Chapter 3
Installation

New OpenBSD users often complain that the installation process is hard and, consequently, give up before they have even gotten a system up and running. For most of these people, simply reading the documentation for the installation process would have solved most of their problems. Unlike many other systems, OpenBSD isn't designed to be configured with the click of a button—there is no graphical installation process.[1] This chapter will walk through some of the more important parts of the installation process and give information about some of the options. It will not try to guide the user through a normal installation, as that process is well documented on the OpenBSD Web site[2] and included with the official CDs.

3.1 Supported Hardware

OpenBSD supports a great deal of hardware, but it's more centered on being a server operating system, so support for fancy desktop hardware sometimes lags behind. People seeking support for the newest hardware should look more closely at FreeBSD or Linux as a way to satisfy their needs. Because the list of supported hardware is a moving target, it has not been included in this text. Consult the list at the OpenBSD Web site, which is kept current.[3] Careful consideration must be taken before installing OpenBSD on a system. While Linux will support almost any given piece of hardware, the same cannot be said of an OpenBSD system. Review the Web site listing to make sure the system will be supported before proceeding with an installation.

[1] Although a graphical installation process is being developed by some folks outside of the OpenBSD project, it is not a part of the main installation process. More information about this project is available at **http://www.gobie.net**.
[2] The installation guide is available at **http://www.openbsd.org/faq/faq4.html**.
[3] Check **http://www.openbsd.org/plat.html** for more information.

3.2 System Preparation

One of the most important steps in the installation process is getting to know the hardware. Make sure the documentation for the main parts of the system is at hand, or at least readily accessable from the Internet. Having this information will make the installation go much more smoothly if any problems occur.

Before an installation is started, it is good practice to review the hardware used in the system and to make notes to be used during the installation. The main components include the following:

- CPU and system architecture
- Network interface card or cards
- Disk devices and controllers
- Video hardware
- Sound hardware

The hardware in the system can be referenced against the supported hardware list for each system type and the drivers that OpenBSD will use for each device. Note that limited models of some platforms, like the Digital Alpha platform, are supported due to the diverse hardware used in their product lines. The Intel i386 platform is very well supported, however.

If the machine already has an operating system loaded, it is important to perform a full backup before installing OpenBSD. When installing the new OS, you have two options: replacing the entire hard drive with OpenBSD or partitioning the hard drive such that both operating systems can coexist. When you have multiple operating systems on the same computer, you can choose which will run when the machine is started, a practice commonly called "dual booting." You may find it easiest to back up your data, repartition the hard drive to have dedicated space for both operating systems, reinstall the original OS, reinstall your data from backups, and only then install OpenBSD. For pointers see **ftp://ftp.openbsd.org/pub/OpenBSD/3.4/i386/INSTALL.linux**. Some free boot loaders for dual booting an OpenBSD i386 system are provided in the **ftp://ftp.openbsd.org/pub/OpenBSD/3.4/tools** directory; several commercial boot managers also work with a dual-boot scenario for your OpenBSD system.

Another thing to be considered before the installation begins is what the machine will be used for. Not only can this affect which programs are installed, but it can also change how the disk should be partitioned. If the machine is to be used as a workstation, more space should be allocated for applications; if it will be used as a mail or Web server, space for the respective data files and logs should be reserved.

As when working with any new system, be prepared to reinstall a few times to get the hang of the installation and to find the best possible disk layout. Because Linux, Solaris, and OpenBSD all spread themselves across the hard disk in different ways, a disk partitioning scheme that may have worked for one OS may not work for another.

3.3 Getting the Files for Installation

The main ways to install OpenBSD are either by using the installation CD or by doing an FTP installation through the Internet or from a local mirror, if one exists. Many people who are new to OpenBSD search in vain for a downloadable copy of the OpenBSD ISO image to create a CD for themselves. Unlike with most other free operating systems, OpenBSD ISO images are not available online because the official CD image itself is trademarked. Although this practice may seem restrictive, sales represent one of the main ways to fund the project. Many people even buy the CD every six months[4] even if it won't be used, just to support the project.

Until recently, the fact that OpenBSD used strong cryptography meant that serious export restrictions applied to it. Now that the United States has modified these restrictions, many of these concerns for U.S. citizens do not exist.

3.4 Selecting Boot Media

Some architectures, such as i386, give several floppy disk choices. These accommodate various configurations and device options. Because of the small capacity of a floppy disk, only a smaller kernel with a limited amount of drivers is available for installation purposes.

The most commonly used installation image is **floppy34.fs**. It supports most IDE i386 systems and most generic laptop systems. The image **floppyB34.fs** contains several SCSI and RAID drivers enabled, and the image **floppyC34.fs** supports additional PCMCIA and cardbus devices.

For the Alpha platform, the floppy images **floppy34.fs** and **floppyB34.fs** differ in their supported hardware. Because of the diversity of Alpha hardware, these images are mutually exclusive and specific to various hardware lines. Check the documentation to determine which one to use.

As always, boot the CD if possible. It is a much easier approach to installation, with most drivers and configurations being rolled into one system. Some architectures don't

[4] OpenBSD releases a new version about every six months; OpenBSD 3.3 was released May 1, 2003, and 3.4 was released November 1, 2003.

have such installation choices, such as HP300 and Mac68k. Here an alternate operating system or netbooting is used for the installation purposes. Mac68k, for example, installs from an existing MacOS installation.

3.5 Booting

For most systems, booting and installing are as easy as putting the OpenBSD CD[5] in the system and away you go. On some architectures, however, this is not the case. In addition, many systems don't have a CD drive, or even a floppy disk drive for that matter. If these systems have newer network cards, network boots can be done. If they lack all of these options, more creative methods need to be used.

3.5.1 The Boot Configuration

Configuration of the boot process is done in the file **/etc/boot.conf**. Various options can be set here, including the ability to have the device boot via the command *set device root_device* (e.g., *set dev hd1a*). Notice that IDE devices are hd* devices, not wd* devices as they are listed in the booted kernel.

The *boot* keyword tells the system to boot immediately. A timeout can be set using the *set timeout* command—for example, *set timeout 30* to wait for 30 seconds. The kernel file can be specified here as well. Use *boot/bsd.new* to boot a new kernel named **/bsd.new**, or even *boot hd1a:/bsd* to boot the kernel **/bsd** from the device **hd1a**.

Serial console options can also be set here using the *stty* command. To set the serial port 0 to 19,200 baud, use the command *stty com0 19200*. Login prompts will still be displayed on screens 1–7 as normal.

To boot into single user mode—for example, to edit a lost root password or a broken root shell—use *boot -s*. Note that you will have to remount the disks with read-write privileges to access and edit files. No root password is needed to perform this option. The command *boot -a* can be used to specify an alternate root device (the system defaults to **wd0a**).

If you have a kernel problem to debug, you can issue the command *boot -c* to boot while configuring the kernel. This option is very time intensive, but it does allow you to enable or disable devices and options as needed, one by one. Lastly, use the command *boot -d* to drop to the ddb prompt immediately, which is useful for kernel testing.

[5]This section presumes that a CD installation will use a bootable OpenBSD CD that was purchased from the OpenBSD Web site.

3.5.2 Creating a Serial Console

For systems that run "headless," or without a monitor and keyboard, the next best thing to use is a serial console. Connecting two computers via a NULL serial cable, a serial console gives almost all of the same benefits available with a monitor and keyboard attached to the system but without extra equipment. Some form of non-networked console is needed for operations such as installation, firmware or BIOS-level operations, and kernel debugging. Serial console setup differs between the major hardware platforms.

On a typical i386-class PC, a serial console can be used to control the system almost entirely. Most desktop PCs and servers will begin using the serial device once the OpenBSD boot loader software begins, configured through the file **/etc/boot.conf** and described in the previous section. While most PC systems forbid BIOS access via the serial console, some more expensive server hardware does allow for this possibility.

Sun SPARC and UltraSPARC systems are designed to operate using serial consoles in the absence of a monitor or keyboard being attached. Simply attach a serial cable with a NULL modem adapter, and it should work without issue (as long as a keyboard isn't plugged in, the PROM automatically switches to serial mode). To create a serial console in the presence of a monitor, you need to edit two firmware values and save them:

```
ok  setenv input-device ttya
ok  setenv output-device ttya
ok  reset
```

By configuring **/dev/ttyc0** in the file **/etc/ttys**, the keyboard and monitor can still be used for normal login operations, just not console access.

MacPPC hardware, which uses OpenFirmware, is quite similar in the setup of a serial console to the SPARC family. Once at the firmware (arrived at by booting the system with the key sequence of command-option-O-F), edit the values for the input and output devices:

```
ok  setenv output-device scca
ok  setenv input-device scca
ok  reset-all
```

This will activate the serial console at 57,600 bps with 8N1 settings. Note that some older MacPPC machines default to using the serial console for OpenFirmware access. Newer models, required for using OpenBSD on the MacPPC platform, typically do not do so.

For Mac68k machines, the MacOS-based boot program can be used to select the serial console. The modem port is **tty00** and the printer serial port is **tty01**.

For all of the above systems, you will have to activate the login processes on the serial terminal as well. This is controlled through the file **/etc/ttys**. By default, the serial consoles (known as devices **tty00** and **tty01**) are off, but they can be enabled. Simply set the terminal type (in this example, *vt100*) and enable it (using the *on* keyword) and allow for root logins (with the keyword *secure*):

```
tty00   "/usr/libexec/getty std.9600"   vt100   on secure
```

For MacPPC machines, the setting *std.57600* should be used rather than the 9600 baud setting.

From an OpenBSD machine, you can access this serial console on another system by using a serial terminal program such as *minicom*, available in ports, or the built-in command *cu*:

```
#  cu -l cua00 -s 9600
```

This would launch a serial console to the device **cua00** (the first serial port) at 9600 baud. Serial consoles are also used to access embedded devices and networking equipment.

3.5.3 Platform-Specific Information

i386 Hardware

Most i386 systems are easy to work with using either a CD player or floppy drive and a network card to perform the installation. The BIOS simply needs to be told to use the CD as the boot source. On some very old systems, the CD drive isn't bootable and the floppy installation path must be used. If the system has an adequate network card, the installation can be done using just the network card by doing a network installation with the help of a network boot server. If none of the options will work, a last-ditch method is to simply remove the system disk, install it in a different computer, and put the disk back in. As long as the architecture is the same, OpenBSD is very forgiving. The same software is installed no matter what hardware is on the system.

SPARC and SPARC64

For the most part, installation of OpenBSD on non-Intel i386 platforms is very much the same as installation on Intel platforms. The major differences come in booting and setting the machine to boot the OpenBSD kernel. In some cases the booting firmware needs only a few options set to boot correctly.

On Sun SPARC hardware, booting from a floppy disk is the most straightforward method to install the OpenBSD system. You can boot from the floppy disk by getting to the PROM, typically by pressing Stop-A on the keyboard. Once there, you can type the following to boot from the floppy disk:

```
ok  boot floppy
```

(The "ok" is the PROM prompt.) The installation procedure at this point closely resembles the i386 installation procedure. Once that completes, you will need to modify your PROM settings to boot from the hard disk and to load the file **bsd**, rather than the **vmunix** file that is expected by the Sun hardware. To do this, at the PROM type the following:

```
ok  setenv boot-file bsd
ok  setenv boot-from sd@0,0       for SCSI ID 0
 or
ok  setenv boot-from disk         for the default disk
                                   at SCSI ID 3
 then
ok  nvramrc                       to make these changes
                                   permanent
```

If additional information is needed about the devices on the SCSI chain, the PROM can be used to query the devices and show their names:

```
ok  probe-scsi-all
ok  probe-scsi                  an optional command
```

Systems that lack a CD or floppy drive can also easily be booted off a network server for their installation:

```
ok  boot net
```

This requires knowledge of the boot server's Ethernet address as well as a properly configured boot server that can serve the appropriate files.

For additional information, read the file **INSTALL.sparc** in the distribution directory.

3.5.4 Boot Example

```
boot>
booting fd0a:/bsd: 4162256+710560=0x4a5bf8
entry point at 0x100120

Copyright (c) 1982, 1986, 1989, 1991, 1993
The Regents of the University of California.  All rights reserved.
Copyright (c) 1995-2003 OpenBSD. All rights reserved.
http://www.OpenBSD.org

OpenBSD 3.4 (RAMDISK_CD) #127: Wed Sep 17 03:49:42 MDT 2003
deraadt@i386.openbsd.org:/usr/src/sys/arch/i386/compile/RAMDISK_CD
cpu0: Intel Pentium III ("GenuineIntel" 686-class, 512KB L2 cache)
    499 MHz
cpu0: FPU,V86,DE,PSE,TSC,MSR,PAE,MCE,CX8,SEP,MTRR,PGE,MCA,CMOV,PAT,
    PSE36,MMX,FXSR,SIMD
real mem  = 268009472 (261728K)
...
...

....
rootdev=0x1100 rrootdev=0x2f00 rawdev=0x2f02
de0: enabling Full Duplex 100baseTX port
erase ^?, werase ^W, kill ^U, intr ^C, status ^T
(I)nstall, (U)pgrade or (S)hell?  i

Welcome to the OpenBSD/i386 3.4 install program.

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

Terminal type? [vt220]  enter
```

```
Do you wish to select a keyboard encoding table? [n]  enter

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

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

Proceed with install? [n]  y
Cool!  Let's get to it...
```

3.6 Filesystem Partitioning

The layout of a disk is highly dependent on the nature of the system being created. Four major system types are described below: a personal system (e.g., a laptop), a shell server, various server types, and a firewall. These constitute the major points on the spectrum of security and convenience. The *disklabel* program, which is used to set up the partitions in the installation program, is described in Chapter 14.

Filesystem sizes are relatively easy to figure out, but some thought must be given to them. The base system, with a separate **/usr** filesystem, takes up approximately 150MB to 250MB of space plus some room in which to move around. The **/usr** filesystem takes up 1GB to 1.5GB of space with all of the base system and a small set of common ports and packages installed. If the system source is installed, such as in **/usr/src**, and the ports tree is installed in **/usr/ports**, additional space is required. Letting the **/usr** filesystem be the remainder of a filesystem is probably the most convenient way to ensure that adequate space is allocated.

Many users also choose to separate their **/usr/local** and **/home** directories. This allows filesystems to be segmented for security and maintenance purposes.

3.6.1 A Private System

A personal system, such as a laptop, has dramatically different needs than a multiuser system. Because typically only one user is using the system, and it is often used as a primary machine, additional third-party packages are usually installed. For developers with source code installed, the size requirements can increase dramatically.

The largest changes will occur in the **/home** and **/usr** filesystems. At least 3GB of space for the **/usr** filesystem to hold the OpenBSD source tree, the ports tree, and dozens of distribution files for the packages is sufficient. A large **/home** filesystem is also important. Typically 2GB to 5GB of space is sufficient. Developers will also want to keep their **/home** directories executable for source code testing. Another solution is to place home directories on the **/usr** filesystem in a location like **/usr/home**. This will permit the remaining disk space on the drive to be allocated to the **/usr** filesystem without worrying about space needs.

3.6.2 A Multiuser System with Untrusted Users

World writable filesystems are the greatest concern for a system with many users, most of whom are untrusted. The need for world writable filesystems arises because of temporary files, such as editor recovery files or mail spools. They simply cannot be escaped, but their risk can be mitigated. World writable directories include temporary storage locations such as **/tmp** and **/var/tmp**, a user's own directory in **/home**, and the mail queue directory **/var/spool/mqueue**.

The simplest option for preventing an attack is to partition the disk according to this set of concerns and to specify mount options for those filesystems requiring greater security. With this in mind, a general-purpose server system with shell users would be partitioned as follows:

- **/usr** to hold executables
- **/tmp** to hold temporary files
- **/home** for the users' home directories
- **/var** for system volatile files
- **/** for everything else

The concerns here are space and security. Space concerns come from logfiles or temporary storage space filling the root filesystem and causing a crash.

For security issues, there are four mount options to consider:

- **nodev** No device files are allowed on the filesystem.
- **noexec** No application can be executed on this filesystem. This can be circumvented through a variety of techniques, however.
- **nosuid** No application can run with a privilege level different than the user's level.
- **rdonly** Mounts a device as a read-only device.

The option **rdonly** forces a filesystem to be mounted as read-only, preventing any write attempts to it. It can be used with base system filesystems (such as **/** and **/usr**) to prevent alteration of the system's executables. Note that the filesystem will have to be remounted with read-write privileges enabled to update the system. The root and **/usr** filesystems should not be mounted with the ***noexec*** or ***nosuid*** options as they *must* be able to execute applications and *setuid* binaries.

An additional consideration is the backup of the system. If the system is being backed up to tape, the relationship between the size of the tape and the filesystem needs to be considered. If, for example, a DDS2 tape is being used to back up the **/home** filesystem, the partition should not be larger than about 6g[6] so it can all fit onto one tape.

3.6.3 Server Partitioning

The size of the partitions on a server system greatly depends on the use of the system. If the server will be used for only one task, such as being a Web server, then space considerations are easy to resolve. If, however, the system will be a database server, a mail server, and a Web server, then obviously more must be considered.

For each service that the machine will be running, space will be needed. For a mail server, that means a large **/var/spool** filesystem. For an FTP server, space must be set aside for **/home/ftp**. The only server that doesn't really have disk requirements would be a DNS server, as it takes almost no space and doesn't log excessively. An important issue is the **/var** filesystem for a nameserver. Because the *named* process runs in a jail located in **/var/named**, it has to create a logging device to replace **/dev/log**. For a nameserver running on such a system, a separate partition for **/var/named** should be created and allowed to have a device for the system logger. In addition to the space for each application, the space that the log files will take up needs to be evaluated. In a Web server, for example, Apache will keep its logs in **/var/www/logs** by default. How much traffic will the Web server get that will be logged? Will Web log processing software be running on the system that will generate more information as well? In general, a lot of space should be allocated for the **/var** filesystem to allow for logging, even with log rotation in operation.

Because these systems will be servers, they will change a lot less than other types of systems. As a consequence, they won't need XF86 installed and may not even need the system source code or ports. Thus the **/usr** filesystem can be smaller and the **/home** filesystem can even be ignored given that there should be no users on the machine.

[6] The 6g estimate assumes 33 percent compression.

3.6.4 Firewall

Although setting up and running a firewall can be tricky, the partitioning for it is quite easy. The only space consideration arises for **/var** if *pflogd* is being used to log data. A firewall system also shouldn't need much of anything installed from the ports tree. In addition, filesystems should be mounted with *noexec, nosetuid*, and *nodev*. The **/usr** filesystem could even be mounted as read-only if there will be few changes.

3.6.5 Swap Space Allocation

Swap size calculation is something that is sometimes debated, but relies on a set of simple principles. While it is important to have enough memory to run all of your applications, in the event of a large-scale system fault, you'll want to have enough memory to handle a full dump of the physical RAM pages and any affected applications. This amount is typically twice the amount of physical RAM in the system. Thus, for a system with 128MB of physical RAM, a 256MB swap space should suffice for both running the system and allowing all of the system pages to be dumped to disk should the need arise.

3.6.6 Partitioning Example

The following is an example of partitioning for a firewall system.

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

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

# using MBR partition 3: type A6 off 63 (0x3f) size 26699967
    (0x19768bf)

Treating sectors 63-26700030 as the OpenBSD portion of the disk.
You can use the 'b' command to change this.

Initial label editor (enter '?' for help at any prompt)
> p
device: /dev/rwd0c
```

```
type: ESDI
disk: ESDI/IDE disk
label: WDC AC313600D
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
cylinders: 16383
total sectors: 26712000
free sectors: 0
rpm: 3600

16 partitions:
#        size   offset    fstype   [fsize bsize   cpg]
  a:   262017       63    4.2BSD     2048 16384   260
  b:   262080   262080      swap
  c: 26712000        0    unused        0     0
  d:   262080   524160    4.2BSD     2048 16384   260
  e:  2097648   786240    4.2BSD     2048 16384   328
  f: 23816142  2883888    4.2BSD     2048 16384   328
```

As can be seen above, OpenBSD was set up on this system before. We are going to delete all of the old partitions and create new ones.

```
> d a
> d b
> d d
> d e
> d f
> p
device: /dev/rwd0c
type: ESDI
disk: ESDI/IDE disk
label: WDC AC313600D
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
```

```
cylinders: 16383
total sectors: 26712000
free sectors: 26699967
rpm: 3600

16 partitions:
#        size   offset    fstype   [fsize bsize   cpg]
  c: 26712000        0    unused        0     0
> a a
stray interrupt
offset: [63]  enter
size: [26699967] 128M
Rounding to nearest cylinder: 262017
FS type: [4.2BSD]  enter
mount point: [none]  /
> a b
offset: [262080]  enter
size: [26437950] 128M
Rounding to nearest cylinder: 262080
FS type: [swap]  enter
> a d
offset: [524160]  enter
size: [26175870] 128M
Rounding to nearest cylinder: 262080
FS type: [4.2BSD]  enter
mount point: [none]  /tmp
> a e
offset: [786240]  enter
size: [25913790] 1024M
Rounding to nearest cylinder: 2097648
FS type: [4.2BSD]  enter
mount point: [none]  /var
> a f
offset: [2883888]  enter
size: [23816142]  enter
FS type: [4.2BSD]  enter
mount point: [none]  /usr
> p
```

```
device: /dev/rwd0c
type: ESDI
disk: ESDI/IDE disk
label: WDC AC313600D
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
cylinders: 16383
total sectors: 26712000
free sectors: 0
rpm: 3600

16 partitions:
#        size   offset    fstype   [fsize bsize   cpg]
  a:   262017       63    4.2BSD     2048 16384    16  # /
  b:   262080   262080      swap
  c: 26712000        0    unused        0     0
  d:   262080   524160    4.2BSD     2048 16384    16  # /tmp
  e:  2097648   786240    4.2BSD     2048 16384    16  # /var
  f: 23816142  2883888    4.2BSD     2048 16384    16  # /usr
>  w
>  p
device: /dev/rwd0c
type: ESDI
disk: ESDI/IDE disk
label: WDC AC313600D
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
cylinders: 16383
total sectors: 26712000
free sectors: 0
rpm: 3600

16 partitions:
#        size   offset    fstype   [fsize bsize   cpg]
```

```
  a:   262017       63    4.2BSD     2048 16384    16  # /
  b:   262080   262080      swap
  c: 26712000        0    unused        0     0
  d:   262080   524160    4.2BSD     2048 16384    16  # /tmp
  e:  2097648   786240    4.2BSD     2048 16384    16  # /var
  f: 23816142  2883888    4.2BSD     2048 16384    16  # /usr
> q
No label changes.
The root filesystem will be mounted on wd0a.
wd0b will be used for swap space.
Mount point for wd0d (size=131040k)? (or 'none' or 'done') [/tmp]
Mount point for wd0e (size=1048824k)? (or 'none' or 'done') [/var]
Mount point for wd0f (size=11908071k)? (or 'none' or 'done') [/usr]
Mount point for wd0d (size=131040k)? (or 'none' or 'done') [/tmp]
Mount point for wd0e (size=1048824k)? (or 'none' or 'done')
    [/var]  done
No more disks to initialize.


You have configured the following partitions and mount points:

wd0a /
wd0d /tmp
wd0e /var
wd0f /usr

The next step creates a filesystem on each partition, ERASING
existing data.
Are you really sure that you're ready to proceed? [n]  y
Warning: cylinder groups must have a multiple of 2 cylinders
Warning: 64 sector(s) in last cylinder unallocated
/dev/rwd0a: 262016 sectors in 260 cylinders of 16 tracks, 63 sectors
        127.9MB in 1 cyl groups (260 c/g, 127.97MB/g, 16384 i/g)
/dev/rwd0d: 262080 sectors in 260 cylinders of 16 tracks, 63 sectors
        128.0MB in 1 cyl groups (260 c/g, 127.97MB/g, 16384 i/g)
Warning: cylinder groups must have a multiple of 2 cylinders
/dev/rwd0e: 2097648 sectors in 2081 cylinders of 16 tracks,
    63 sectors
```

```
        1024.2MB in 7 cyl groups (328 c/g, 161.44MB/g, 20608 i/g)
Warning: cylinder groups must have a multiple of 2 cylinders
Warning: 884 sector(s) in last cylinder unallocated
/dev/rwd0f: 23816140 sectors in 23628 cylinders of 16 tracks,
    63 sectors
        11629.0MB in 73 cyl groups (328 c/g, 161.44MB/g, 20608 i/g)

/dev/wd0a on /mnt type ffs (rw, asynchronous, local,
    ctime=Sat Nov 15 14:16:10 2003)
/dev/wd0d on /mnt/tmp type ffs (rw, asynchronous, local, nodev,
    nosuid, ctime=Sat Nov 15 14:16:10 2003)
/dev/wd0f on /mnt/usr type ffs (rw, asynchronous, local, nodev,
    ctime=Sat Nov 15 14:16:11 2003)
/dev/wd0e on /mnt/var type ffs (rw, asynchronous, local, nodev,
    nosuid, ctime=Sat Nov 15 14:16:11 2003)
```

3.7 Network Configuration

During the installation process, there isn't much configuration to be done in terms of networking. The installation will set the hostname for the system and the domain name. Static or dynamic IP addresses are set for any interfaces that are seen with the base kernel. Some drivers may not be supported by the installation kernel and can be configured only after the system's normal kernel has booted.

3.7.1 Network Setup Example

The next step in the installation process is to configure the networking on the machine.

```
System hostname? (short form, e.g. 'foo')  server
Configure the network? [y]  enter
Available interfaces are: de0.
Which one do you wish to initialize? (or 'done') [de0]  enter
Symbolic (host) name for de0? [server]  enter
The default media for de0 is
        media: Ethernet autoselect (100baseTX full-duplex)
Do you want to change the default media? [n]  enter
```

The first option is to set the IP address for the interface using DHCP.

```
IP address for de0? (or 'dhcp')  dhcp
Issuing hostname-associated DHCP request for de0.
Internet Software Consortium DHCP Client 2.0pl5-OpenBSD
Listening on BPF/de0/00:40:05:a3:36:56
Sending on   BPF/de0/00:40:05:a3:36:56
Sending on   Socket/fallback/fallback-net
DHCPDISCOVER on de0 to 255.255.255.255 port 67 interval 1
DHCPOFFER from 10.0.10.10
DHCPREQUEST on de0 to 255.255.255.255 port 67
DHCPACK from 10.0.10.10
New Network Number: 10.0.0.0
New Broadcast Address: 10.0.255.255
bound to 10.0.0.201 -- renewal in 21600 seconds.
No more interfaces to initialize.
DNS domain name? (e.g. 'bar.com') [crimelabs.net]  enter
DNS nameserver? (IP address or 'none') [10.0.0.2]  enter
Use the nameserver now? [y]  enter
Default route? (IP address, 'dhcp' or 'none') [dhcp]  enter
```

The second option, using static addresses, is not much different.

```
IP address for de0? (or 'dhcp')  10.0.10.50
Netmask? [255.255.255.0]  255.255.0.0
No more interfaces to initialize.
DNS domain name? (e.g. 'bar.com') [my.domain]  crimelabs.net
DNS nameserver? (IP address or 'none') [none]  10.0.0.2
Use the nameserver now? [y]  enter
Default route? (IP address, 'dhcp' or 'none')  10.0.0.1
add net default: gateway 10.0.0.1
```

At this point you can invoke a shell and modify networking as needed. This can include setting the media options on the device or manually adjusting any DHCP leases you received.

```
Edit hosts with ed? [n]  enter
Do you want to do any manual network configuration? [n]  enter
Password for root account? (will not echo)  password
Password for root account? (again)  password
```

3.8 Base Software Set Installation

The OpenBSD sets are distributed as a collection of compressed tar files, usually ending with the extension **.tgz**. They should not be confused with the ports and packages system, which also uses this filename extension. These files are not added with the *pkg_add* commands (see Chapter 12), but instead are unpacked directly with *tar* and *gunzip*.

The installation program will prompt you for which sets to install and provide you with a way to select all or groups of sets. These include the base system, compilers, manual pages, and the X11 windowing system. They also include the kernel file, **bsd**, which is not compressed.

3.8.1 Types of Installations

The sets to install depend on the installation itself. For example, a NIDS sensor or a firewall may be better suited to installation without the X11 system, the games packages, and the compiler package. However, a workstation or a laptop would be fairly useless for most people without a windowing system.

Additional sets may be selected and installed after the initial round of sets is added. Thus, if you change your mind later, you can modify the installed sets by adding a set.

Sets may be added after the installation program is long gone and the system is running normally because they are simply compressed tar files. However, the right flags must be used to preserve file ownership and modes. The files should be unpacked in the root directory (**/**). The following example shows how to add the compiler package, located in **/tmp/comp34.tgz**, to a running system:

```
#  tar -C / -zxvpf /tmp/comp34.tgz
```

The files will be unpacked relative to the root of the filesystem and prepared for immediate use. This is not the official way to unpack installation sets, but it has been used by the authors on several occasions.

Filesets may be deleted as well, albeit with a bit more difficulty. This is because the files are not registered with any packaging system. However, as described in Chapter 15, the *tar* command can list the contents of the archive. The following commands will remove the contents of any of the sets:

```
#  cd /
#  rm -f ' tar -ztvf /tmp/comp34.tgz | awk '{print $9}'
```

This code will show only the filenames from the output of the contents argument to *tar* and remove the files. We do not want to remove directories, as they may be used by other packages. Although example shows the package installed above, the same technique should work for any of the base packages. Note that this is also a very unofficial and unsupported method of removing software, but it has been used by one of the authors on several occasions to alter installations.

3.8.2 Descriptions of the Installation Sets

The binary sets from an installation (or upgrade) are named to indicate what they hold and what they provide. The asterisk indicates the version independence of the name (typically the name includes the system version, such as 34 for OpenBSD 3.4). The sets are listed below.

- **base*.tgz** The base OpenBSD system is installed from this set, which includes core system functionality in userland. **You must install this set**.
- **etc*.tgz** This set contains the **/etc** files for configuration of the system. **You are required to install this set**.
- **comp*.tgz** The compilers (C, C++, F77) are installed from this set. This set is not required, but it is recommended for a typical server or system use.
- **man*.tgz** The manual pages are installed in this set, documenting the system. This set is not required, but is recommended.
- **misc*.tgz** This set contains miscellaneous information, such as the BSD 4.4 manuals (SMM, USD, PSD).
- **game*.tgz** This set contains the system games, typically console games.
- **xbase*.tgz** The base of the X11 system is installed from this set.
- **xfont*.tgz** This set contains fonts for the X11 system; it is required for the X11 system.
- **xserv*.tgz** The X11 server is required for the X11 system.
- **xshare*.tgz** The manual pages and locale settings in this set include files for the X11 system.
- **bsd** The BSD kernel is **required for a working system**.
- **bsd.rd** This ramdisk version of the kernel is suitable for booting from a CD-Rom device. Download it if you plan to build your own bootable CD filesystems for installation. This set is not suitable for a running kernel.

3.8.3 Installation Example

The example installation will be done from a local FTP mirror server. We will install all packages except the games package.

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

Sets can be located on a (m)ounted filesystem; a (c)drom, (d)isk
or (t)ape device; or a (f)tp, (n)fs or (h)ttp server.
Where are the install sets? (or 'done')  f
HTTP/FTP proxy URL? (e.g. 'http://proxy:8080', or 'none')
    [none]  enter
Display the list of known ftp servers? [y]  enter
Getting the list from 129.128.5.191 (ftp.openbsd.org)...done.
  1  ftp.openbsd.org/pub/OpenBSD                Alberta, Canada
  2  mirror.aarnet.edu.au/pub/OpenBSD           Canberra, Australia
  3  ftp.it.net.au/mirrors/OpenBSD              Perth, Australia
....
....

....
 74  mirror.cs.wisc.edu/pub/mirrors/OpenBSD     Madison, WI, USA
Server? (IP address, hostname, list#, 'done' or '?')  10.0.0.2
Does the server support passive mode ftp? [y]  enter
Server directory? [pub/OpenBSD/3.4/i386]  enter
Login? [anonymous]  enter

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

        [X] bsd
        [ ] bsd.rd
        [X] base34.tgz
```

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

File name? (or 'done') [xbase34.tgz]  +x*

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

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

File name? (or 'done') [done]  -ga*

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

        [X] bsd
        [ ] bsd.rd
        [X] base34.tgz
```

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

File name? (or 'done') [game34.tgz]  done
Ready to install sets? [y]  enter
Getting bsd ...
0% |*                                      |     16 KB     00:00
...
...

...
Getting xserv34.tgz ...
100% |***************************************|  14797 KB     00:08

Sets can be located on a (m)ounted filesystem; a (c)drom, (d)isk or
(t)ape device; or a (f)tp, (n)fs or (h)ttp server.
Where are the install sets? (or 'done')  done
Do you expect to run the X Window System? [y]  enter
Saving configuration files...done.
Generating initial host.random file...done.
```

3.9 Post-Installation

3.9.1 Time Zone Information and Example

The last stage of the installation sets the time zone.

```
What timezone are you in? ('?' for list) [US/Pacific]  US/Eastern
You have selected timezone 'US/Eastern'.
```

If you need to change this selection in the future, you can change the link in the file **/etc/localtime**, which points to an entry in the directory **/usr/share/zoneinfo**.

3.9.2 After Reboot

Once the system has finished installing, a reboot is needed. Note that the system will generate its OpenSSH keys on this first boot. This rather CPU-intensive process may take considerable time depending on the hardware type.[7]

3.10 Customizing the Installation Process

For some users, a customized installation may be desirable. This can be useful for large-scale rollouts, reinstallations from backups, installation support for hardware needed to complete the process, or local setups (e.g., a university campus). There are three major ways to customize the process, characterized by differing capabilities and levels of difficulty.

3.10.1 Creating Site-Specific Files

The easiest way to customize the installation process is to create a local archive to install along with the base system. This is handled gracefully by the normal installation floppy. Along with the installation sets (such as **base34.tgz**), a corresponding file named **site*.tgz** is listed. The version of this file matches the version of the installation sets being loaded (i.e., **site34.tgz** for an OpenBSD 3.4 installation). This file is installed last, after any other sets are installed (even if additional sets are chosen).

This file is unpacked in the root directory of the root filesystem, and therefore provides a way of customizing the installation for the particular site. For example, **/etc** files can be created for a local configuration (such as Kerberos files), additional or different binaries can be installed, and even some basic packages can be installed this way.

Because this file overwrites any file that previously existed, it cannot be used to remove files. However, it can be used to modify permissions on files and executables or to zero files out, thereby effectively removing them.

Creating this file is easy. After a system is installed and configured, simply use *tar* and *gzip* to build the **site*.tgz** file. In this example, a system is configured for a local Kerberos V realm and has a default user added to the **/etc/passwd** file. These will be installed on all other machines in this laboratory setup:

[7] SSH key generation took about 20 minutes on a 60 MHz SPARC system, but only a few seconds on a modern Pentium IV processor.

```
 modify the system for the local settings
#  tar -zcvf /tmp/site33.tgz /etc/krb5.conf /etc/kerberosV
/etc/passwd /etc/master.passwd /home/guest
 now copy the file to the FTP server next to the install sets
```

Now the other machines will receive the Kerberos configurations and the information for the guest user account, along with the home directory for the guest user account. Permissions and file modes are preserved by the installation program.

3.10.2 Jumpstarting Installations

The next method to customize an installation is to replicate the process using a custom installation process.[8] This method is similar to the RedHat "Kickstart" installations and the Sun Solaris "Jumpstart" method. The basic premise is that you build a system image that meets your needs and then automate the installation process. This option requires that the source code to the system be checked out and available for alteration.

One of the most challenging aspects of this installation method is the need to automate the disk partitioning. The easiest way to do this is to use the *disklabel(8)* command and to write the predefined label to the system. Once the prototype system is installed, the *disklabel* command can be used to automatically build a disk label for the disk you will be using:

```
#  disklabel -t wd0 >>  /etc/disklabel  (or sd0)
```

This method assumes that all of the systems will have identical hardware and disks. If this is not the case, but a small number of disk types will be seen instead, build the disklabel for each one and write them to the **/etc/disklabel** file.

Having set up the disks and fileystems, the next step is to edit the file **/usr/src/distrib/miniroot/dot.profile**. Change the lower part to read as follows:

```
        # Installing or upgrading?
        /install
fi
```

This will force an installation, rather than querying for the next step.

[8]This section is based on a message to the misc@openbsd.org mailing list by Chuck Yerkes on December 23, 2002, with the subject "Re: Jumpstart install." Chuck was also very helpful in private e-mails in clarifying some information.

Next, edit the files **/usr/src/distrib/miniroot/install.sh** and **/usr/src/distrib/miniroot/install.sub** to remove the "ask" functions. Instead, set a default answer to autopartition the disk and create filesystems (using *newfs*), and then mount these created filesystems. Setting the installation server and choosing the sets are relatively easy tasks at this point. This technique requires a tremendous amount of editing of the installation script, which is not detailed here. However, advanced users will find that it is a reasonably well-documented script and, by removing the "ask" functions and replacing them with variable settings, the process can be automated rather smoothly. Note that the hard disk's root filesystem is mounted on the boot system's **/mnt** filesystem, so all paths are relative to it.

The remaining challenge is to set the hostname and IP address of the system. A DHCP server can be used to assign both, which can then be parsed and inserted automatically into the booting system. This will requiring reordering of the settings, as the installation process asks for the hostname before the IP address (or the use of DHCP).

Having edited the installation scripts (in **install.sh** and **install.sub**) to automatically select the operations and choices, now rebuild all of the boot floppies (or choose one of the floppies):

```
# cd /usr/src/distrib/i386
# make
```

This will create the kernel and populate the ramdisks, leaving you with a floppy-sized image to copy to a floppy disk using *dd*, just as you would for a normal installation with the provided floppies:

```
# dd if=floppy34.fs of=/dev/fd0c
```

This command copies the entire floppy image to the floppy disk itself, suitable for booting.

Now, when you boot this crafted disk, the ramdisk will use its new **/etc/dot.profile** and launch into the script **install.sh**, which performs the installation process. This set will complete the installation process and leave the system ready for disk ejection and a final reboot.

This method is the most labor-intensive process, by far, and is recommended only for experienced developers. The level of detail required to make it work is beyond the scope of this book.

3.10.3 Customized Installation Floppies

The next method available to customize the installation process is to alter the kernel configurations of the boot floppies. This approach is needed for hardware that lacks,

support on the installation floppies provided—for example, a new Ethernet device or a disk driver.

By modifying the kernel configurations in **/usr/src/sys/arch/i386/conf**, which are used by the **Makefile** files in **/usr/src/distrib**, new kernels can be built. The kernel configurations **RAMDISKA, RAMDISKB**, and **RAMDISKC** are used by the installation floppies. **RAMDISK_CD** is used by the CD installation process. For other architectures, different kernel configuration files are listed.

The main step is to modify the kernel configuration to support your hardware as needed. For example, one may enable the Ethernet device *bge*, which is only supported on the CD configuration[9] for the installation sets. Once that step is complete, simply run *make* in **/usr/src/distrib/i386** or whatever architecture you are using (e.g., *sparc64, alpha*).

Note, however, that size is a crucial factor for the installation floppies. If a device is added, another one (or two) must be taken away. The easiest set of devices to remove comprises unneeded Ethernet drivers, which merely consume space. It may take several attempts to get the size changes correct.

It is much easier to just boot the larger CD image. More options and devices are available using that method.

3.11 Upgrading an Installation

For those who wish to upgrade an existing OpenBSD installation, a variety of approaches are feasible. The easiest is to use the binary upgrade path. Alternatively, upgrading from a source is available. These steps are discussed in Chapter 31.

[9] The *bge* device is a gigabit Ethernet device.