-1730
SONY CPD-17SF1
SONY CPD-17SFII
SONY CPD-20SFII
SONY GDM-17SE1
SONY GDM-17SEII
SONY GDM-1936
SONY GDM-2000TC
SONY GDM-2036S
SONY GDM-2038
SONY GDM-20SE1
VIEWSONIC 4E
VIEWSONIC 6E
VIEWSONIC 14ES
VIEWSONIC 15
VIEWSONIC 15E
VIEWSONIC 15ES
VIEWSONIC 15EX
VIEWSONIC 15G
VIEWSONIC 17

VIEWSONIC 17E
VIEWSONIC 17G
VIEWSONIC 20
VIEWSONIC 20G
VIEWSONIC 20PS
VIEWSONIC 21
VIEWSONIC 21PS

Appendix B

XF86 Video Cards

2 the Max MAXColor S3 Trio64V+
928Movie
Actix GE32+ 2MB
Actix GE32i
Actix GE64
Actix ProStar
Actix ProStar 64
Actix Ultra
Acumos AVGA3
AGX (generic)
ALG-5434(E)
Alliance ProMotion 6422
Ark Logic ARK1000PV (generic)
Ark Logic ARK1000VL (generic)
Ark Logic ARK2000MT (generic)
Ark Logic ARK2000PV (generic)
ASUS 3Dexplorer
ASUS PCI-AV264CT
ASUS PCI-V264CT
ASUS Video Magic PCI V864
ASUS Video Magic PCI VT64
AT25
AT3D
ATI 3D Pro Turbo
ATI 3D Xpression
ATI 3D Xpression+ PC2TV
ATI 8514 Ultra (no VGA)
ATI All-in-Wonder
ATI Graphics Pro Turbo
ATI Graphics Pro Turbo 1600
ATI Graphics Ultra
ATI Graphics Ultra Pro
ATI Graphics Xpression with 68875 RAMDAC
ATI Graphics Xpression with AT&T 20C408 RAMDAC
ATI Graphics Xpression with CH8398 RAMDAC
ATI Graphics Xpression with Mach64 CT (264CT)
ATI Graphics Xpression with STG1702 RAMDAC
ATI Mach32
ATI Mach64
ATI Mach64 3D RAGE II+, Internal RAMDAC
ATI Mach64 3D RAGE II, Internal RAMDAC
ATI Mach64 CT (264CT), Internal RAMDAC

ATI Mach64 GT (264GT), aka 3D RAGE, Internal RAMDAC
ATI Mach64 VT (264VT), Internal RAMDAC
ATI Mach64 with AT&T 20C408 RAMDAC
ATI Mach64 with CH8398 RAMDAC
ATI Mach64 with IBM RGB514 RAMDAC
ATI Pro Turbo+PC2TV, 3D Rage II+DVD
ATI Ultra Plus
ATI Video Xpression
ATI Win Boost with AT&T 20C408 RAMDAC
ATI Win Boost with CH8398 RAMDAC
ATI Win Boost with Mach64 CT (264CT)
ATI Win Boost with STG1702 RAMDAC
ATI Win Turbo
ATI Wonder SVGA
ATI Xpert@Play PCI and AGP, 3D Rage Pro
ATI Xpert@Work, 3D Rage Pro
ATrend ATC-2165A
Avance Logic 2101
Avance Logic 2228
Avance Logic 2301
Avance Logic 2302
Avance Logic 2308
Avance Logic 2401
Binar Graphics AnyView
Boca Vortex (Sierra RAMDAC)
California Graphics SunTracer 6000
Canopus Co. Power Window 3DV
Canopus Total-3D
Cardex Challenger (Pro)
Cardex Cobra
Cardex Trio64
Cardex Trio64Pro
Chips & Technologies CT64200
Chips & Technologies CT64300
Chips & Technologies CT65520
Chips & Technologies CT65525
Chips & Technologies CT65530
Chips & Technologies CT65535
Chips & Technologies CT65540
Chips & Technologies CT65545
Chips & Technologies CT65546
Chips & Technologies CT65548
Chips & Technologies CT65550
Chips & Technologies CT65554
Chips & Technologies CT65555
Chips & Technologies CT68554

Cirrus Logic GD542x
Cirrus Logic GD543x
Cirrus Logic GD544x
Cirrus Logic GD5462
Cirrus Logic GD5464
Cirrus Logic GD62xx (laptop)
Cirrus Logic GD64xx (laptop)
Cirrus Logic GD754x (laptop)
Colorgraphic Dual Lightning
COMPAQ Armada 7380DMT
COMPAQ Armada 7730MT
Creative Labs 3D Blaster PCI (Verite 1000)
Creative Labs Graphics Blaster 3D
Creative Labs Graphics Blaster MA201
Creative Labs Graphics Blaster MA202
Creative Labs Graphics Blaster MA302
Creative Labs Graphics Blaster MA334
DataExpert DSV3325
DataExpert DSV3365
Dell onboard ET4000
Dell S3 805
DFI-WG1000
DFI-WG5000
DFI-WG6000
Diamond Edge 3D
Diamond Multimedia Stealth 3D 2000
Diamond Multimedia Stealth 3D 2000 PRO
Diamond SpeedStar (Plus)
Diamond SpeedStar 24
Diamond SpeedStar 24X (not fully supported)
Diamond SpeedStar 64
Diamond SpeedStar HiColor
Diamond SpeedStar Pro (not SE)
Diamond SpeedStar Pro 1100
Diamond SpeedStar Pro SE (CL-GD5430/5434)
Diamond SpeedStar64 Graphics 2000/2200
Diamond Stealth 24
Diamond Stealth 32
Diamond Stealth 3D 2000
Diamond Stealth 3D 2000 PRO
Diamond Stealth 3D 3000
Diamond Stealth 64 DRAM SE
Diamond Stealth 64 DRAM with S3 SDAC
Diamond Stealth 64 DRAM with S3 Trio64
Diamond Stealth 64 Video VRAM (TI RAMDAC)
Diamond Stealth 64 VRAM

Diamond Stealth Pro
Diamond Stealth Video 2500
Diamond Stealth Video DRAM
Diamond Stealth VRAM
Diamond Stealth64 Graphics 2001 series
Diamond Stealth64 Graphics 2xx0 series (864 + SDAC)
Diamond Stealth64 Graphics 2xx0 series (Trio64)
Diamond Stealth64 Video 2001 series (2121/2201)
Diamond Stealth64 Video 2120/2200
Diamond Stealth64 Video 3200
Diamond Stealth64 Video 3240/3400 (IBM RAMDAC)
Diamond Stealth64 Video 3240/3400 (TI RAMDAC)
Diamond Viper 330
Diamond Viper PCI 2Mb
Diamond Viper VLB 2Mb
DSV3325
DSV3326
EIZO (VRAM)
ELSA Gloria-4
ELSA Gloria-8
ELSA Victory 3D
ELSA Victory 3DX
ELSA VICTORY ERAZOR
ELSA WINNER 1000/T2D
ELSA Winner 1000AVI (AT&T 20C409 version)
ELSA Winner 1000AVI (SDAC version)
ELSA Winner 1000ISA
ELSA Winner 1000PRO with S3 SDAC
ELSA Winner 1000PRO with STG1700 or AT&T RAMDAC
ELSA Winner 1000PRO/X
ELSA Winner 1000TRIO
ELSA Winner 1000TRIO/V
ELSA Winner 1000TwinBus
ELSA Winner 1000VL
ELSA Winner 2000
ELSA Winner 2000AVI
ELSA Winner 2000AVI/3D
ELSA Winner 2000PRO-2
ELSA Winner 2000PRO-4
ELSA Winner 2000PRO/X-2
ELSA Winner 2000PRO/X-4
ELSA Winner 2000PRO/X-8
ELSA Winner 3000
ELSA Winner 3000-L-42
ELSA Winner 3000-M-22
ELSA Winner 3000-S

ET3000 (generic)
ET4000 (generic)
ET4000 W32i, W32p (generic)
ET4000/W32 (generic)
ET6000 (generic)
ET6100 (generic)
ExpertColor DSV3325
ExpertColor DSV3365
Generic VGA compatible
Genoa 5400
Genoa 8500VL(-28)
Genoa 8900 Phantom 32i
Genoa Phantom 64i with S3 SDAC
Genoa VideoBlitz III AV
Hercules Dynamite
Hercules Dynamite 128/Video
Hercules Dynamite Power
Hercules Dynamite Pro
Hercules Graphite HG210
Hercules Graphite Power
Hercules Graphite Pro
Hercules Graphite Terminator 64
Hercules Graphite Terminator 64/DRAM
Hercules Graphite Terminator Pro 64
Hercules Stingray
Hercules Stingray 128 3D
Hercules Stingray 64/V with ICS5342
Hercules Stingray 64/V with ZoomDAC
Hercules Stingray Pro
Hercules Stingray Pro/V
Hercules Terminator 3D/DX
Hercules Terminator 64/3D
Hercules Terminator 64/Video
Integral FlashPoint
Intel 5430
Interay PMC Viper
Jaton Video-58P
Jaton Video-70P
JAX 8241
Jazz Multimedia G-Force 128
LeadTek WinFast 3D S600
LeadTek WinFast 3D S680
LeadTek WinFast S200
LeadTek WinFast S430
LeadTek WinFast S510
Matrox Comet

Matrox Marvel II
Matrox Millennium (MGA)
Matrox Millennium II
Matrox Millennium II AGP
Matrox Mystique
MediaVision Proaxcel 128
MELCO WGP-VG4S
MELCO WGP-VX8
Mirage Z-128
Miro Crystal 10SD with GenDAC
Miro Crystal 12SD
Miro Crystal 16S
Miro Crystal 20SD PCI with S3 SDAC
Miro Crystal 20SD VLB with S3 SDAC (BIOS 3.xx)
Miro Crystal 20SD with ICD2061A (BIOS 2.xx)
Miro Crystal 20SD with ICS2494 (BIOS 1.xx)
Miro Crystal 20SV
Miro Crystal 22SD
Miro Crystal 40SV
Miro Crystal 80SV
Miro Crystal 8S
miro miroMedia 3D
Miro MiroVideo 20TD
Miro Video 20SV
NeoMagic (laptop/notebook)
Number Nine FX Motion 331
Number Nine FX Motion 332
Number Nine FX Motion 531
Number Nine FX Motion 771
Number Nine FX Vision 330
Number Nine GXE Level 10/11/12
Number Nine GXE Level 14/16
Number Nine GXE64
Number Nine GXE64 Pro
Number Nine GXE64 with S3 Trio64
Number Nine Imagine I-128 (2-8MB)
Number Nine Imagine I-128 Series 2 (2-4MB)
Number Nine Visual 9FX Reality 332
Oak (generic)
Ocean (octek) VL-VGA-1000
Orchid Celsius (AT&T RAMDAC)
Orchid Celsius (Sierra RAMDAC)
Orchid Fahrenheit 1280
Orchid Fahrenheit VA
Orchid Fahrenheit-1280+
Orchid Kelvin 64

Orchid Kelvin 64 VLB Rev A
Orchid Kelvin 64 VLB Rev B
Orchid P9000 VLB
Orchid Technology Fahrenheit Video 3D
Paradise Accelerator Value
Paradise/WD 90CXX
Rendition Verite 1000
RIVA128
S3 801/805 (generic)
S3 801/805 with ATT20c490 RAMDAC
S3 801/805 with ATT20c490 RAMDAC and ICD2061A
S3 801/805 with Chrontel 8391
S3 801/805 with S3 GenDAC
S3 801/805 with SC1148{2,3,4} RAMDAC
S3 801/805 with SC1148{5,7,9} RAMDAC
S3 864 (generic)
S3 864 with ATT 20C498 or 21C498
S3 864 with SDAC (86C716)
S3 864 with STG1703
S3 868 (generic)
S3 868 with ATT 20C409
S3 868 with ATT 20C498 or 21C498
S3 868 with SDAC (86C716)
S3 86C260 (generic)
S3 86C325 (generic)
S3 86C357 (generic)
S3 86C375 (generic)
S3 86C385 (generic)
S3 86C764 (generic)
S3 86C765 (generic)
S3 86C775 (generic)
S3 86C785 (generic)
S3 86C801 (generic)
S3 86C805 (generic)
S3 86C864 (generic)
S3 86C868 (generic)
S3 86C911 (generic)
S3 86C924 (generic)
S3 86C928 (generic)
S3 86C964 (generic)
S3 86C968 (generic)
S3 86C988 (generic)
S3 86CM65
S3 911/924 (generic)
S3 924 with SC1148 DAC
S3 928 (generic)

S3 964 (generic)
S3 968 (generic)
S3 Aurora64V+ (generic)
S3 Trio32 (generic)
S3 Trio64 (generic)
S3 Trio64V+ (generic)
S3 Trio64V2 (generic)
S3 Trio64V2/DX (generic)
S3 Trio64V2/GX (generic)
S3 ViRGE (generic)
S3 ViRGE (old S3V server)
S3 ViRGE/DX (generic)
S3 ViRGE/GX (generic)
S3 ViRGE/GX2 (generic)
S3 ViRGE/MX (generic)
S3 ViRGE/VX (generic)
S3 Vision864 (generic)
S3 Vision868 (generic)
S3 Vision964 (generic)
S3 Vision968 (generic)
SHARP 9080
SHARP 9090
Sierra Screaming 3D
Sigma Concorde
Sigma Legend
SiS SG86C201
SNI PC5H W32
SNI Scenic W32
SPEA Mercury 64
SPEA Mirage
SPEA/V7 Mercury
SPEA/V7 Mirage P64
SPEA/V7 Mirage P64 with S3 Trio64
SPEA/V7 Mirage VEGA Plus
SPEA/V7 ShowTime Plus
Spider Black Widow
Spider Black Widow Plus
Spider Tarantula 64
Spider VLB Plus
STB Horizon
STB Horizon Video
STB LightSpeed
STB LightSpeed 128
STB MVP-2
STB MVP-2 PCI
STB MVP-2X

STB MVP-4 PCI
STB MVP-4X
STB Nitro (64)
STB Nitro 3D
STB Nitro 64 Video
STB Pegasus
STB Powergraph 64
STB Powergraph 64 Video
STB Powergraph X-24
STB Systems Powergraph 3D
STB Systems Velocity 3D
STB Velocity 128
STB Velocity 64 Video
TechWorks Thunderbolt
Techworks Ultimate 3D
Toshiba Tecra 540CDT
Toshiba Tecra 550CDT
Toshiba Tecra 750CDT
Toshiba Tecra 750DVD
Trident 8900/9000 (generic)
Trident 8900D (generic)
Trident Cyber 9382 (generic)
Trident Cyber 9385 (generic)
Trident Cyber 9388 (generic)
Trident Cyber 9397 (generic)
Trident TGUI9400CXi (generic)
Trident TGUI9420DGi (generic)
Trident TGUI9430DGi (generic)
Trident TGUI9440 (generic)
Trident TGUI9660 (generic)
Trident TGUI9680 (generic)
Trident TGUI9682 (generic)
Trident TGUI9685 (generic)
Trident TVGA9200CXr (generic)
Unsupported VGA compatible
VI720
VideoLogic GrafixStar 300
VideoLogic GrafixStar 400
VideoLogic GrafixStar 500
VideoLogic GrafixStar 550
VideoLogic GrafixStar 600
VideoLogic GrafixStar 700
VidTech FastMax P20
ViewTop PCI
VL-41
WD 90C24 (laptop)

WD 90C24A or 90C24A2 (laptop)
WinFast 3D S600
WinFast S200
WinFast S430
WinFast S510
XGA-1 (ISA bus)
XGA-2 (ISA bus)

NAME

boot – bootstrap procedures for BSD/OS

DESCRIPTION

BSD/OS is normally booted by simply turning on the power to the computer and waiting. Under normal conditions, the floppy drive will be empty, so the BIOS will read and execute the boot block from the hard disk, which will read and execute the program `/boot`. Then, `/boot` will normally size memory, then read and execute the kernel, `/bsd`. Next, the kernel does a series of probes to determine what hardware is present, then executes the program `init`(8). Init then runs the shell script `/etc/rc`. This script checks the file systems, configures the network hardware and starts background system tasks. If it completes normally, init will spawn getty processes for the terminals listed in `/etc/ttys`, to allow users to log in. If `/etc/rc` exits with an error code, init will instead run a single user shell.

A number of these steps may be configured or modified during a single execution, or in configuration files.

The bootstrap normally uses the standard PC display and keyboard for output and input when needed. If no keyboard is present, and if carrier is present on the primary serial port, com0 (known as COM1 to DOS), the bootstrap "auto switches" to that port for input. It copies output to both possible consoles. For permanent use of a serial console, see the **-console** command below.

During the normal loading procedure, you will be presented with the option to press any key to interrupt the standard boot sequence. A counter will then start to count down backwards. If you strike a key during this countdown the normal boot sequence will be aborted and you will be prompted with:

```
Boot:
```

At this point, you can specify an alternate kernel to boot, potentially on a different device. If the kernel named has a suffix of `gz` (e.g., `bsd.gz`) it is assumed to be compressed by `gzip`(1) using the deflate compression method. The **boot** program will decompress the file as it is read. The default boot flags for the kernel are changed to request a boot to single user, possibly asking for a root filesystem location (see below).

The bootstrap checks the amount of memory in the system just before loading the kernel. On a PC, memory is in two sections. Base memory starts at location 0 and is usually 640K, although the BIOS may reserve some of this memory. Extended memory starts at 1 MB and continues upward.

The complete boot command is:

> *dev*(*adapter*, *controller*, *unit*, *partition*)*pathname* [**-adrsw** | *howto*]

or

> *-bootcommand*

If the command begins with `-` the line is treated as a boot command (see below), otherwise the line is used as a boot specification. The device specification (*dev* through the closing parenthesis) is optional; if omitted, the kernel is loaded from the device and partition from which the bootstrap was loaded. Otherwise, *dev* is the type of the desired device:

`bios`	any disk defined by the BIOS (A:, B:, C: or D:)
`fd`	floppy disk
`wd`	ST506/RLL/ESDI/IDE hard disks
`sd`	SCSI hard disks on `aha` or `bha` host adapter
`aha`	disk on Adaptec or Buslogic ISA SCSI host adapter
`eaha`	disk on Adaptec EISA SCSI host adapter

The values inside the parentheses should be numbers. If fewer than 4 arguments are provided, the left most (i.e. starting with *adapter*) will be defaulted to 0, e.g., `wd(1,2)` is the same as `wd(0,0,1,2)`. The arguments have the following general meanings. Specific device types use them somewhat differently as indicated.

adapter The adapter to use. Normally unused, but see below for special use with sd and bios types.

controller When more than a single controller is present, *controller* determines which controller to use. See below for special use with any SCSI controller.

unit A single controller may have more than a single disk assigned to it. The *unit* value selects which disk, 0 being the lowest disk number. See below for special use with any SCSI disk.

partition The *partition* determines which BSD/OS partition on the selected disk will be used. Partition 0 is the a partition, 1 is b and so on.

The BIOS device type uses the *adapter* field differently. If the adaptor is 0, the *unit* is a floppy unit (A: or B:); otherwise, *unit* is a hard disk known to the BIOS (C: or D:).

SCSI disks use the *adapter*, *controller* and *unit* fields slightly differently. When used with the `sd` device an *adapter* of 0 implies the use of an `aha` host adapter, and an *adapter* of 1 implies the use of an `eaha` host adapter. The `aha` and `eaha` device types are shorthand for this (and hence do not need the adapter number set). For `sd`, `aha` and `eaha` devices the *controller* number actually refers to the physical SCSI target ID (0-7, normally set with jumpers or switches on the device). The *unit* number actually refers to the logical disk number to which BSD/OS maps the device, i.e. the 1 in `sd1`. The *controller* determines which disk *pathname* will be loaded from while the *unit* determines which disk the kernel will be told to use as the root device.

The pathname identifies the desired file to be booted. If no device specification or pathname is present (that is, only a RETURN is typed), the default boot sequence is initiated. The default boot command is "`/bsd`".

The following options to the loaded kernel are available. With no options, the default is the same as specifying the **-as** options if the boot was interrupted, and no options if the bootstrap was automatic. If any options are present, only those present are actually used; a **-** alone explicitly disables all options.

-a Askname; if loading a generic kernel, the kernel will prompt for the name of the root filesystem device. If loading a bootstrap program, prompt for a boot command.

-d Enable debugging. Currently, this enables diagnostic output during the startup operations of the kernel.

-r Use the default (compiled-in) root filesystem rather than using the load device as the root.

-s Boot to a single-user shell rather than checking filesystems and going into multiuser operation. Also, if specified to the bootstrap using the **-bootflags** command (below) or implicitly by interrupting the boot sequence, the bootstrap prompts for commands rather than booting automatically.

-w Mount the root filesystem read/write rather than read-only (not yet implemented in the kernel).

The boot flags may also be set using the *howto* parameter, which is a decimal, octal or hex number that sets the boot flags– see the RB_* constants in `sys/reboot.h`. The use of the flags above is generally recommended rather than a numeric value.

If a bootstrap command line begins with **-**, it is treated as an internal boot command.

If the file `/etc/boot.default` exists, the `/boot` program reads this as a series of commands to be executed, one command per line, just as if they were typed from the keyboard. This can be used to boot the kernel from a different device than the BIOS uses to boot, or to change various parameters listed in the next section.

The commonly-used boot commands are the following:

-cat *filename*
: Display the contents of the file named by *filename*. For example: `-cat /etc/boot.default`

-changedisk
: Pause to request changing of floppy disks prior to execution of the kernel. This is useful when the root filesystem is not on the same floppy disk as the kernel.

-console *dev*
: Set the console to be *dev* for the bootstrap and for the kernel that is booted. Valid device names are:

  `kbd` The standard keyboard and display (the default if a keyboard is present or the com0 port does not have carrier asserted).

  `com` [*X* [*port* [*speed*]]]
  The serial device `com`*X* where *X* refers to the given port, default 0. Note that BSD/OS starts with `com0` where as DOS labels this `COM1`. The port address can be specified, or the standard PC values for com0 through com3 will be used. The speed defaults to 9600.

  `auto` If the bootstrap detects no keyboard and if the com0 port has carrier, it will "auto switch" to com0 as its input, copying output to both possible consoles. If this situation exists, `auto` causes the bootstrap to act as if `-console com` was used, causing the kernel to use the serial console; otherwise it has no effect. Note that when the bootstrap auto switches its output, it does not automatically direct the kernel to use the serial console; this must be done either with `-console com` or `-console` `auto`.

-dev *devname ioconf=value ...*

Reconfigure the device specified by *devname* to have the field specified by *ioconf* set to *value*. Wildcard parameters (such as *sd**) are overridden by specific parameters (such as *sd1*). If both wildcard and specific entries match a given device, only the specific parameters are used.

Valid *ioconf* fields are:

port	Base port address for device; if −1, the device is not probed.
iosiz	Number of ports used by device
maddr	Base address of memory mapped by device.
msize	Amount of memory mapped by device.
irq	The IRQ interrupt used by device; if the value is `IRQNONE`, no IRQ is used.
drq	The DRQ (DMA) channel used by device
flags	The flags field for the device

-help Display list of available commands.

-parm *recipient parameter=value*

Pass in a parameter for subsystem or device *recipient.* Typically the *recipient* field is used to specify a hardware device(s) such as *ncr1* or *ncr**. The value part of the *parameter=value* pair can be a comma separated list. For example:

```
-parm ncr0
disconnect=target1,target2
```

It is also possible to specify a class answer such as:

```
-parm ncr0 disconnect=all
```

When a class is used it is then possible to subtract one or more individuals from the class. An example of this would be:

```
-parm ncr0 disconnect=all-target1
```

or:

```
-parm ncr0 disconnect=all-
target1-target2
```

Specific legal values for the *parameter=value* pair are not discussed here because they are are specific to the *recipient*. These are discussed in the man page associated with the *recipient*.

If multiple **-parm** commands are issued with same *recipient* and *parameter*, but different *values*, the last one will be used. If an error is made entering a **-parm** command it can be correctly by simply re-entering it correctly.

No priority is given to *recipients* specified specifically over *recipients* specified with wild cards. Specific *recipients* should be entered after wild cards for them to override wild card specifications.

-pnpid *devname pnpid*

Assign an alternate *pnpid* for *devname*. Hardware vendors are constantly creating basically compatible devices with new Plug and Play ids. This command is used to tell the kernel that a new Plug and Play id is assigned to a device that a given driver knows how to handle. The *devname* specified applies to all units; which means that a tailing unit number should not be supplied. For instance a legal *devname* would be "com" while "com0" would be an illegal specification.

-rootdev devspec

Rather than passing the load device to the kernel for use as root device, pass the value of `devspec`, which has the same form as a device specification used when booting a kernel with no trailing pathname: *dev*(*adapter*, *controller*, *unit*, *partition*). For example, this may be used to load the kernel from a floppy disk, then instruct the kernel to use a SCSI CDROM as the root filesystem with `sr(0,0)` (`/dev/sr0a`).

The following boot commands are used to modify the level of output from autoconfiguration and for debugging problems:

-autodebug *flag ...*
Modify output and/or enable debugging of autoconfiguration in the kernel. The available values for *val* are:

-q Quieter output, suppresses addressing details.

-v Verbose output, prints additional device configuration information.

-d Print information about each device and location probed and the result, pausing after each screenful.

-a Print information as in **-d**, but confirm whether each device should be probed. This is useful if probing some device location causes a failure.

-p Page output if using the standard console (default with **-a**).

-bootdebug *val*
Enable debugging of the boot procedure. The available values for *val* are:

0 No debugging.

1 Print additional information about memory sizing and unusual events.

2 Very verbose messaging, depending on the device type used.

If the left shift key is held down while loading the bootstrap (that is, until `loading /boot` is printed), **-bootdebug** will be initialized to 1 instead of the default of 0. With some systems, this can cause a keyboard error message or a stuck key error message. If this happens, simply clear the error and then depress the shift key again.

The following boot commands are used primarily for debugging memory and cache problems:

-basemem *mem* Normally the amount of base memory is determined automatically. The **-basemem** command can be used to force the amount of base memory to the value specified by *mem*. Appending a K or M specifies an amount in kilobytes or megabytes respectively.

-cmosmem Limit the autosizing memory search to the amount of memory indicated by the CMOS.

-extendend *mem*
Some large memory machines may cause the memory sizing code to hang the machine. The **-extendend** command limits the amount of memory to be checked to *mem*. Appending a K or M specifies an amount in kilobytes or megabytes respectively.

-memsize *mem* Normally the end of extended memory is determined automatically. The **-memsize** command sets the size of extended memory to *mem*, disabling the automatic memory sizing test. Appending a K or M specifies an amount in kilobytes or megabytes respectively.

-noflushcache
While sizing memory, do not use the `wbinvd` instruction to flush the cache. This instruction hangs on some motherboards.

The following boot commands are used primarily in scripts in `/etc/boot.default`:

-apm Enable Advanced Power Management features.

-bootflags [**-adrsw** | *howto*]
Set the boot flags [**-adrsw** | *howto*]. Note that specifying the bootflags on the load line will override the bootflags set by **-bootflags**.

-default_kernel *file*
Specify an alternate default kernel pathname (the default is `/bsd`). This can be useful when a compressed kernel is used (by setting the default to `/bsd.gz`).

-echo [**-n**] *str*
Print *str* to the console. If **-n** is specified then do not append a newline to *str*.

-echoon Turn on echoing of commands.

-echooff Turn off echoing of commands.

-estopon Stop parsing from file on error. This is the default.

-estopoff Do no stop parsing from file on error.

-include *file*
Read boot commands from *file*.

-iomem [**enable** | **reserve**] *addr size*
Mark *size* bytes of io memory starting at *addr* as available (`enable`) or not available (`reserve`) This feature is normally only required when using PC Cards. By default, the following ranges are disabled:

```
0xa0000 0x10000
0xb0000 0x10000
0xc0000 0x10000
0xe0000 0x10000
0xf0000 0x10000
```

-ioport [**enable** | **reserve**] *base count*
Mark *count* io ports starting at *base* as available (`enable`) or not available (`reserve`). By default, the following range is disabled:

```
0x3b0 16
```

This feature is normally only required when using PC Cards or devices that fall into the default reserved range.

-irq [**enable** | **reserve**] *irq* [...]
Mark the specified *irq*'s as available (`enable`) or not available (`reserve`) for use by PC Cards.

-kdebug [**-i** | *val*]
Set kernel debugging flags. The **-i** flag requests the kernel attach to the kgdb port as soon as that port is attached. The flags can be specified numerically by specifying *val*. This would be used when an older `/boot` is used with a newer kernel which supports more than just the **-i** kdebug option.

-kernspace *mem*
Supply an estimate of how much memory will be used by the kernel. This value is used to calculate the size of the buffer cache, and can be useful on small memory machines (to make the buffer pool

smaller than the default). Appending a K or M specifies an amount in kilobytes or megabytes respectively. This option is primarily intended for use with installation floppies, reserving additional space for memory-based filesystems.

-load *file* [**-adrsw** | *howto*]
Load the program *file* (which may include a device specification), optionally setting the boot flags to [**-adrsw** | *howto*].

-pause *sec* Pause for up to *sec* seconds, counting the seconds down. Prompt the user to press any key to abort reading of the `/etc/boot.default` file. If the user presses a key and aborts the reading of `/etc/boot.default` , the boot flags are reset to **-as**.

-pccard_iowait [on | off]
Turn on or off the inserting of wait states when using PC Cards.

-ramdisk *size*
Create a ramdisk of *size* bytes at the end of memory. Currently only one ramdisk may be specified.

-ramdiskimage *file*
Load a ramdisk image from *file.* The ramdisk must first be allocated by the **-ramdisk** command.

-show Show the boot parameters that are currently set.

-start [**-adrsw** | *howto*]
Execute the program previously loaded with the **-load** command, optionally setting the boot flags to [**-adrsw** | *howto*].

-waitnl Wait for the user to press <ENTER> on the keyboard.

The boot instructions in a `/etc/boot.default` file can be used to override defaults (memory sizing, console device, or kernel name location). It should contain any **-console** or **-dev** commands first. If it then loads a kernel, it should include a **-pause** line first to allow the script to be interrupted. If the file does not load a kernel, the bootstrap will pause for 5 seconds after running the script to allow interruption, and will then load the default kernel if not interrupted.

BOOT HINTS

Here are a few common boot procedures. The sections that follow describe what is happening in more detail.

To boot a single user shell from the hard disk: make sure the floppy drive is empty, reset the processor, press any key when prompted, then type a RETURN at the Boot: prompt. You should usually run `fsck`(8) to check the file systems before doing anything else. If you type control-D, the system will come up multi-user. Don’t forget that the root is read-only at the start, so if you want to do maintenance in single-user mode, you need to do

```
      mount -u /
```

to enable writes.

To boot a backup kernel from the hard disk: make sure the floppy drive is empty, reset the processor, press any key when prompted, then type the desired kernel name at the Boot: prompt (e.g., `/bsd.good`). This will land you in a single user shell.

To boot BSD/OS from the hard disk, even though the FDISK label is set to boot DOS by default: insert a BSD/OS boot floppy and reset the processor. At the Boot: prompt, type “`wd()bsd -`” (replace `wd` with `sd` for a SCSI main disk on `aha` or `bha`, or `bios` for other types of SCSI disks) . Leave off the - if you want a single user shell. The floppy drive is now free, and can be used normally when the system finishes booting. (See below for creating a floppy that boots straight BSD/OS without any operator intervention.)

To boot with a CD-ROM root: insert a BSD/OS boot floppy and a BSD/OS CD-ROM and reset the processor. At the Boot: prompt, press RETURN. Type RETURN again at when offered the chance to change floppies. After kernel autoconfiguration, you will be prompted “`root device?`”. Type “`sd`*N*`a`” for a SCSI CD-ROM on SCSI drive *N*, or “`mcd0a`” for the Mitsumi ISA CD-ROM. The boot floppy is no longer needed, so you may insert another floppy if desired.

If there is a floppy in drive 0 when the processor is reset, then its boot block with be read and executed by the BIOS. The boot floppies included with the system use a `/etc/boot.default` script to control the loading process. A boot floppy can also be constructed that sets the boot flags to **-s**, always starting at the Boot: prompt. If you just type a RETURN, the default action is to boot /bsd from the floppy.

If you boot with a floppy root, then you must not remove the floppy while running; however, if you specify an alternate device, then the floppy is free and can be used for any purpose.

To make a floppy that boots straight to BSD/OS on a hard disk, make a copy of your distribution boot floppy, and in the copy, put a line like

```
wd(0,0)/bsd
```

in the file `/etc/boot.default`. Replace `wd` with `sd` if you have a SCSI main disk; change the unit numbers as needed.

BOOT DETAILS

The boot process on a PC consists of a number of stages.

The first stage is handled by the system BIOS. It selects a boot device, then loads and executes the first sector of the device. The device selection is sometimes configurable, but often defaults to the "A" floppy if a diskette is present, otherwise the "C" (primary) hard disk. If the disk is a hard disk, the first block may be a BSD/OS boot block, a standard PC master boot record, or a program like `bootany` (see `disksetup`(8)), which allows a choice of systems to be booted. In the latter two cases, a PC FDISK partition table in the first block lists the PC-style partitions and indicates which should normally be booted. Another bootstrap facility may also be used, such as the OS/2 boot manager. If a BSD/OS partition is selected, a BSD/OS boot block is then executed (usually one of `fdboot`, `wdboot`, `ahaboot`, or `eahaboot`). The boot block loads a slightly larger program from the following 15 sectors (usually one of `bootfd`, `bootwd`, `bootaha`, or `booteaha`). That program prints "`Loading /boot`" and then loads and executes `/boot` from the indicated partition. `/boot` then loads and executes the kernel, passing the boot flags and additional information to the kernel.

FILES

`/boot`	Second stage boot
`/etc/boot.define`	Defines symbols used by -parm command
`/sbin/init`	The first user program
`/etc/rc`	Startup shell script
`/etc/boot.default`	Default boot command

SEE ALSO

`bootany`(8), `disksetup`(8), `init`(8), `reboot`(8), `shutdown`(8)

ACKNOWLEDGEMENTS

The **boot** program incorporates de-compression code by the Info-ZIP group. There are no extra charges or costs due to the use of this code, and the original compression sources are freely available from Com-

puServe in the IBMPRO forum and by anonymous ftp from the Internet site ftp.uu.net:/pub/archiving/zip. The sources used in the **boot** program are also available from ftp.bsdi.com:/pub/bsdi/misc/unziplib.

NAME

bootany – install/update bootany master boot record

SYNOPSIS

bootany [**-dfinSz**] [**-A** *part*] [**-F** *bootfile*] [**-s** *savembr*] *disk0* [*disk1*]

DESCRIPTION

The **bootany** utility allows the installation of the bootany master boot record (MBR) or editing of the partitions from which to boot on *disk0*. The bootany MBR allows booting from up to 4 partitions on either of the two disks accessible through the BIOS. Although BSD/OS, starting with version 2, may be booted from either disk defined by the BIOS, other operating systems may not be able to be booted from other than the first disk.

disk0 should refer to the primary disk (e.g., /dev/rwd0c or /dev/rsd0c). In order to boot from the second drive, it must be specified as *disk1* (e.g., /dev/rwd1c).

With no options, **bootany** lists the current partitions from which to boot. The options available are:

-A Set the partition defined by *part*, which must be a number between 1 and 4, as the current active partition.

-d Treat the secondary drive as bootable with no FDISK table (i.e. it is dedicated to BSD/OS.) In this case the *disk1* argument may be omitted.

-f Print the FDISK tables as well as bootany tables.

-F Use *bootfile* as the path to the file which contains the bootany MBR instead of /usr/bootstraps/bootany.sys.

-i Ask for each of the partitions if they are bootable, and if so, what to call them.

-n Install a new copy of bootany.sys.

-s Save a copy of the old MBR in *savembr*.

-S Used during the express install. Do not ask any questions, simply intuit the answers.

-z Zero the bootany tables. If specified with the **-i** flag, the tables are first zeroed and then interactive mode is entered.

The **bootany** program is called by `disksetup`(8) to install the BSD/OS MBR.

If no partitions are marked as bootable then the **bootany** bootstrap will automatically try to boot what ever partition is marked active. It is undefined as to what happens if no partition is marked active or the active partition does not have a valid boot sector. In this case, the **bootany** bootstrap will still attempt to load and boot some random sector for the disk.

FILES

/usr/bootstraps/bootany.sys

SEE ALSO

`boot`(8), `disksetup`(8)

NAME

diskdefect – read/write standard bad sector information

SYNOPSIS

diskdefect [**-acnswvy**] [**-b** *bad-block-info*] [**-C** *copy-retries*] [**-g** *geometry-label*] [**-t** *replacement-type*] *disk* [*disk-snum* [*bad-block ...*]]

DESCRIPTION

Diskdefect can be used to inspect and/or change the information stored on a disk that is used by the disk drivers to implement bad sector forwarding. The **diskdefect** program is capable of using several different bad-block formats. At present, however, only the bad144(5) format and a special type called "null" are supported. The null format is never stored on disk, and can be used to run a simple reliability scan.

Available options:

-a The argument list (and the results of the disk scan if the **-s** or **-w** options were given) are new bad sectors to be added to an existing bad-block table. The new replacement sectors will be filled with zero bytes unless the **-c** or **-C** flags are used.

If invoked without the **-a** option **diskdefect** will overwrite the existing bad-block table and you must specify a new disk serial number to be stored with the bad-block table.

-b *bad-block-info*
Controls how the bad block map is handled.

For the bad144 format, *bad-block-info* may be a digit in 0 through 4; this specifies which copy of the bad block information to use. Only that copy is loaded and updated. The default is to search for a good header, and to update all the copies.

-c Forces an attempt to copy the old sector to the replacement. This may be useful when replacing an unreliable sector. Ten attempts will be made to read the original data.

-C *copy-retries*
Similar to **-c**, but allows the number of retries to be specified.

-g *geometry-label*
Allows **diskdefect** to proceed on an unlabeled disk. Rather than reading the disk geometry from the pack label,

diskdefect will use the specified type from disktab(5).

-t *replacement-type*
: Tells **diskdefect** to use the specified type of bad block replacement. By default, **diskdefect** will choose the correct type for the disk, but if desired, the default type can be overridden. Note that this has no effect on the underlying driver; using the wrong type of bad block replacement is not likely to do anything useful. Thus, **-t** really exists only to set the null replacement type.

-n Disables writing the bad-block table to disk (no write mode). This option is like the **-n** option to the make(1) and sh(1) programs. This is independent of the **-w** option.

-s Scans the disk and collects sectors that could not be read (or with the **-w** option, written). With the **-v** (verbose) option, **-s** will print a cylinder map with periods (.) for good cylinders and x's for bad cylinders and at the end will print a bad-block summary containing the logical sector number, cylinder, head, and sector of the error. When scanning sector by sector it spins the sequence '/', '-', '\', '|'.

-w Like **-s**, but rewrites each cylinder after reading it. (This tests the disk a little more thoroughly.)

WARNING: Use **-w** with caution and only on idle disks. Between reading data and writing it back out the system may have changed it.

-v The entire process is described as it happens in gory detail if **-v** (verbose) is given.

-y Don't ask any questions (assumes yes).

The **diskdefect** program is invoked by giving a device name (e.g. wd0, wd1, etc.). With no optional arguments it reads the current defect list of the corresponding disk and prints out the bad sector information. It issues a warning if the list is out of order. The **diskdefect** program may also be invoked with a serial number for the pack and a list of bad sectors. It will write the supplied information into all copies of the bad-sector file, replacing any previous information. Note, however, that **diskdefect** does not arrange for the specified sectors to be marked bad in this case. This procedure should only be used to restore known bad sector information which was destroyed.

It is no longer necessary to reboot to allow the kernel to reread the bad-sector table from the drive.

EXAMPLES

diskdefect -swcv wd0 12345678

Initialize the bad-block table to the result of the scan (erase any existing bad-block information). Set the disk serial number to 12345678. Attempt to copy the old data to the replacement sectors, if that fails zero the replacement sector.

diskdefect -a -swcv wd0

Scan the disk (read and write) and add any new bad sectors the the existing bad-block table. Attempt to copy the old data to the replacement sectors, if that fails zero the replacement sector.

SEE ALSO

`bad144`(5), `disktab`(5), `badsect`(8)

BUGS

It should be possible to format disks on-line under BSD.

It should be possible to mark bad sectors on drives of all types.

More thorough (but destructive) "worst case pattern" writes should be available.

HISTORY

The **diskdefect** command appeared in 4.1BSD as bad144(8). It has been merged with drck(8) at BSDI, and the name changed to be more intuitive and otherwise protect the innocent.

NAME

disksetup – setup and label disk drives

SYNOPSIS

disksetup -i [**-DEFKPns**] [**-I**] [**-II**] [**-A** *bootany*] [**-M** *memsize*] [*disk*]
disksetup [**-NW**] *disk*
disksetup [**-DKs**] [**-I**] [**-II**] *disk*
disksetup -R [**-DKns**] [**-I**] [**-II**] *disk proto-file* [*xxboot bootxx* [*mboot*]]
disksetup -e [**-DKns**] [**-I**] [**-II**] [*disk* [*xxboot bootxx* [*mboot*]]]
disksetup -B *disk* [*xxboot bootxx* [*mboot*]]

DESCRIPTION

The **disksetup** program is used to examine and alter the various disk labels on a disk drive specified by *disk*. If *disk* is indicated as optional, disksetup will either assume the only hard drive on the system is the target, or, if there is more than one drive, prompt for which drive should be used.

The **disksetup** utility can be called in a variety of ways. After a short description of the six major ways **disksetup** may be called, a combined list of options will be presented.

Method 1 – Interactive mode

When the **-i** flag is specified, **disksetup** will interactively setup the disk. See the section below on interactive setup.

Method 2 – Mark disk writable/not writable

Normally the second sector the disk (or BSD/OS FDISK partition) contains the BSD/OS disk label and is not writable. Using **disksetup** with the **-W** or **-N** flag makes this sector writable/not writable. When disksetup needs to write the BSD/OS disk label, it automatically enables the ability to write it.

Method 3 – Displaying existing labels

In this mode, disksetup outputs the existing label in ASCII, suitable for use as a *proto-file*.

Method 4 – Restoring disk label

When the **-R** flag is specified, **disksetup** will restore the label from the file specified by *proto-file*, which is usually generated using Method 3 (above). See the section below on installing boot blocks for information on the other optional arguments.

Method 5 – Editing disk label

When the **-e** flag is specified, a temporary ASCII version of the disklabel is created and an editor (either the contents of the EDITOR environment variable or vi(1)) is invoked on that file. After exiting the editor, **disksetup** will label the disk the with altered label. See the section below on installing boot blocks for information on the other optional arguments.

Method 6 – Install boot blocks.

When the **-B** flag is specified, **disksetup** will only install boot blocks, and possibly a Master Boot Record (MBR). An MBR is only needed when the disk is set up for co-residency using the DOS style FDISK table. The *xxboot* and *bootxx* arguments specify the primary and secondary bootstraps. The bootstraps are normally found in /usr/bootstraps. If the primary and secondary bootstraps are not specified, **disksetup** will prompt for the type of bootstrap to install. The *mboot* argument is used to specify a new MBR. The standard MBR used by BSD/OS is **bootany.sys**. If an FDISK table exists and *mboot* is not specified, disksetup will prompt for the file containing the MBR, with the option to keep the existing MBR. If **bootany.sys** is chosen as the MBR, the bootany(8) program will be run to install and initialize the MBR.

Flags and Options

-A *bootany*
: Use *bootany* as the path to the program to install **bootany.sys**.

-D
: Only read the on disk label.

-E
: Expert mode. Turn off some consistency checks, such as partition overlap, partition length, etc. Allow specification of the FDISK geometry. Only useful with the **-i** flag.

-F Edit the FDISK table. Implies the **-i** flag.

-I Ignore the on-disk BSD/OS disk label. Also causes the device to be open non-blocking.

-II In addition to the the effects of the **-I** flag, ignore any FDISK table on the disk.

-K Only read the kernel's (in-core) disk label.

-M *memsize*
In interactive mode, the amount of memory in the system is used to help determine the size of some partitions. When **-M** is supplied, *memsize* (specified in megabytes) will be used instead of the actual size of memory.

-N Disable writes to the BSD/OS label sector.

-P Edit the BSD/OS partition table. Implies the **-i** flag.

-Q Do not ask any questions when using the express setup.

-S Express mode setup. Used by the installation scripts.

-Z *dossize*
When using express setup. reserve *dossize* MB of disk space for DOS. A value of 0 (zero) implies no DOS partition.

-W Enable writes to the BSD/OS label sector.

-n Do not write any output to the disk.

-p *xxboot*
Use *xxboot* as the primary bootstrap. Used by express setup.

-q *bootxx*
Use *bootxx* as the secondary bootstrap. Used by express setup.

-s Write only the kernel's in-core label.

-t *path* Use by the installation script to specify a file prefix for storing filesystem information. This information is used by the installation script to generate `fstab`(5) entries.

Although **disksetup** attempts to be self explanatory (via prompts and help screens), it is useful to understand the basics of how the interactive mode works and the goals for configuring disks.

The **disksetup** utility will present a series of questions, with a default or recommended response enclosed in brackets [like this]. To use the default response, simply press the **<Enter>** key. If you enter an invalid response, **disksetup** will prompt you again until a valid response is received.

When requesting information about the geometry, or when editing FDISK and BSD/OS partition tables, **disksetup** uses a full screen display/edit mode. Use the **<Tab>** key to change fields. Generally the **<Esc>** key will cancel the current changes. When editing the partition tables, the **?** key may be pressed for context sensitive help. You may want to read through these help screens as they contain valuable tips on using **disksetup**.

FILES

/etc/disktab
/usr/bootstraps/*xx*boot
/usr/bootstraps/boot*xx*
/usr/bootstraps/bootany.sys

SEE ALSO

disktab(5), disklabel(5)

NAME

fsck – filesystem consistency check and interactive repair

SYNOPSIS

fsck **-p** [**-f**] [**-m** *mode*] [*filesystem*] *...*
fsck [**-ny**] [**-b** *block#*] [**-c** *level*] [**-l** *maxparallel*] [**-m** *mode*] [*filesystem*] *...*

DESCRIPTION

The first form of **fsck** preens a standard set of filesystems or the specified filesystems. It is normally used in the script /etc/rc during automatic reboot. Here **fsck** reads the table /etc/fstab to determine which filesystems to check. Only partitions in fstab that are mounted "rw," "rq" or "ro" and that have non-zero pass number are checked. Filesystems with pass number 1 (normally just the root filesystem) are checked one at a time. When pass 1 completes, all remaining filesystems are checked, running one process per disk drive. The disk drive containing each filesystem is inferred from the longest prefix of the device name that ends in a digit; the remaining characters are assumed to be the partition designator. In preening mode, filesystems that are marked clean are skipped. Filesystems are marked clean when they are unmounted, when they have been mounted read-only, or when **fsck** runs on them successfully.

The kernel takes care that only a restricted class of innocuous filesystem inconsistencies can happen unless hardware or software failures intervene. These are limited to the following:

Unreferenced inodes
Link counts in inodes too large
Missing blocks in the free map
Blocks in the free map also in files
Counts in the super-block wrong

These are the only inconsistencies that **fsck** with the **-p** option will correct; if it encounters other inconsistencies, it exits with an abnormal return status and an automatic reboot will then fail. For each corrected inconsistency one or more lines will be printed identifying the filesystem on which the correction will take place, and the nature of the correction. After successfully correcting a filesystem, **fsck** will print the number of files on that filesystem, the number of used and free blocks, and the percentage of fragmentation.

If sent a QUIT signal, **fsck** will finish the filesystem checks, then exit with an abnormal return status that causes an automatic reboot to fail. This is useful when you want to finish the filesystem checks during an automatic reboot, but do not want the machine to come up multiuser after the checks complete.

Without the **-p** option, **fsck** audits and interactively repairs inconsistent conditions for filesystems. If the filesystem is inconsistent the operator is prompted for concurrence before each correction is attempted. It should be noted that some of the corrective actions which are not correctable under the **-p** option will result in some loss of data. The amount and severity of data lost may be determined from the diagnostic output. The default action for each consistency correction is to wait for the operator to respond yes or no. If the operator does not have write permission on the filesystem **fsck** will default to a **-n** action.

Fsck has more consistency checks than its predecessors *check*, *dcheck*, *fcheck*, and *icheck* combined.

The following flags are interpreted by **fsck**.

-b Use the block specified immediately after the flag as the super block for the filesystem. Block 32 is usually an alternate super block.

-c Convert the filesystem to the specified level. Note that the level of a filesystem can only be raised. There are currently four levels defined:

0 The filesystem is in the old (static table) format.

1 The filesystem is in the new (dynamic table) format.

2 The filesystem supports 32-bit uid's and gid's, short symbolic links are stored in the inode, and directories have an added field showing the file type.

3 If maxcontig is greater than one, build the free segment maps to aid in finding contiguous sets of blocks. If maxcontig is equal to one, delete any existing segment maps.

In interactive mode, **fsck** will list the conversion to be made and ask whether the conversion should be done. If a negative answer is given, no further operations are done on the filesystem. In preen mode, the conversion is listed and done if possible without user interaction. Conversion in preen mode is best used when all the filesystems are being converted at once. The format of a filesystem can be determined from the first line of output from `dumpfs`(8).

-f Force **fsck** to check 'clean' filesystems when preening.

-l Limit the number of parallel checks to the number specified in the following argument. By default, the limit is the number of disks, running one process per disk. If a smaller limit is given, the disks are checked round-robin, one filesystem at a time.

-m Use the mode specified in octal immediately after the flag as the permission bits to use when creating the `lost+found` directory rather than the default 1777. In particular, systems that do not wish to have lost files accessible by all users on the system should use a more restrictive set of permissions such as 700.

-n Assume a no response to all questions asked by **fsck** except for `CONTINUE?`, which is assumed to be affirmative; do not open the filesystem for writing.

-p Preen filesystems (see above).

-y Assume a yes response to all questions asked by **fsck**; this should be used with great caution as this is a free license to continue after essentially unlimited trouble has been encountered.

If no filesystems are given to **fsck** then a default list of filesystems is read from the file `/etc/fstab`.

Inconsistencies checked are as follows:
1. Blocks claimed by more than one inode or the free map.
2. Blocks claimed by an inode outside the range of the filesystem.
3. Incorrect link counts.
4. Size checks:
   Directory size not a multiple of DIRBLKSIZ.
   Partially truncated file.
5. Bad inode format.
6. Blocks not accounted for anywhere.
7. Directory checks:
   File pointing to unallocated inode.
   Inode number out of range.
   Directories with unallocated blocks (holes).
   Dot or dot-dot not the first two entries of a directory or having the wrong inode number.

8. Super Block checks:
   More blocks for inodes than there are in the filesystem.
   Bad free block map format.
   Total free block and/or free inode count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the `lost+found` directory. The name assigned is the inode number. If the `lost+found` directory does not exist, it is created. If there is insufficient space its size is increased.

Because of inconsistencies between the block device and the buffer cache, the raw device should always be used.

FILES

`/etc/fstab` contains default list of filesystems to check.

DIAGNOSTICS

The diagnostics produced by **`fsck`** are fully enumerated and explained in Appendix A of *Fsck – The UNIX File System Check Program*.

SEE ALSO

`fstab`(5), `fs`(5), `newfs`(8), `reboot`(8)

NAME

installsw – install software packages to disk from installation media

SYNOPSIS

installsw [**-DELU**] [**-c** *path*] [**-d** *root_dir*] [**-h** *remote_host* [**-l** *remote_user*]] [**-m** *cdrom* | *floppy* | *tape*] [**-s** *d* | *o* | *r*]

DESCRIPTION

The **installsw** utility is an interactive program for installing software packages from floppy disk, tape, or CDROM to a filesystem. The floppy disk, tape and CDROM may be either on the local computer or on a remote host. In addition, **installsw** can be used to delete software packages that were previously installed.

If no command-line arguments are specified, **installsw** prompts the user for all necessary information.

The options are as follows:

-c The **-c** option specifies the mount point for installation from CDROM and floppy disks, and the name of the special device when installing from a tape drive.

-D The **-D** option specifies that selected packages should be deleted, not installed. As the programs that perform the deletion are part of the original software archive, this mode differs little from the installation mode of the program.

-d The **-d** option specifies all software packages are installed as if *root_dir* were the root of the filesystem, i.e. if no *root_dir* is specified, **installsw** behaves as if a *root_dir* of / was specified.

-E Treat the user as an expert, minimizing explanations and questions.

-h The **-h** option specifies that the installation is being done from the remote host *remote_host*, i.e. that the CDROM or floppy is mounted on a filesystem on the remote host, or that the tape drive belongs to the remote host.

-L The **-L** option specifies that the installation is being done from a local host, i.e. that the CDROM or floppy is mounted on a filesystem on the local machine, or that the tape drive is a device on the local machine.

-l By default, **installsw** uses the current user ID for all commands executed on the remote host. The **-l** option permits you to specify an alternate user ID for this purpose.

-m The **-m** option specifies the media type: the supported types are "cdrom", "floppy" and "tape".

-s The **-s** option specifies the packages to be installed, as follows:

r	Install the required packages.
d	Install the required and desirable packages.
o	Install the required, desirable and optional packages.
u	Install the packages which are candidates for updating.

If the **-s** option is specified, there is no further interaction with the user, and all necessary information must be specified on the command line, i.e., the **-s** option requires the **-c** and **-m** options. In addition, if the **-s** option is specified and the **-h** option is not specified, **installsw** expects the media to be available on the local machine, as specified by the **-c** option.

-U The **-U** option specifies that the user is doing an update, not an initial install. This causes **installsw** to pre-select packages that are candidates for upgrades in the initial screen, and to warn the user if they select a package which would not normally be installed during an update.

If the **-s** option was not specified, and any necessary information is not specified by command-line options, **installsw** first prompts the user for it.

All questions asked by **installsw** are shown as a question, a default (between square brackets), and then a colon. Replies are entered on the same line as the question. If the user types a return (with no other characters entered), the default is used. Otherwise, the user should specify a case-insensitive prefix of a possible alternative, e.g. "cd" and "CDROM" are both acceptable responses when "cdrom" is a possible answer. If **installsw** is unable to determine the answer intended by the user, the user is re-prompted for another answer to the same question.

To exit **installsw** at any time, enter an interrupt from the keyboard, usually `^C` (i.e. control-C, the key generated by entering the control key and the `C` key simultaneously). This should only be used if absolutely necessary once installation or deletion has started, as packages may be left partially installed or deleted.

Questions asked by **installsw** include:

1. The media type for the installation. The possible answers are "cdrom", "floppy" and "tape".
2. If the installation is from a remote host, or from the local computer. The possible answers are "local" and "remote".
3. If the installation is from a remote host, the hostname of the remote host, for example, "mysystem.bsdi.com". Note, the user is not prompted for a user ID to be used on the remote system. If the user wishes to use a user ID other than their own, the **-l** option should be specified.
4. The mount directory (for CDROM or floppy archives) or special device filename (for tape archives). If installation is from a cdrom or floppy disk, the pathname is the place where the cdrom or floppy is mounted, for example `/cdrom` or `/mnt`. If installation is from a tape drive, the name of the special device where the *no-rewind* tape drive can be found, for example `/dev/nrst0`.

The user is not prompted for a non-standard root directory. If the installation is not intended for the normal filesystem hierarchy, i.e. under `/`, the **-d** option should be specified.

During the question/response process, **installsw** will check the information given. For example, when a remote host is specified, **installsw** will attempt to contact the remote host. Or, if a CDROM on a remote host is specified, **installsw** will attempt to retrieve the ".MAP" file from the remote host. If these tests fail, the user will be reprompted for the information that caused the failure.

After the environment has been specified, **installsw** provides a full-screen interactive display of the packages that are available to be installed (or deleted). The name of each package is displayed preceded by three fields. The first field is a selection letter, from "a" to "z", and from "1" to "9". By entering this letter, the user can select/deselect an individual package.

The second field is a status letter. If the package has been selected, it is a "+" sign. Otherwise, it is blank.

The third field is a status letter, as follows:

*	The software package is already installed on the system.
D	The package is not installed, and is considered desirable.
R	The package is not installed, and is required for BSD/OS to run.

If the third field is empty, it means that the package is not currently installed, and is completely optional.

The commands that may be entered are:

A	Select all normally installed packages. If **installsw** is upgrading the system, (the **-U** option or the **-s** option with the "u" argument was specified), packages which should only be installed during initial installation will not be selected by the "A" command.
C	Deselect all selected packages.
D	Select/Deselect the "desirable" packages, i.e. those marked with an "D" in the status field. This command is a toggle: if it is used to add the desirable packages to the list of selected packages, entering it again will remove the desirable packages from the list of selected packages.
R	Select/Deselect the "required" packages, i.e. those marked with an "R" in the status field. This command is a toggle: if it is used to add the required packages to the list of selected packages, entering it again will remove the required packages from the list of selected packages.
N	Move to the next phrase, and attempt to either install or delete the selected packages.
U	Select/Deselect the packages which are candidates for updating. This command is a toggle: if it is used to add the update packages to the list of selected packages, entering it again will remove the update packages from the list of selected packages.
X	Exit **installsw** without further action.
?	Display information on a selected package. After entering "?", the user will be prompted for the letter field of the package for which information is desired. If information for that package is available, it will be displayed.
^L	Control-L (i.e., the key generated by entering the control key and the "L" key simultaneously) will refresh the screen.

If the user is deleting packages, i.e. the **-D** option was specified, the "D", "R" and "U" commands may not be entered.

If the user is not deleting packages and the **-U** option was specified, any packages on the system which are candidates for updating will be automatically selected before the initial selection screen is displayed.

If the user is not deleting packages and the **-U** option was not specified, any required and desirable packages not already present on the system will be automatically selected before the initial selection screen is displayed.

As packages are selected, the name of the package will be highlighted in reverse video (if the monitor supports that functionality), and a "+" sign will appear in the second field. As packages are deselected, package names will return to normal display and the "+" sign will disappear.

There is one other feature in the display. As packages are selected (or deselected) a running total of the kilobytes of disk space required by them is maintained in order to assist the user in deciding what packages to install (or which packages to delete to free the appropriate amount of disk space).

When selection is complete, **installsw** proceeds with package deletion or installation. As it is possible for the screen to be overwritten by other utilities, the "^L" key may continue to be used during installation to refresh the screen.

Various utilities are used by **installsw** to pre-process and extract software packages from the archives, and their normal and error output messages are displayed on the screen as they execute. For more information as to the meaning of these messages, see the referenced commands' manual pages.

If the file `/var/db/install.log` already exists or can be created in the root tree specified (i.e. /, or the path specified by the **-d** option), the attempt to delete or install a package is logged into that file. If the file is created, it will be created readable and writable by the owner and group (as modified by the user's umask).

The four possible log messages are as follows:

```
package <name> deleted <timestamp>
package <name> delete FAILED <timestamp>
package <name> installed <timestamp>
package <name> install FAILED <timestamp>
```

In all cases, "<name>" is the package name as specified by the second field of the archive file, and "<timestamp>" is the time of deletion or installation as returned by `ctime`(3).

Any errors occurring during package deletion or installation cause **installsw** to exit without further action, except that end_install scripts for any packages that were successfully installed will still be executed.

The **installsw** utility exits 0 on success, and >0 if an error occurs.

ARCHIVE FORMAT

The following information is intended to permit users to create their own software archives for use with the **installsw** utility.

If installing from a cdrom or floppy, the **installsw** utility reads the file `PACKAGES/PACKAGES`, which must exist in the directory specified by the user (either using the **-c** option or as a response to a prompt). If installing from a tape archive, **installsw** interprets the first file of the archive as this file. Each line of the file has eight, blank separated fields, which are interpreted as follows:

1. The number of the file. This field is used as the number of the file on in a tape archive, as well as the order of installation. The **installsw** utility will seek forward (or rewind and then seek forward) to move to the specified file on the tape. Any entries numbered 0 will be ignored.
2. The name of the file. This field is largely ignored by **installsw** with one exception, "INSTALLSW-SCRIPTS". See below for more information on these files.
3. The kilobytes of disk space required to install the software package.
4. The type of the file. Currently supported types are:

   N Normal.
   Z Compressed using the `compress`(1) utility.
   z Compressed using the `gzip`(1) utility.
5. The "preference" of the file. Currently supported preferences are:

   D Desirable.

   H Hidden. Any software package flagged as hidden will not be displayed to the user during package selection.

   I Initial. Any software package that should be installed only during an initial installation and not during an upgrade. If **installsw** is upgrading the system, (the **-U** option or the **-s** option with the "u" argument was specified), any software package marked as an initial package will be treated as already installed. If **installsw** is not upgrading the system, any software package marked as an initial package will be treated as a required package.

O	Optional.
R	Required.

6. The root of the installation for this software package.
7. A sentinel file that indicates that the software package has already been installed.

   For a software package to be flagged during package selection as already installed, the sentinel file must exist, there must be a reference to the package in the `/var/db/install.log` file, and the last reference in that file must be a record of its installation.

   For a software package to be flagged before and during package selection as a candidate for updating, the sentinel file must exist, and the last reference for the package in the `/var/db/install.log` file not be a record of its installation.

   Note, the `/var/db/install.log` file relative to the root tree specified (i.e. `/`, or the path specified by the **-d** option) is checked.

   As a special case, if installation of a package fails and the sentinel file did not exist before the installation was attempted, the sentinel file is removed.
8. A *short* description of the package for display to the user.

Once the file has been read in and interpreted, **installsw** takes special actions for an entry in the archive named "INSTALLSW-SCRIPTS". If this entry exists, it is immediately retrieved and extracted into the directory `/var/tmp/installsw.script`. It is expected to be a directory hierarchy, with subdirectories named by the second field of the entries in the PACKAGES file. There are six files that **installsw** treats specially when they are found in these subdirectories:

delete
: If this file exists, and the user specifies the **-D** option to **installsw**, the corresponding entry in the software archive is displayed to the user as part of package selection. If the user then selects the package, this file is executed. The program may read from stdin and write to stdout, including terminal control information. Any non-zero exit status by this script will be reflected in the final exit status of **installsw**.

information
: If this file exists, and the user requests information on a package by using the **?** key during package selection, its contents are displayed to the user. No text formatting or pre-processing

is performed before this display.

begin_install
: If this file exists, the user is doing package installation, and the user selects its corresponding entry during package selection, this file is executed after the user has finished package selection but before any packages have been installed. The program may read from stdin and write to stdout, including terminal control information. Any non-zero exit status by this script will stop packages from being installed, and will be reflected in the final exit status of **installsw**.

end_install
: If this file exists, the user is doing package installation, the user selects its corresponding entry during package selection, and the package is successfully installed, this file is executed after all selected packages have been installed. The program may read from stdin and write to stdout, including terminal control information. Any non-zero exit status by this script will be reflected in the final exit status of **installsw**.

pre_install
: If this file exists, the user is doing package installation, and the user selects its corresponding entry during package selection, this file is executed before installing the selected package. The program may write to stdout, but should not attempt to read from the terminal or send terminal control information. If this script exits non-zero, the software package is not installed, and the non-zero exit status will be reflected in the final exit status of **installsw**.

post_install
: If this file exists, the user is doing package installation, the user selects its corresponding entry during package selection, and the package is successfully installed, this file is executed after the package is installed. The program may write to the terminal, but should not attempt to read from the terminal or send terminal control information. Any non-zero exit status by this script will be reflected in the final exit status of **installsw**.

All of the programs are executed with the current directory set to the directory in which they reside. They may execute other programs, however, those programs must obey the same guidelines for accessing the terminal as they do.

Before any special files are executed, the environment variable `INSTROOT` is set to the root of the installation (i.e. `/`, or the path specified by the **-d** option).

If the user is installing software packages, once the user has completed the selection phase of **installsw**, the selected packages are installed in file number order. To install a selected package, the following steps occur:

1. If a pre-install script exists, it is run.
2. If the directory named by a path constructed of the user-specified root directory, suffixed by the root path of the software package (field 6) doesn't exist, it is created. Any intermediate directories created as part of this process are created with file permissions as specified by `mkdir`(1), using the **-p** option.
3. The **installsw** utility changes its current working directory to the directory specified in step 2.
4. If the software archive is being extracted from a cdrom or floppy filesystem, the archive file is named based on the type of the file as specified in field 4 of the PACKAGES file.

N	If the archive type is N, the file is named by field 2 followed by the suffix ".tar".
Z	If the archive type is Z, the file is named by field 2 followed by the suffix ".tar.Z".
z	If the archive type is z, the file is named by field 2 followed by the suffix ".tar.gz".

   If the software archive is being extracted from a tape, the archive file is retrieved based on the file number as specified in field 1 of the PACKAGES file.

   The archive file is appropriately pre-processed based on its type, and its contents extracted.
5. If a post-install script exists, it is run.

FILES

`/var/tmp/installsw.maptmp`
: The copy of the "MAP" file from the remote system.

`/var/tmp/installsw.packages`
: The copy of the "PACKAGES" file from the remote system.

`/var/tmp/installsw.status`
: The status of commands executed on both the local and remote host.

`/var/tmp/installsw.script`
: The directory where the "INSTALLSW-SCRIPTS" archive entry is ex-

tracted.

`/var/tmp/installsw.script/<package>/begin_install`
: The name of the software package script run before any deletion or installation.

`/var/tmp/installsw.script/<package>/end_install`
: The name of the software package script run after all deletion or installation.

`/var/tmp/installsw.script/<package>/delete`
: The name of the software package deletion script.

`/var/tmp/installsw.script/<package>/information`
: The name of the software package information file.

`/var/tmp/installsw.script/<package>/post_install`
: The name of the software package post-installation script.

`/var/tmp/installsw.script/<package>/pre_install`
: The name of the software package pre-installation script.

`<rootdir>/var/db/install.log`
: The file where successful installation or deletion is logged.

BUGS

Access to remote media by **installsw** requires that the current (or specified user) have remote access to the remote host.

The archive format is baroque and unwieldy.

SEE ALSO

compress(1), dd(1), mt(1), pax(1), zcat(1)

HISTORY

The *installsw* utility first appeared in BSD/OS release 0.9.3.

NAME

login.conf – login class capability data base

SYNOPSIS

/etc/login.conf

DESCRIPTION

The **login.conf** file describes the various attributes of login classes. A login class determines what styles of authentication are available as well as session resource limits and environment setup. While designed primarily for the `login`(8) program, it is also used by other programs, e.g., `ftpd`(8), to determine what means of authentication are available. It is also used by programs, e.g., `rexecd`(8), which need to set up a user environment.

CAPABILITIES

Refer to `getcap`(3) for a description of the file layout. All entries in the **login.conf** file are either boolean or use a '**=**' to separate the capability from the value. The types are described after the capability table.

Name	**Type**	**Default**	**Description**
alwaysuseklogin	bool	`false`	Always check the `.klogin` file for kerberos style authentication. Normally this file is only checked if a non-null kerberos instance is provided (e.g., `user.root`).
approve	program		Default program to approve login.
approve-*service*		program	Program to approve login for *service*.
auth	list	`passwd`	Allowed authentication styles. The first value is the default styles.

auth-*type*	list		Allowed authentication styles for the authentication type *type*.
classify	program		Classify type of login.
copyright	file		File containing additional copyright information.
coredumpsize	size		Maximum coredump size limit.
cputime	time		CPU usage limit.
datasize	size		Maximum data size limit.
filesize	size		Maximum file size limit.
harduserlimit	bool	`false`	Do not allow the user to log in if this session will be beyond the licensed number of simultaneous users.
hushlogin	bool	`false`	Same as having a `$HOME/.hushlogin` file. See `login`(8).
ignorenologin	bool	`false`	Not affected by `nologin` files. See `login`(8).
stacksize	size		Maximum stack size limit.
maxproc	number		Maximum number of process.
memorylocked	size		Maximum locked in core memory size limit.

memoryuse	size		Maximum in core memoryuse size limit.
minpasswordlen	number	6	The minimum length a local password may be. Used by the `passwd(1)` utility.
nologin	file		If the file exists it will be displayed and the login session will be terminated.
openfiles	number		Maximum number of open files per process.
password-dead	time	0	Length of time a password may be expired but not quite dead yet. When set (for both the client and remote server machine when doing remote authentication), a user is allowed to log in just one more time after their password (but not account) has expired. This allows a grace period for updating their password.
password-warn	time	2w	If the user's password will expire within this length of time then warn the user of this.
path	path	`/bin /usr/bin`	Default search path.
priority	number		Initial priority (nice) level.

requirehome	bool	`false`	Require home directory to login.
shell	program		Session shell to execute rather than the shell specified in the password file. The `SHELL` environment variable will contain the shell specified in the password file.
term	string	`su`	Default terminal type if not able to determine from other means.
umask	number	`022`	Initial umask. Should always have a leading `0` to assure octal interpretation. See `umask`(2).
welcome	file	`/etc/motd`	File containing welcome message.
widepasswords	bool	`false`	Use the new wide password format when using the `passwd`(1) utility. The wide password format allows up to 128 significant characters in the password.

The resource limit entries (*cputime*, *filesize*, *datasize*, *memoryuse*, *memorylocked*, *maxproc*, and *openfiles*) actually specify both the maximum and current limits (see `getrlimit`(2)). The current limit is the one normally used, although the user is permitted to increase the current limit to the maximum limit. The maximum and current limits may be specified individually by appending a *−max* or *−cur* to the capability name (e.g., *openfiles-max* and *openfiles-cur*).

BSDI BSD/OS will never define capabilities which start with `x−` or `X−`, these are reserved for external use (unless included through contributed software).

The argument types are defined as:

file	Path name to a text file.
list	A comma separated list of values.
number	A number. A leading `0x` implies the number is expressed in hexadecimal. A leading `0` implies the number is expressed in octal. Any other number is treated as decimal.
path	A space separated list of path names. If a ~ is the first character in the path name, the ~ is expanded to the user's home directory.
program	A path name to program.
size	A *number* which expresses a size in bytes. It may have a trailing `b` to multiply the value by 512, a `k` to multiply the value by 1 K (1024), and a `m` to multiply the value by 1 M (1048576).
time	A time in seconds. A time may be expressed as a series of numbers which are added together. Each number may have a trailing character to represent time units:

y	Indicates a number of 365 day years.
w	Indicates a number of 7 day weeks.
d	Indicates a number of 24 hour days.
h	Indicates a number of 60 minute hours.
m	Indicates a number of 60 second minutes.
s	Indicates a number of seconds.

For example, to indicate 1 and 1/2 hours, the following string could be used: `1h30m`.

AUTHENTICATION

BSDI BSD/OS uses BSD Authentication, which is made up of a variety of authentication styles. The authentication styles currently provided are:

`activ`	Authenticate using an ActivCard token. See `login_activ`(8).
`auth`	Authenticate using the remote authentication protocol. Normally linked to another authentication type. See

login_auth(8).

chpass	Change user's password. See login_chpass(8).
crypto	Authenticate using a CRYPTOCard token. See login_crypto(8).
kerberos	Request a password and use it to request a ticket from the kerberos server. See kerberos(1).
krb-or-pwd	Request a password and first try the kerberos authentication style and if that fails use the same password with the passwd authentication style. See kerberos(1).
lchpass	Change user's local password. See login_chpass(8).
passwd	Request a password and check it against the password in the master.passwd file.
radius	Normally linked to another authentication type, contact the radius server to do authentication. See login_radius(8).
rchpass	Change user's rpasswd password. See login_rchpass(8).
reject	Request a password and reject any request. See login_reject(8).
rpasswd	Request a password and check it against the password in the rpasswd.db file.
skey	Send a challenge and request a response, checking it with S/Key™ authentication. See skey(1).
snk	Authenticate using a SecureNet Key token. See login_snk(8).
token	Authenticate using a generic X9.9 token. See login_token(8).

Local authentication styles may be added by creating the login script for the style (see below). To prevent collisions with future official BSD Authentication style names all local style names should start with a dash (-). Current plans are for all official BSD Authentication style names to begin with a lower case alphabetic character. For example, if you have a new style you refer to as `slick` then you should create an authentication script named `/usr/libexec/login_-slick` using the style name `-slick`. When logging in via the `login(8)` program, the syntax *user*`:-slick` would be used.

Authentication requires several pieces of information:

class	The login class being used.
service	The type of service requesting authentication. The service type is used to determine what information the authentication program can provide to the user and what information the user can provide to the authentication program. The service type `login` is appropriate for most situations. Two other service types, `challenge` and `response`, are provided for use by programs like `ftpd`(8) and `radiusd`(8). If no service type is specified, `login` is used.
style	The authentication style being used.
type	The authentication type, used to determine the available authentication styles.
username	The name of the user to authenticate. The name may contain an instance, e.g. "user.root", as used by Kerberos authentication. If the authentication style being used does not support such instances, the request will fail.

The program requesting authentication must specify a username and an authentication style. (For example, `login`(8) requests a username from the user. Users may enter usernames of the form "user:style" to optionally specify the authentication style.) The requesting program may also specify the type of authentication that will be done. Most programs will only have a single type, if any at all, i.e. `ftpd`(8) will always request the `ftp` type authentication, and `su`(1) will always request the `su` type authentication. The `login`(8) utility is special in that it may select an authentication type based on information found in the `/etc/ttys` file for the appropriate tty (see `ttys`(5)).

The class to be used is normally determined by the `class` field in the password file (see `passwd`(5)).

The class is used to look up a corresponding entry in the `login.conf` file. If an authentication type is defined and a value for `auth-`*type* exists in that entry, it will be used as a list of potential authentication styles. If an authentication type is not defined, or `auth-`*type* is not specified for the class, the value of `auth` is used as the list of available authentication styles.

If the user did not specify an authentication style the first style in the list of available styles is used. If the user did specify an authentication style and the style is in the list of available styles it will be used, otherwise the request is rejected.

For any given style, the program `/usr/libexec/login_`*style* is used to perform the authentication. The synopsis of this program is:

`/usr/libexec/login_`*style* [**-v** *name=value*] [**-s** *service*] *username class*

The **-v** option is used to specify arbitrary information to the authentication programs. Any number of **-v** options may be used. The `login`(8) program provides the following through the **-v** option:

`auth_type`	The type of authentication to use.
`fqdn`	The hostname provided to login by the **-h** option.
`hostname`	The name `login`(8) will place in the utmp file for the remote hostname.
`local_addr`	The local ip address given to `login`(8) by the **-L** option.
`remote_addr`	The remote ip address given to `login`(8) by the **-R** option.
`style`	The style of authentication used for this user (see approval scripts below).

The `su`(1) program provides the following through the **-v** option:

`wheel`	Set to either "yes" or "no" to indicate if the user is in group wheel when they are trying to become root. Some authentication types require the user to be in group wheel when using the `su`(1) program to become super user.

When the authentication program is executed, the environment will only contain the values PATH=`/bin:/usr/bin` and SHELL=`/bin/sh`. File descriptor 3 will be open for reading and writing. The authentication program should write one or more of the following strings to this file descriptor:

`authorize` The user has been authorized.

`authorize secure`
The user has been authorized and root should be allowed to login even if this is not a secure terminal. This should

only be sent by authentication styles that are secure over insecure lines.

`reject` Authorization is rejected. This overrides any indication that the user was authorized (though one would question the wisdom in sending both a *reject* and an *authorize* command).

`reject challenge`
Authorization was rejected and a challenge has been made available via the value `challenge`.

`reject silent`
Authorization is rejected, but no error messages should be generated.

`remove` *file*
If the login session fails for any reason, remove *file* before termination (a kerberos ticket file, for example).

`setenv` *name value*
If the login session succeeds, the environment variable *name should be set* to the specified *value*.

`value` *name value*
Set the internal variable *name* to the specified *value*. The *value* should only contain printable characters. Several \ sequences may be used to introduce non printing characters. These are:

`\n`	A newline
`\r`	A carriage return
`\t`	A tab
`\`*xxx*	The character represented by the octal value *xxx*. The value may be one, two, or three octal digits.
`\`*c*	The string is replaced by the value of *c*. This allows quoting an initial space or the \ character itself.

The following values are currently defined:

`challenge`
See section on challenges below.

`errormsg`
If set, the value is the reason authentication failed. The calling program may choose to dis-

play this when rejecting the user, but display is not required.

In order for authentication to be successful, the authentication program must exit with a value of 0 as well as provide an `authorize` or `authorize root` statement on file descriptor 3.

An authentication program must not assume it will be called as root, nor must it assume it will not be called as root. If it needs special permissions to access files it should be setuid or setgid to the appropriate user/group. See `chmod`(1).

CHALLENGES

When an authentication program is called with a service of `challenge` it should do one of three things:

If this style of authentication supports challenge response it should set the internal variable `challenge` to be the appropriate challenge for the user. This is done by the `value` command listed above. The program should also issue a `reject challenge` and then exit with a 0 status. See the section on responses below.

If this style of authentication does not support challenge response, but does support the `response` service (described below) it should issue `reject silent` and then exit with a 0 status.

If this style of authentication does not support the `response` service it should simply fail, complaining about an unknown service type. It should exit with a non-zero status.

RESPONSES

When an authentication program is called with a service of `response`, and this style supports this mode of authentication, it should read two null terminated strings from file descriptor 3. The first string is a challenge that was issued to the user (obtained from the `challenge` service above). The second string is the response the user gave (i.e., the password). If the response is correct for the specified challenge, the authentication should be accepted, else it should be rejected. It is possible for the challenge to be any empty string, which implies the calling program did first obtain a challenge prior to getting a response from the user. Not all authentication styles support empty challenges.

APPROVAL

An approval program has the synopsis of:

approve [**-v** *name*=*value*] *username class service*

Just as with an authentication program, file descriptor 3 will be open for writing when the approval program is executed. The **-v** option is the same as in the authentication program. Unlike an authentication program, the approval program need not explicitly send an `authorize` or `authorize root` statement, it only need exit with a value of 0 or non-zero. An exit value of 0 is equivalent to an `authorize` statement, and non-zero to a `reject` statement. This allows for simple programs which have no information to provide other than approval or denial.

CLASSIFICATION

A classify program has the synopsis of:

classify [**-v** *name=value*] [**-f**] [user]

See `login`(8) for a description of the **-f**, option. The **-v** option is the same as for the authentication programs. The *user* is the username passed to `login`(8) login, if any.

The typical job of the classify program is to determine what authentication type should actually be used, presumably based on the remote IP address. It might also re-specify the hostname to be included in the `utmp`(5) file, reject the login attempt out right, or even print an additional login banner (e.g., `/etc/issue`).

The classify entry is only valid for the `default` class as it is used prior to knowing who the user is. The classify script may pass environment variables or other commands back to `login`(8) on file descriptor 3, just as an authentication program does. The two variables **AUTH_TYPE** and **REMOTE_NAME** are used to specify a new authentication type (the type must have the form `auth- Na` *type*) and override the **-h** option to login, respectively.

SEE ALSO

`authenticate`(3), `bsd_auth`(3), `getcap`(3), `login_cap`(3), `ttys`(5), `ftpd`(8), `login`(8)

NAME

newfs, **mount_mfs** – construct a new file system

SYNOPSIS

newfs [**-NO**] [**-S** *sector-size*] [**-T** *disktype*] [**-a** *maxcontig*] [**-b** *block-size*] [**-c** *cylinders*] [**-d** *rotdelay*] [**-e** *maxbpg*] [**-f** *frag-size*] [**-i** *bytes*] [**-k** *skew*] [**-l** *interleave*] [**-m** *free space*] [**-n** *rotational positions*] [**-o** *optimization*] [**-p** *sectors*] [**-r** *revolutions*] [**-s** *size*] [**-t** *tracks*] [**-u** *sectors*] [**-x** *sectors*] *special*

mount_mfs [**-N**] [**-S** *sector-size*] [**-T** *disktype*] [**-a** *maxcontig*] [**-b** *block-size*] [**-c** *cylinders*] [**-d** *rotdelay*] [**-e** *maxbpg*] [**-f** *frag-size*] [**-i** *bytes*] [**-m** *free space*] [**-n** *rotational positions*] [**-o** *options*] [**-s** *size*] [**-t** *tracks*] [**-u** *sectors*] [*special*] *node*

DESCRIPTION

Newfs replaces the more obtuse mkfs(8) program. Before running **newfs** or **mount_mfs**, the disk must be labeled using disksetup(8). **Newfs** builds a file system on the specified special device basing its defaults on the information in the disk label. Typically the defaults are reasonable, however **newfs** has numerous options to allow the defaults to be selectively overridden.

Mount_mfs is used to build a file system in virtual memory and then mount it on a specified node. **Mount_mfs** exits and the contents of the file system are lost when the file system is unmounted. If **mount_mfs** is sent a signal while running, for example during system shutdown, it will attempt to unmount its corresponding file system. The parameters to **mount_mfs** are the same as those to **newfs**. If the **-T** flag is specified (see below), the special file is unused. Otherwise, it is only used to read the disk label which provides a set of configuration parameters for the memory based file system. The special file is typically that of the primary swap area, since that is where the file system will be backed up when free memory gets low and the memory supporting the file system has to be paged. If neither a disktype nor a special file are specified, a default configuration is used. This default configuration has a sector size of 512 bytes, 32 sectors per track, and 16 heads, for a total of 256K bytes per cylinder.

The following options define the general layout policies.

-N Causes the file system parameters to be printed out without really creating the file system.

-O Creates a 4.3BSD format filesystem. This options is primarily used to build root filesystems that can be understood by older boot ROMs.

-T Uses information for the specified disk from `/etc/disktab` instead of trying to get the information from a disklabel.

-a *maxcontig*
This specifies the maximum number of contiguous blocks that will be laid out before forcing a rotational delay (see the **-d** option). The default value is one. See `tunefs`(8) for more details on how to set this option.

-b *block-size*
The block size of the file system, in bytes.

-c *#cylinders/group*
The number of cylinders per cylinder group in a file system. The default value is 16.

-d *rotdelay*
This specifies the expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. The default is 4 milliseconds. See `tunefs`(8) for more details on how to set this option.

-e *maxbpg*
This indicates the maximum number of blocks any single file can allocate out of a cylinder group before it is forced to begin allocating blocks from another cylinder group. The default is about one quarter of the total blocks in a cylinder group. See `tunefs`(8) for more details on how to set this option.

-f *frag-size*
The fragment size of the file system in bytes.

-i *number of bytes per inode*
This specifies the density of inodes in the file system. The default is to create an inode for each four fragments of data space, 2048 on a 512/4K filesystem and 4096 on a 1K/8K filesystem. If fewer inodes are desired, a larger number should be used; to create more inodes a smaller number

should be given.

-m *free space %*
: The percentage of space reserved from normal users; the minimum free space threshold. The default value used is 10%. See `tunefs`(8) for more details on how to set this option.

-n *rotational positions*
: Determines how many rotational time slots there are in one revolution of the disk.

-o *optimization preference*
: ("space" or "time") The file system can either be instructed to try to minimize the time spent allocating blocks, or to try to minimize the space fragmentation on the disk. If the value of minfree (see above) is less than 10%, the default is to optimize for space; if the value of minfree is greater than or equal to 10%, the default is to optimize for time. See `tunefs`(8) for more details on how to set this option.

-s *size* The size of the file system in sectors.

The following options override the standard sizes for the disk geometry. Their default values are taken from the disk label. When using **newfs**, changing these defaults is only useful when building a file system whose raw image will eventually be used on a different type of disk than the one on which it is initially created (for example on a write-once disk). Note that changing any of these values from their defaults will make it impossible for `fsck` to find the alternate superblocks if the standard superblock is lost.

-S *sector-size*
: The size of a sector in bytes (almost never anything but 512).

-k *sector 0 skew, per track*
: Used to describe perturbations in the media format to compensate for a slow controller. Track skew is the offset of sector 0 on track N relative to sector 0 on track N-1 on the same cylinder.

-l *hardware sector interleave*
: Used to describe perturbations in the media format to compensate for a slow controller. Interleave is physical sector interleave on each track, specified as the denominator of the ratio:

```
     sectors read/sectors passed over
```

Thus an interleave of 1/1 implies contiguous layout, while 1/2 implies logical sector 0 is separated by one sector from logical sector 1.

-p *spare sectors per track*
: Spare sectors (bad sector replacements) are physical sectors that occupy space at the end of each track. They are not counted as part of the sectors/track (**-u**) since they are not available to the file system for data allocation.

-r *revolutions/minute*
: The speed of the disk in revolutions per minute.

-t *#tracks/cylinder*
: The number of tracks/cylinder available for data allocation by the file system.

-u *sectors/track*
: The number of sectors per track available for data allocation by the file system. This does not include sectors reserved at the end of each track for bad block replacement (see the **-p** option.)

-x *spare sectors per cylinder*
: Spare sectors (bad sector replacements) are physical sectors that occupy space at the end of the last track in the cylinder. They are deducted from the sectors/track (**-u**) of the last track of each cylinder since they are not available to the file system for data allocation.

The options to the **mount_mfs** command are as described for the **newfs** command, except for the **-o** option.

This option is as follows:

-o Options are specified with a **-o** flag followed by a comma separated string of options. See the mount(8) man page for possible options and their meanings.

SEE ALSO

disktab(5), fs(5), dumpfs(8), diskpart(8), disksetup(8), fsck(8), mount(8), tunefs(8)

M. McKusick, W. Joy, S. Leffler, and R. Fabry, "A Fast File System for UNIX,", *ACM Transactions on Computer Systems 2*, 3, pp 181-197, August 1984, (reprinted in the BSD System Manager's Manual).

HISTORY

The **newfs** command appeared in 4.2BSD.

NAME

scsicmd – SCSI disk labeller and diagnostic

SYNOPSIS

scsicmd [**-f** *device*] **-l** [**-Z**]
scsicmd **-a** *asc/ascq*
scsicmd [**-T** *command-table*] [**-f** *device*] **-c** *command* [**-P** *parameter=value, . . .*] [**-p** *parameter, . . .*] [**-v** *parameter, . . .*] [**-dx**]

DESCRIPTION

The first form of the **scsicmd** command is used to generate a default disk label for a SCSI disk, sending requests to the drive to determine its geometry. The second form provides a convenient translation service for ASC and ASCQ codes. The last form is used to run SCSI commands on a specified device as a diagnostic. With the last form, options may be repeated and their order is significant. The options are as follows:

-a For a given ASC or ASC/ASCQ code, print an English translation from the SCSI-2 standard. The ASC should be a number in C notation (decimal, octal or hex); if an ASCQ is provided, it should be appended to the ASC after a slash. This service is now provided by the system, so ASC translation is probably unneeded now.

-c Set the current SCSI *command*. Parameter lists which follow are interpreted in the context of this command. The special command *all* lists all of the available commands (in parentheses following their descriptions); it does not set the command context. The **-c** flag may be repeated to change the command context.

-d Print numeric parameters in decimal (the default). This flag may be repeated to change the output radix between **-p** operations.

-f Open the named *device* file for SCSI operations. When repeated, the current device is closed and the new one is opened. If the environment variable SCSI is set, the program performs an implicit **-f** *$SCSI* when it starts up.

-l Probe a SCSI disk for its geometry parameters and send an ASCII version of a disk label to the standard output. **Scsicmd** is generally very conservative in its assumptions, especially when dealing with cylinder and sector/track counts. Commentary on the progress of probing appears on the standard error.

If **scsicmd** can't elucidate particular drive parameters, it will prompt for them.

-P Set *parameters* to *values*. Parameters are named fields in SCSI data structures. A *parameter=value* pair sets a field in a SCSI data structure to the specified value, given in decimal, hex or octal using C notation. Values wider than a byte are translated to big-endian byte order. High bits are truncated to make values fit. If the *=value* is omitted, a value of 1 is assumed. Multiple parameter/value pairs may be entered separated by commas. Certain fields are specified in the SCSI standard as containing strings rather than numbers; the value for such a parameter is treated as a string.

-p Execute a command and print the resulting parameters using the comma-separated parameter list. The special parameter *all* prints all of the named fields associated with the current command. The special parameter *none* inhibits printing; this is useful for commands which don't return any data. A parameter which matches the name of a buffer page (see below) will cause parameters in that page to be printed. If no **-p** or **-v** flag is given, **scsicmd** performs an implicit **-p** *none* after processing all its arguments.

-T Choose a different command table file. The default is /usr/share/misc/scsicmdtab. **Scsicmd** reads this table to get its list of SCSI commands and associated structures. A detailed description of this file appears below.

-v Verify fields without executing a command. This is useful for getting lists of parameter or buffer page names. The special parameter *all* prints all of the named fields associated with the current command. The special parameter *pages* prints just the names of the individual buffer pages. A parameter which matches the name of a buffer page will cause parameters in that page to be printed.

-x Print numeric fields in hexadecimal. This flag may be alternated with **-d** to change printing format dynamically.

-Z By default on zone recorded disks, **scsicmd** generates labels that use only a single (huge) track per cylinder. Since the system can't optimize the rotational layout of these disks, these labels do the next best thing and take advantage of the buffering in most modern disks. It is *not* necessary for the geometry in the label to bear a direct relationship to the physical geometry of the disk, although this is desirable on disks with a fixed geometry (not zone recorded). The **-Z** flag causes **scsicmd** to generate labels using the median cylinder size, well as print the sizes of each zone. The scanning process that **scsicmd** uses with **-Z** has been known to hang disks, and using the median cylinder size can produce unreasonable rotational layouts on most cylinders of a zone recorded disk, so the **-Z** flag is not recommended.

When **scsicmd** is used as a diagnostic, its arguments are treated as a small interpreted language. A **-c** flag sets the context, then subsequent **-P** and **-p** flags set and obtain values, respectively. More than one **-c** flag may be used so that more than one SCSI command can be issued with a single invocation of **scsicmd**. Arguments are processed sequentially from left to right.

Each command is associated with one or more named buffer pages. Buffer pages contain the bits which are sent to and retrieved from the SCSI device. The SCSI CDB is always the first buffer page and always has the name `cdb`. Other pages are specific to the command. Some pages have a variable length that depends on an internal field returned from the device; **scsicmd** will not print fields which are outside the page length.

Each buffer page contains a number of named fields. **Scsicmd** inserts and extracts values from these fields and provides names and descriptions. Most fields contain numbers; these are printed in decimal or hex. Some fields contain strings, which are stripped of trailing blanks and otherwise printed literally. Some numeric fields are translated into English phrases for clarity; for example, sense keys are looked up in a table so that a value of "5" is printed as "illegal request". Field names and page names are generally constructed by using either the abbreviation chosen by the SCSI standard or an acronym built from the first letters of a multi-word description in the standard.

Scsicmd handles one pair of SCSI commands specially. The MODE SENSE (`msen`) and MODE SELECT (`msel`) commands share buffer pages, to make it easy to get a page, modify it, and write it back out. The MODE SELECT code reprograms a few fields in the MODE SENSE buffer to make them appropriate for MODE SELECT. Specifically, the operation code in the CDB is set to MODE SELECT, and the `dbd`, `pc`, `pcode` and `mdl` fields are cleared, while the `pf` field is set to 1 if `pcode` was previously set, and the `length` field is set to the previous value of the `mdl` field plus 1. What this means is that you can use MODE SENSE (`msen`) to read a mode page, then use MODE SELECT (`msel`) to alter the mode page parameter(s) that you want to change, without worrying about the other differences between a MODE SENSE buffer and a

MODE SELECT buffer.

SCSI COMMAND TABLE

The SCSI command table file `/usr/share/misc/scsicmdtab` tells **scsicmd** how to initialize and display SCSI data. The table describes each SCSI command that **scsicmd** knows about and tells how to interpret the buffers that are associated with it. The table is written using a simple descriptive programming language.

The language is oriented around keywords. Keywords are fixed strings consisting of upper-case letters and underscores. Keywords like `COMMAND` begin every description of every unit. Some keywords are attribute or type keywords and appear later in a unit.

Keywords are sometimes followed by numeric parameters. Numbers have C syntax: hexadecimal numbers are prefixed with '`0x`', octal numbers are prefixed with '`0`', and other numbers are treated as decimal numbers. Some numeric parameters may be ranges of numbers, consisting of a low value, a hyphen '`-`', and a high value. White space (spaces or tabs) may appear in ranges.

Some keywords may be followed by strings. Strings are any non-keyword, non-numeric text that runs to a delimiter character, where delimiters are colons '`:`', parentheses '`()`', line breaks or comments. Delimiters are just 'syntactic sugar'; all delimiters are equivalent, and sequences of delimiters and white space are treated as a single delimiter. Delimiters only differ from white space in their ability to break strings.

Comments start with a hash-mark '#' character and run to the end of a line.

Each SCSI command is defined by a `COMMAND` description followed by `BUFFER` descriptions. The `BUFFER` descriptions are broken down into `BYTE` and (if necessary) `BIT` descriptions. Following the SCSI standard, bytes have big-endian order while bits have little-endian order - the most significant bit is the highest numbered bit in a field, while the most significant byte is the lowest numbered byte in a field. When bit fields cross byte boundaries, they wrap at the 0 bit. Here is the syntax:

`COMMAND`: *`description`* (*`abbreviation`*) [: *`command-type`*]

This directive begins a unit that describes a command. The *`description`* and *`abbreviation`* are the text that gets printed with **`-c`** `all`; the abbreviation is used to select the command with **`-c`** *`abbreviation`*. The *`command-type`* is optional and tells **scsicmd** what sort of operation to perform:

`CMD_READ`	Read a fixed-length buffer (or no data at all). This operation is the default.
`CMD_WRITE`	Write a buffer.
`CMD_READ_VAR`	Read a variable length buffer. Some SCSI commands like `INQUIRY` have variable length buffers. **Scsicmd** will limit the number of bytes of buffer that it prints on output to the value of `addlen` plus the offset of the byte following the `addlen` field.
`CMD_READ_DATA`	Read unstructured data and write it to standard output. **Scsicmd** can execute SCSI READ commands to dump data. The amount of data dumped is controlled by the '`tl`' or transfer length field in the CDB.
`CMD_SELECT`	Tell **scsicmd** to share buffers with the MODE SENSE command. **Scsicmd** uses a few tricks to make it possible to use data from a MODE SENSE command as input to a MODE SELECT command.
`CMD_SENSE`	Tell **scsicmd** to share buffers with the MODE SELECT command.
`CMD_RS`	Use a special `ioctl`(2) to recover REQUEST SENSE data from the last command that failed on this logical unit.

`CDB`: *description* (*abbreviation*) [: *size*]

The `CDB` directive is exactly the same as the `BUFFER` directive, but the explicit name makes that buffer's function more obvious. The first buffer of a command is always used for the CDB; subsequent buffers are controlled by the command type.

`BUFFER` [+ *offset*]: *description* (*abbreviation*) [: *size*]

There is at least one and possibly two pieces of memory associated with each SCSI command. These pieces of memory are broken down into 'buffers', which are descriptions of the bytes and bits contained at a particular offset in a chunk of memory.

The obligatory piece of memory is the CDB or command descriptor block. The CDB is always described by exactly one buffer, and that buffer must be listed first.

If a command reads or writes memory, then the second and subsequent `BUFFER` directives tell how to break that memory down. Buffers are assumed to start at offset zero in the memory chunk, but the offset may be provided explicitly. The size of the memory chunk is the explicit size of the first buffer, or if no size is given, 256 bytes. Non-described bits are assumed to be 0 when writing, and they can't be printed on reading. More than one buffer may be provided for decomposing a given chunk of memory at the same offset; this has the effect of providing alternative views of the same memory.

`BYTE[S]` *range* [: *description* (*abbreviation*) [: *format*] [: *value*]]

This directive describes a byte or byte range in a buffer. The mandatory argument may be a single number or a range, describing the offset of the byte or bytes within the buffer. If a description for a byte or byte range is provided, **scsicmd** assumes that the description covers all of the bits in the entire range of bytes.

A *format* key may be provided to cause the output for the field to be processed specially:

`FORMAT_DEFAULT`	Print the value of the field as a number (decimal or hexadecimal).
`FORMAT_STRING`	Print the value of the field as a string. Trailing blanks are stripped.
`FORMAT_CODE`	Print the value of the field as a special field-dependent code. **Scsicmd** uses this format key as a clue to check command dependent and field name dependent routines to print certain values in a special way. Currently three fields in INQUIRY and two fields in REQUEST SENSE are translated this way.

(The directives `BYTE` and `BYTES` are treated exactly the same; use the one that looks best.)

If there is no description, then **scsicmd** assumes that subsequent `BIT` directives will cover bits in the given byte.

`BIT[S]` *range*: *description* (*abbreviation*) [: *format*] [: *value*]

The `BIT` or `BITS` directive is just like the `BYTE` directive, but it covers bits or ranges of bits in a given byte. One or more `BIT` directives must be preceded by a `BYTE` directive giving the offset of the byte that contains the bits.

`BITLENGTH` *number*: *description* (*abbreviation*) [: *format*] [: *value*]

This directive is much like the `BIT` directive, but it is used to describe bit fields that cross byte boundaries. The SCSI standard defines such bitfields in a very restrictive way. The bitfield fills the low bits of the first byte, then fills whole subsequent bytes. For example, a field of 'BITLENGTH 21' will occupy the low 5 bits (21 mod 8) of the given byte, plus the next 2 bytes (21 div 8).

ENVIRONMENT

SCSI The default device filename. It may be overridden with **-f**.

FILES

/usr/share/misc/scsicmdtab
Default command table file.

EXAMPLES

How to label a SCSI disk using **scsicmd** directly:

```
# scsicmd -f /dev/rsd0c -l > /etc/label.sd0
Scanning for logical cylinder sizes...
Zone 1: 1626 cylinders, 427 sectors per cylinder
Total number of logical cylinders found in scan (1626)
differs from reported drive parameters (1629+3);
assuming 1626+6 cylinders for disk label.
# disksetup -R sd0 /etc/label.sd0 \
    /usr/bootstraps/ahaboot /usr/bootstraps/bootaha
```

How to format a disk:

```
# scsicmd -f /dev/rsd0c -c fu
```

How to list all the supported SCSI commands:

```
# scsicmd -c all
 ...
```

How to list all the supported MODE SENSE pages:

```
# scsicmd -c msen -v all
 ...
```

How to read the disk format device page:

```
# scsicmd -f /dev/rsd0c -c msen -P pcode=3 -p dfdp
parameters saveable (ps):                                 1
mode page code (mpcode):                                  3
mode page length (mpl):                                  22
tracks per zone (tpz):                                    8
alternate sectors per zone (aspz):                        5
alternate tracks per zone (atpz):                         0
alternate tracks per logical unit (atplu):               24
sectors per track (spt):                                 54
```

```
        data bytes per physical sector (dbpps):            512
        interleave (il):                                     1
        track skew factor (tsf):                             0
        cylinder skew factor (csf):                         16
        soft sectoring enable control (ssec):                0
        hard sectoring enable control (hsec):                1
        removable media (rmb):                               0
        surfaces instead of cylinders (surf):                0
```

How to turn automatic compression off on tape drives that support it:

```
# scsicmd -f /dev/nrst0 -c msen -P pcode=0x0f -p \
       none -c msel -P dce=0 -p none
```

SEE ALSO

sd(4), sg(4), st(4), disksetup(8)

STANDARDS

American National Standard X3.131-1986, Small Computer System Interface (SCSI-1)

American National Standard X3.131-1991 (SCSI-2)

HISTORY

Inspired by the **scsiinfo** program by Van Jacobson of Lawrence Berkeley Laboratories.

BUGS

If you want to use the diagnostic mode, you generally must start with a copy of the SCSI standard. The meanings of the various fields are described only in that document.

Tape devices in diagnostic mode don't always do what you expect. You must have a tape loaded when you run **scsicmd**, and the driver can execute a couple of MODE SENSE and/or MODE SELECT commands before turning over control to **scsicmd**.

Command and parameter abbreviations are ugly. The rules are inconsistent because of the need to reduce ambiguity. It was felt that acronyms were less awkward than using minimal string abbreviations of multi-word field names.

There is no way to access host adapter commands, hence it isn't possible to alter the DMA burst rate and other host adapter parameters.

The driver permits only a certain small subset of all the SCSI commands.

String values in **-P** can't contain commas.

-p *none* is clumsy.

NAME

ttys – terminal initialization information

DESCRIPTION

The file **ttys** contains information that is used by various routines to initialize and control the use of terminal special files. There is one line in the **ttys** file per special device file. Fields are separated by tabs and/or spaces. Fields composed of more than one word should be enclosed in double quotes (""). Blank lines and comments may appear anywhere in the file; comments are delimited by hash marks ("#") and new lines.

The first field is always the name of the terminal special file as it is found in `/dev`. The following fields are normally taken as tags to entries in the `/etc/ttys.conf.local` and `/etc/ttys.conf files.` (See `ttys.conf`(5)) If the field is of the form *name*=*value* then it is the same as specifying the name of a tag in the `ttys.conf` files which contain just *:name=value:*. By convention there is a field for the type of line (e.g., "com"), the type of modem (e.g., "Generic144"), and one or more fields describing the status of the line. Some of the standard fields describing status are:

bidir	This line is managed by `gettyd`(8) and is available for both incoming calls and outgoing requests.
in	This line is managed by `gettyd`(8) and is available for incoming calls.
init	This line is managed by `init`(8) and available for incoming calls. (Identical to "on")
network	This line is accessed via the network (i.e, ptys).
off	This line is not available for use.
on	This line is managed by `init`(8) and available for incoming calls. (Identical to "init")
out	This line is managed by `gettyd`(8) and is available for outgoing requests.
secure	Allow root logins on this line.

For backwards compatibility, if the second field starts with a '/' it is presumed to be the path name to the command to execute, typically `getty`(8). This is the same as specifying `command=`*command-path*.

The third field, in backwards compatibility mode, is taken as the type of terminal usually connected to the line, normally the one found in the `termcap`(5) data base file. This is the same as specifying `term=`*`terminal-type`*.

The string "auth=" may be followed by an authentication type to use for this line. (See `login.conf`(5) for additional information.)

EXAMPLES

```
# root login on ibm pc console, using init
console "/usr/libexec/getty pccons"     ibmpc3  on secure
# Mike's terminal: hp2621
ttyh0   std.9600 term=hp2621-nl on     # 457 Evans
# John's terminal: vt100
ttyh1   std.9600 term=vt100 on         # 459 Evans
ttyA0   digi Generic288 out            # dialout only line
ttyA1   digi Generic288 in             # dialin only line
ttyA2   digi Generic288 bidir          # bi-directional line
# Network pseudo ttys
ttyp0   network
```

FILES

/etc/ttys
/etc/ttys.conf
/etc/ttys.conf.local

SEE ALSO

`getttyent`(3), `ttyslot`(3), `dialer.conf`(5), `gettytab`(5), `login.conf`(5), `termcap`(5), `getty`(8), `gettyd`(8), `init`(8), `lo-gin`(8)

HISTORY

A **`ttys`** file appeared in Version 6 AT&T UNIX.

NAME

ttys.conf – terminal line configuration file

SYNOPSIS

/etc/ttys.conf
/etc/ttys.conf.local

DESCRIPTION

The `/etc/ttys.conf` file contains entries referenced by the `/etc/ttys` file as well user/group permissions/restrictions and destination information for outgoing calls. The over all format of the file is defined in the `cgetent`(3) manual page. Most entries may have more than a single value. Values are separated by the | (pipe) symbol. For example:

```
:entry=value1|value2|value3:
```

To specify a | use \|.

Entries which define restrictions or permissions for a given user or group are tagged by `user-`*name* or `group-`*name*.

The following are the standard entries. There is no limit on the names for new entries.

allow
: A list of the special sequences allowed by this user. If not defined, all special sequences not listed in the `restrict` entry are allowed. The special sequences are defined by the `map` entry for the chosen line. An example entry might be:

```
:allow=<DIALTONE>|<PAUSE>:
```

arguments
: A list of arguments which should be sent to the scripts defined by the `condition`, `dialer`, `hangup`, and `watcher` entries. The arguments are names of entries for which the values should be passed. For example:

```
:arguments=dte-speeds|term|line:
```

might cause the following arguments to be sent to a script

`-dte-speeds 9600 -term unknown -line tty01`

auth
: The type of authentication for this line. This type is used as `auth-`*`type`* in the `/etc/login.conf` file.

banner
: Banner to be displayed by the `login`(8) program.

calltype
: Types of calls allowed on this line, or by this user or group. The special type `DIRECT` implies the request is for the raw line itself with no dialing or setup. Other types of calls are defined by the `/etc/dialer.rules` file. By convention these are: `LOCAL`, `NATIONAL`, `INTL`, `ASSISTED-NATIONAL`, and `ASSISTED-INTL`.

command
: The name of the command to execute when carrier is enabled on the line. By default this is `/usr/bin/login`.

compression
: Types of automatic data compression available from the modem. While this value may be used to help select a line, the default dialing scripts provided with the system do not attempt to use this information.

condition
: Script called to condition a modem to accept incoming calls. The script should leave the modem in such a state that it will automatically answer incoming calls. The synopsis of the script is:

`condition` [**`-f`** *`file`*] [arguments]

The *`file`* argument is the file associated with the line, typically `/dev/fd/`*`xx`*. The arguments are the arguments specified by the `arguments` entry.

cost
: The relative cost of the line. If multiple lines match the request, and a speed was requested, the line with the lowest cost is used. By default the cost of a line is the maximum value in the `speed` entry for the line. If the request does not specify a speed, the lowest cost line with the highest available speed is used.

dce-speeds
: The DCE speeds available for this modem. The DCE speed is

the speed the modem communicates with the serial line.

destination
: The name of a single entry in the database to merge with the request. This entry only has use in a request and has no effects on line definitions. A destination typically includes the phone number or numbers as well as the optimum dialing speeds.

dialer
: Script to dial a number on a modem. The synopsis of the script is:

  dialer [**-f** *file*] [arguments]

  The *file* argument is the file associated with the line, typically `/dev/fd/`*xx*. The arguments are the union of the arguments specified by the `arguments` entry and entry which matched between the request and the entry for the line.

dialin
: This line is marked to accept incoming calls. This is used for both hardwired and modem connected lines.

dialout
: This line is marked to allow outgoing calls. See `gettyd`(8) for more information.

dte-speeds
: The DTE speeds available for this line (or the dte speeds desired). The DTE speed is the speed at which the serial line communicates with the modem. When both `dte-speeds` and `dce-speeds` are specified, the actual set of DTE speeds is the intersection of `dte-speeds` and `dce-speeds.`

error-correction
: The types of error correction available from the modem. While this value may be used to help select a line, the default dialing scripts provided with the system do not attempt to use this information.

flow
: The types of flow control available from the modem. While this value may be used to help select a line, the default dialing scripts provided with the system do not attempt to use this information. Use the `flowcontrol` field to specify a specific type of flow control.

flowcontrol
: Without a value, enable hardware flow control. If a value is

specified it must be one of:

hardware
: Enable hardware flow control for this line. (The same as not specifying a value). May also be abbreviated as `hw`.

none
: No flow control is enabled.

software
: Enable software flow control for this line. May also be abbreviated as `sw`.

groups
: A list of groups allowed to access this line.

hangup
: A script similar to `condition`, however, it simply assures the phone is hung up and the line is ready for another dialout session. It is only called on dialout lines. The synopsis for `hangup` is the same as that of the `condition` script.

hcondition
hdial
hhangup
hinit
hquiet
hreset
: See the `hayes`(8) manual page for details on these fields.

issue
: The name of a file to be displayed after the `login`(8) program prints the login banner (specified by the `banner` field.) No errors are generated if the file pointed to does not exist.

line
: The list of devices to select from for this connection. It may only be specified as part of a request (or in a destination definition).

manager
: The name of the program which manages this line. This does not cause the named program to be run. It is used by a program when it is searching for entries which it should control. If this has no value, or is specified to be `none`, then the line is not managed. Enabled lines normally have the value of `gettyd`, or `init` for compatability with older systems.

map
: A mapping of special sequences to strings used by the modem. Each entry is of the form: <*sequence*>*string*. For example, a hayes compatible modem might have the entry:

```
:map=<FLASH>!|<TONE>T|<PULSE>P|<DIALTONE>W
```

`|<REVERSE>~|<SILENCE>@|<PAUSE>,:`

modemcontrol
: Enable modem control for this line.

modemtype
: A verbose description of the modem. In general its presence indicates this entry is describing a modem.

modulation
: The types of modulation (i.e., `bell103`, `v.21`, `v.34`...) available from the modem. While this value may be used to help select a line, the default dialing scripts provided with the system do not attempt to use this information.

number
: The list of phone numbers to dial. Phone number may be of the form `@`*name* in which case *name* is looked up in `/etc/phones`. Variable expansion, represented by `{`*variable*`}`, is performed on the final number. Variables are defined in the `/etc/dialer.rules` file. (See `dialer.rules`(5).) Special sequences are represented as `<SPECIAL>`. (See the `map` entry and `dialer.rules`(5).)

porttype
: A verbose description of the type of serial port. In general its presence indicates this entry is describing a serial port.

ppp
: Program called by `login`(8) when PPP LCP negotiation is detected. This defaults to the `ppp`(8) program.

pppname
: Name passed to the `ppp`(8) program by `login`(8) when PPP LCP negotiation is detected. This defaults to `ppp_direct`.

required
: A list of fields that must be present in the request in order to match this entry. A common example might be:

```
:required=number:
```

restrict
: A list of special sequences not allowed.

secure
: This line is marked as secure. A secure line allows direct root logins.

speaker
: Describes the various settings the speaker may be set to. While this value may be used to help select a line, the default dialing scripts provided with the system do not attempt to use this information.

speed
: The various modulation speeds available with the connected modem. The modulation speed is the speed at which the modems communicate. The is different from `dte-speeds` and `dce-speeds`.

stty-modes
: Additional modes to set on the line. See `stty`(1) for a list of modes.

term
: The default terminal type for this line.

users
: A list of users allowed to access this line.

uucplocking
: This boolean option indicates of uucp style locking needs to be done on this line.

verbose
: Request the dialer to be verbose with what it is doing.

volume
: Describes the volume settings that may be set for the modem. While this value may be used to help select a line, the default dialing scripts provided with the system do not attempt to use this information.

watcher
: A script similar to `condition`, however, it does not return unless there was an error in initializing the modem. Normally it answers the phone and directly starts the getty program specified. The synopsis for `watcher` is the same as that of the `condition` script.

window
: The name of the window manager to use on this line. This is archaic and is only present for backwards compatibility. It is only used by `init`(8).

SEE ALSO

`dialer.rules`(5), `ttys`(5), `gettyd`(8), `gettystat`(8), `hayes-condition`(8), `init`(8), `login`(8)

INDEX