

Introduction to FreeBSD.

An Absolute Beginners Guide to FreeBSD

This project was started on Feb 13, 1997. If you have anything to comment on, please email us at fbsd-book@vmunix.com

Contents

- Part 1. *Introduction*
- Part 2. *Installation*
- Part 3. *Running FreeBSD*
- Part 4. *Customizing your FreeBSD System*
- Part 5. *Setting up your Network*
- Part 6. *Setting up the X Window System*
- Part 7. *System Administration*
- Part 8. *Practical FreeBSD Applications*

Part I:

Introduction

- 1. *Discovering FreeBSD*
- 2. *Where the Information is Stored online*
- 3. *General Guidelines for FreeBSD*
- 4. *FreeBSD Mailing Lists*
- 5. *Subscribing To A Mailing List*

1. Discovering FreeBSD

The book is designed for the new user and new system administrator of FreeBSD, This was written to help those who have no real UNIX background easily get started using FreeBSD. No matter what application, whether as a desktop system, or installed as a Internet server, FreeBSD has the power and the flexibility required to meet even the most demanding situations. FreeBSD, however, requires more of an administrative approach than most over-the-counter operating systems. Yet, because of this approach, FreeBSD requires less administration per functionality than Windows 95 and certainly several other operating systems. Given the correct approach, managing a FreeBSD system is simple and can be quite fun.

Even if you are just using FreeBSD as a desktop machine, it has the full power of a server and you are the new system administrator. Its hard to imagine yourself as a “system administrator” on a single Win95 machine, but technically it is a system, and you do administrate it. The reason it seems odd to call yourself a system administrator is because only one person can use a Win95 computer at a time.

FreeBSD is a multi-user system: Several people can access the same computer at the same time.

This includes more than the “File Serving” capabilities of WinNT and Novell Servers. The users connected to the FreeBSD machine have access all their normal functionality as though they were actually at the server console. Users can even interact if they choose. FreeBSD also gives you much more control over the system than DOS or Windows 95. You have the ability to modify system parameters “live”, not just edit the configuration file and have the changes take affect after you have rebooted. For example, you can change the IP address of your machine and then test immediately to see if it is working. You don’t have to wait 5 minutes for your computer to reboot to see if a change has worked.

More and more as the Internet becomes a part of our lives and shapes the way we do business, companies and individuals are looking for economical ways of enhancing their abilities. Companies, in particular, are now in the position of providing Internet and intranet services to attract customers and improve business processes. At the same time, although hardware costs are coming down, software and support costs are steadily increasing. For the individual, the Internet is an entrepreneur’s paradise. However, as they start up, they have to find inexpensive ways to provide the same range of services as large businesses on a budget that barely permits purchasing the hardware.

In a sense, UNIX is the touchstone of the Internet. There is now a strong emphasis on the standard protocols of the Internet---TCP/IP, DNS, HTTP, FTP, SMTP, POP3, IMAP, and the rest of the alphabet soup--- and most of those standards were developed or refined on the UNIX operating system. FreeBSD addresses both issues: It’s based on one of the standards of the UNIX world, BSD 4.4, optimized for common, inexpensive Intel-based hardware, and (as the name implies) it’s *free*.

Internet protocols are some of the strongest and most reliable features of FreeBSD. FreeBSD has many powerful WWW, DNS, and E-mail servers available. It also has a wide variety of TCP/IP related software available.

FreeBSD has may practical applications:

- Server Applications
 - Run a "turnkey" WWW server.
 - Handle Internet E-Mail(including Pop3 E-Mail and IMAP protocols).
 - Domain Name Service.
 - Allow dial-up access to the Internet through PPP.
 - Allow Windows 3.x and Win95 users to map network drives and share server resources.
 - Act as a network router.
 - Be an Internet firewall or gateway.
 - Network file sharing with a Novell server.
 - Translate network addresses, allowing you to conserve your registered IP addresses.

-

- An affordable UNIX X Window Workstation.
- Run a wide range of FreeBSD Desktop applications:
 - Word Processing
 - Spreadsheets
 - Database processing
 - Personal finance
 - Multi-media
 - Video conferencing
- Run applications from other platforms:
 - Linux
 - BSDi
 - SCO
 - Windows 3.1/DOS
 - Windows 95/NT
- Provide security control for a single user.
- Plus, lots of other company specific applications.

2. Where the Information is Stored online

Information about FreeBSD is available from several sources. Most of the documentation is on-line, in a wide variety of formats.

Internet Web Sites:

There are a lot of good web sites out there on FreeBSD. The simplest method to find information is to start with the Official FreeBSD web sites.

- <http://www.freebsd.org/> (Official Site)
- <http://www.freebsd.org/handbook.handbook.html> (FreeBSD Handbook)
- <http://www.freebsd.org/FAQ/FAQ.html> (FreeBSD FAQ)
- <http://www.de.freebsd.org/de/cgi/man.cgi/> (FreeBSD manual pages)

The above are part of the ongoing FreeBSD Documentation Project, which includes tutorials written by FreeBSD experts, and references to useful books.

There are also a lot of other web pages out there that have to do with FreeBSD and UNIX in General.

- <http://www.ugu.org>(Unix Guru Universe)

The people at [freebsd.org](http://www.freebsd.org) also provide a wide variety of mailing lists to talk about FreeBSD, report bugs, ask questions, keep you up to date on developments, and so on-- all the support you could want, and you don't have to pay anybody \$195/hour. You can find out more about these at <http://www.freebsd.org/support.html>

CD-ROM:

Most of the Documentation is also available on the Installation CD-ROM, and can be accessed through DOS or Windows. It might be a good idea to look at the online documentation, and if you anticipate problems, print off the appropriate sections before trying to install FreeBSD.

Documentation, in limited forms, is available during installation.

Your system:

On your system you can find the most documentation. Including:

- A full set of manual pages.
- The FreeBSD Handbook and FAQ in HTML format. You will find these in `/usr/share/doc`. They can be viewed with Netscape or Lynx.
- Examples included with the operating system.

3. General Guidelines for FreeBSD

Unix is case-sensitive.

FreeBSD treats uppercase and lowercase versions of a word as different words. Operating systems such as DOS and Windows 95 don't care if you use capital letters in filenames, or passwords. To those operating systems, `FileName`, `filename` and `FiLeNaMe` are all the same. In FreeBSD this is NOT true.

Keep the Caps Lock key off, almost everything is in lowercase.

E-mail me if you have any suggestions for this section. I am looking for things that you would have liked to have known about FreeBSD when you first started.

4. FreeBSD Mailing Lists

A mailing list is an easy way for a group of people to keep in contact about a specific topic. Once

you have subscribed to a particular mailing list, you will receive a copy of all the e-mail sent to that list. You will see people's questions as well as the answers. FreeBSD has several mailing lists designed to help people with their questions, to report problems, and to announce new releases.

Sending a question to the appropriate mailing list is an easy way to get a fast, correct answer. Responses come back usually in a couple of hours, but may take just minutes if someone knows the answer and responds immediately. You will not only get answers---you'll get *solutions* to your problems, because the participants on the list have often had the same experience, and a few of them were probably involved in writing the software in question.

Before you post a question, however, you will want to search the mailing list archives and read the on-line documentation mentioned previously. If you can't find what you are looking for in the documentation, pick the right list, and send the e-mail.

There are several lists available to open subscription. You will probably want to subscribe only to the lists you are actually going to use, and then just send letters to the other lists as the need arises. Otherwise, your message traffic will be unbearable.

You should avoid "cross-posting"; that is, posting the same question to more than one list. Many of the people who receive your e-mail are members of several lists, and if you cross-post, they'll see your message several times. Pick the list you think the message should go to, and if it's wrong, you will be told, and your message will probably get forwarded for you.

Most likely you will start out subscribing to the `freebsd-questions` List. All questions pertaining to how to use and run FreeBSD belong on this list.

When writing to the mailing list about a problem, try to include as much information as possible about the system you are using, such as:

- The version of FreeBSD you are using.
- Amount of RAM installed
- Hard drive type and size.
- Hardware make and model,
- All other computer configuration stuff related to the problem.
- Include all error messages exactly as they appear.

Note:

Also mention that you have searched the online documentation, and specify which documentation, because they may be able to direct you to something that you haven't looked at yet.

5. Subscribing To A Mailing List

Here is a list of available FreeBSD mailing lists. This is only a partial list, intended to get you started. This list changes as new lists are created to keep track of the development of FreeBSD.

When subscribing to a mailing list, there are two addresses to be aware of. The subscription address and the distribution address. The subscription address is the address where you send an e-mail to become part of the list. The subscription address for all FreeBSD lists is the same: `majordomo@freebsd.org`. “`majordomo@freebsd.org`” is an automated mailing list manager. He only understands requests that are specially formatted, such as `subscribe`, `unsubscribe`, and `help`.

Each FreeBSD mailing list has it’s own distribution address. This is the address that you send e-mail to if you want people to read it. Each member of the list receives a copy of the e-mail that you send to it’s distribution address. Do not send `subscribe`, or `unsubscribe` commands to distribution addresses.

- `freebsd-newbies@freebsd.org`
- `freebsd-questions@freebsd.org`
- `freebsd-isp@freebsd.org`
- `freebsd-current@freebsd.org`
- `freebsd-bugs@freebsd.org`
- `freebsd-chat@freebsd.org`
- `freebsd-announce@freebsd.org`
- `freebsd-stable@freebsd.org`
- Plus several other mailing lists.

To subscribe to a mailing list, send an e-mail message to `majordomo@freebsd.org`

In the body of the message type:

```
subscribe freebsd-questions
```

```
subscribe freebsd-announce
```

```
END
```

The above example will add your name to the `freebsd-questions` and the `freebsd-announce` mailing lists. Majordomo will send you a confirmation notice that you have been added to the mailing list, and shortly thereafter you will begin receiving e-mail from the list.

The `end` command tells majordomo that you are done issuing requests. If you don’t use `end` in your e-mail to majordomo, it will try to parse anything else in the document, such as your name and e-mail address, as a request for it to do something.

As soon as majordomo receives your request to join a mailing list, he will send you a confirmation e-mail. This e-mail will contain a special authorization code that you must re-send to majordomo. If

you do not send this authorization code back, you will not be added to the mailing list.

This protects you from being subscribed to a FreeBSD mailing list with out you wanting to. It also prevents people with bogus accounts from subscribing.

Part 2:

Installation

6. *Choosing a RELEASE to install*
7. *Getting A FreeBSD boot floppy from the Net.*
8. *Preparing Distribution Sets.*
9. *Using the Boot Floppy and Kernel Configuration*
10. *Installing FreeBSD with the boot floppy*
11. *Upgrading FreeBSD Versions*
12. *Post install Configuration*
13. *Adding Packages*
14. *Installing the Ports Collection.*

6. Choosing a RELEASE to install

FreeBSD is a work in progress, continually developing and improving. A release is a stable snapshot of the development process. It is given a version number followed by -RELEASE. There are several releases to choose from, each providing unique opportunities for stability and innovation.

Currently there are three different lines of development.

- 2.1.X-STABLE

The development on this branch has all but stopped. Only security and bug fixes make it in to this branch. The last version of this branch was FreeBSD-2.1.7.1-RELEASE. There will be no more versions of this branch.

- 2.2.X-STABLE

This branch of development contains all of the stable, production ready applications and improvements. This is not the newest technology, however it is rock-solid. The last version of this branch was FreeBSD-2.2.6-RELEASE and it is available on CD-ROM. The next version will come from this branch.

- 3.0.X-CURRENT

This branch is where all the development takes place. As things are developed, tested and proven to be stable, they are added to 2.2.X-STABLE. There has not be a stable version of 3.0-CURRENT released yet, however daily snapshots are available from current.freebsd.org

The three branches can be found in either “binary”(pre-compiled, ready to run) distributions, or as

“source” distributions.

- **BINARY RELEASES:**

- **-RELEASE**

Considered the most stable. This is the Official Release. The 2.2.X-STABLE branch is compiled, given a -RELEASE number and shipped out on CDROM. However this may not be the most current, depending on the release date. Currently, the latest version is FreeBSD-2.2.6-RELEASE.

- **-SNAP**

This is a binary release or "snapshot" of the -CURRENT or -STABLE version. It is not always stable. SNAPS are also available on CD-ROM, and are only for those who wish to easily track the state of FreeBSD's development, and are open to experimentation. -SNAP versions are available on a daily basis for all three branches of development.

- **SOURCE RELEASES:**

- **-STABLE**

This is the most current and stable release. It is the version that will become -RELEASE when it totally solidifies and has all the projected upgrades completed to it. It may however be incomplete in some areas depending on when you access it. -STABLE is also used to supply users who require absolute stability with minor bug fixes to a particular -RELEASE. For example, if you have installed FreeBSD 2.2.5-RELEASE, all of the bug fixes that have happened since the release date will be held in 2.2-STABLE (RELENG_2_2).

- **-CURRENT**

This is the most up to date version out there. It may or may not be stable depending on which day you download it. The parts of -CURRENT that are deemed stable get moved to -STABLE. This is where the bleeding edge developments takes place, and for that reason may not even be bootable depending on when you download it. Again, only recommended for people with an insatiable curiosity and FreeBSD developers. If you run -CURRENT you should be subscribed to the freebsd-current mailing list.

Note:

Source Releases require that you already have FreeBSD installed and running. The Source Releases are provided for upgrade, testing, and experimentation purposes. If this is your first install, you will want to use the -RELEASE distribution.

7. Getting A FreeBSD boot floppy from the Net.

When a computer starts up it requires a certain amount of instructions to be able to start the operating system. This is called ‘‘booting’’ or ‘‘boot-strapping’’ a computer. The ‘‘boot’’ code must be stored on a device that the computer considers bootable. In other words, there are set boot devices that the computer checks for the code required to start the operating system.

The two main boot devices are the floppy drive and the hard disk. CD-ROM Drives and certain network cards can also be configured as boot devices on some systems. Once the boot code finishes initializing the computer, it loads the kernel. The kernel is the single most important part of FreeBSD. The kernel establishes the link between the hardware and the software, it contains all the device drivers, executes any programs, and controls all system resources.

FreeBSD has a floppy disk that contains this ‘‘boot’’ information, the kernel, and setup/install utility. This is often referred to as a ‘‘boot floppy’’. The boot floppy is stored as a disk image, a sector by sector copy of the contents of the floppy disk. The contents of each and every sector of the disk is copied into the contents of a 1.44 Meg file, thus preserving the native FreeBSD formatting and file structure. A conventional file by file copy of the disk would not preserve the FreeBSD file system across platforms. When this disk image is transferred to a floppy disk, using a low-level disk tool called `rawrite.exe`, or `fdimage.exe`, it transforms the floppy disk from a DOS-formatted disk, into a FreeBSD formatted, bootable installation floppy disk.

There are two ways to go about this, depending on which operating system you have most readily available. DOS or UNIX.

Obtaining the File:

The easiest method of obtaining the file requires a Web-Browser such as Netscape, or a FTP client such as NCFTP or WS-FTP and a formatted floppy disk. A 1.44M disk drive is required because the disk images are 1.44Meg in size. Start your web browser or FTP client and open to:
`ftp://ftp.freebsd.org/pub/FreeBSD/`

Remember!:

* (Case is important!) FreeBSD recognizes the difference between Uppercase letters and Lowercase letters and considers them as different from each other. Windows and DOS systems don't.

Boot Floppies are -RELEASE specific. You need to know which version of FreeBSD you want to install and change to the matching directory. All of the versions available for download will be listed as a directory. If you have trouble selecting a -RELEASE, see the section on `Selecting a -RELEASE to install`. Once you have changed to appropriate -RELEASE directory, go to the `floppies` directory. Click on the `boot.flp` file and down load it to your computer.

Warning:

If any text appears on your screen, then it is not being downloaded to the harddisk, but to your screen. You will need to right-click on `boot.flp` and select `save link as in` order to save it to your computer. This happens mostly on Netscape.

Caution:

Do NOT Save it to your Floppy Drive at this point! Save it to a temporary directory on your hard disk or filesystem. You will transfer it to the floppy later using a special utility.

Caution:

If you are using FTP to download the floppy disk image, be sure to use "binary mode" to do the transfer. If you select "ASCII" mode, the file will be too big to fit on the floppy, and it will be corrupted.

The UNIX METHOD:

If you are creating the disk from a FreeBSD, or other UNIX workstation, you need access to the `dd` program and "write permissions" to the floppy disk.

Note:

This may require SuperUser access.

Insert floppy into the SERVER now. If you are using a telnet connection, Do not put the floppy into your local machine, it still belongs in the server.

At the prompt, type:

```
dd if=boot.flp of=/dev/fd0
```

The floppy drive light should come on and after a few minutes you will get a message telling you how many bytes of information were transferred and how fast. If you have any trouble or questions about `dd` read the `dd` man page.

The DOS METHOD:

To use the DOS method, you have to download the program `rawrite` from the FreeBSD FTP site. You can use Netscape or any other ftp client. Open to

```
ftp://ftp.freebsd.org/pub/FreeBSD/tools/
```

Then download the `rawrite.exe` file. To use the program, from the DOS prompt, type `rawrite` When it prompts for the source file type: `boot.flp` and press **ENTER**. When it asks for the destination drive type `a:` and press **ENTER**. Now it should go through a bunch of "writing Sector xx" stuff and the floppy disk light should come on for a while.

Win95 Users:

If you are using Win95, you should download and use `fdimage.exe` instead. It can be found in the same directory as `rawrite.exe`. However if you use `rawrite` under Win95, you must Maximize the DOS window by pressing **ALT + ENTER**

P.S. Be sure Win95 doesn't rename it to something like `rawrite(3).exe` instead, when you make several attempts to download it.

When the programs have finished writing to the floppy disk, you will have a FreeBSD Install/Boot floppy disk. Use it to install FreeBSD on all of your Machines.

8. Preparing Distribution Sets.

This section explains how to prepare a custom media to install FreeBSD from. These medias include:

- Floppy Disks
- MSDOS partitions
- NFS servers
- Local FTP Sites
- Existing FreeBSD Partitions

Note:

If you are not using one of these medias, please *SKIP* this section. (These are not the easiest or the recommended methods of installation. But they work.)

Each installation media looks for its files in a different directory. In order for the install program to find the installation files, you have to copy them into the correct directory.

For example, you want to install the Minimal Distribution Set from floppies. The `Minimal Distribution Set` requires the `/bin` distribution, ie. the `/bin` directory, be copied onto the floppies. The `/bin` directory can be found on the CD-ROM or from the FreeBSD FTP site, in the directory `/pub/FreeBSD/2.2.6-RELEASE/`. The `bin` directory and all its contents need to be copied onto MS-DOS formatted floppy disks. You will be able to fit about 6 files per disk. The files are named `bin.aa`, `bin.ab`, etc.

Important:

You will need to copy the file `bin.inf` onto the first disk.

When you get them on to the floppy disk you should be able to put it in a DOS machine and type `dir`

```

A:\>cd bin

A:\bin>dir

Volume in drive A has no label
Volume Serial Number is 13E4-3021
Directory of A:\bin

.                <DIR>          01-13-98  12:30p  .
..               <DIR>          01-13-98  12:30p  ..
BIN              AB            240,640  01-13-98  12:27p  bin.ab
BIN              AD            240,640  01-13-98  12:26p  bin.ad
BIN              AA            240,640  01-13-98  12:25p  bin.aa
BIN              AE            240,640  01-13-98  12:28p  bin.ae
BIN              AF            240,640  01-13-98  12:29p  bin.af
BIN              AC            240,640  01-13-98  12:28p  bin.ac
                6 file(s)          1,443,840 bytes
                2 dir(s)          13,312 bytes free

A:\bin>

```

When all the files have been copied onto the floppy disks, you will have a set of FreeBSD installation floppy disks. Use them when prompted for during the installation.

To install from a DOS partition, you must first copy all of the required directories to `C:\FreeBSD\`.

To install from an Custom FTP site, copy all of the files into a publically accessible directory followed by `FreeBSD/2.2.6-RELEASE/`.

An NFS install utilizes a Network Shared Filesystem. This the method of filesystem shareing used by UNIX over a network. A directory will need to be exported on the distribution site. Copy all the required directories onto `FreeBSD/2.2.6-RELEASE/` on that exported filesystem.

Listed below are the standard Distribution Sets that you can install and the directories that you will have to copy on to the media you are installing from.

Developer

- /bin
- /doc
- /manpages
- /info
- /proflibs

- /src
- /dict

X-Developer

- /bin
- /doc
- /manpages
- /info
- /proflibs
- /src
- /dict
- /XF8633

Kern-Developer

- /bin
- /doc
- /manpages
- /info
- /proflibs
- /src/sys
- /dict

User

- /bin
- /doc
- /manpages
- /dict

X-User

- /bin
- /doc
- /manpages

- /dict
- /XF8633

Minimal

- /bin

9. Using the Boot Floppy and Kernel Configuration

At this point you should have chosen a -RELEASE to install and have decided what medium you are going to use to install it. It can be installed over the network using FTP or NFS, from an existing FreeBSD or MSDOS partition, from a CDROM, or with floppy disks. All installation media types require a boot floppy, except bootable CDROM installs. This part covers how to use the Boot Floppy, regardless of which media you will be installing from.

If you will be installing from floppy disks, MSDOS, existing UNIX partition, NFS, or custom FTP site, you will need prepare your installation media. Read the section on *Preparing Distribution Sets*, to find out what to download and where to put it.

Before you boot with the Boot Floppy, you should check your CMOS Settings. You need to have the boot sequence set to boot from 'A' drive then 'C' drive. If this is going to be a Server, you probably don't want APM (Power management) enabled. Before you exit CMOS, double check to see that your IDE drives are all properly recognized and installed. SCSI drives should be checked using a SCSI utility.

Note:

If you don't know what each piece of hardware in your computer is, gather all the documentation. You will definitely need it. The best information comes from the packing list that came with your computer, if you still have it. It should tell you what kinds of devices you have installed. If you already have an operating system installed on the computer, you can use various methods of finding out what is in your system. If you have DOS, try using `msd`. If you have Win95, try **right-clicking** on *My Computer*, selecting *Properties*, selecting *device manager* and printing out the information there.

Now put the Boot disk in the A: drive, or the boot drive, and restart the computer.

After a few moments, you should see a screen like:

```
>> FreeBSD BOOT @ 0x10000: 639/31744 K of Memory

Usage: bios_drive:interface(unit,partition)kernel_name options
  bios_drive  0, 1, ...
  interface   fd, wd or sd
  unit        0, 1, ...
  partition   a, c, ...
  kernel_name name of kernel, or ? for list of files in root directory
  options     -a (ask name) -C (cdrom) -c (userconfig) -D (dual consoles)
```

```
-d (debug early) -g (gdb) -h (serial console) -P (probe kbc  
-r (default root) -s (single user) -v (verbose)
```

Examples:

```
1:sd(0,a)mykernel  boot 'mykernel' on the first SCSI drive when one IDE  
                   drive is present  
1:wd(2,a)          boot from the second (secondary master) IDE drive  
1:sd(0,a)?        list the files in the root directory on the specified  
                   drive/unit/partition, and set the default bios_drive,  
                   interface, unit and partition  
-cv               boot with the defaults, then run UserConfig to modify  
                   hardware parameters (c), and print verbose messages (
```

```
Use ? for file list or press Enter for defaults  
Boot:
```

This is the Boot Prompt. FreeBSD is giving you a chance to enter parameters different from the normal booting procedures. This is most often used to enter into the single user mode, or to boot from a different “kernel.” See the section on *Compiling Custom Kernels* for more detail on kernels.

If you don’t press anything, it will assume after a few seconds that you don’t want to enter anything extra and go on booting, using the default values. We want to use the default values at this point.

Next comes the kernel configuration menu:

```
Skip kernel configuration and continue with installation.  
Start kernel configuration in Visual Mode  
Start kernel configuration in CLI Mode (experts only)
```

These options give us a chance to modify our kernel and make it scan for devices in places other than the default settings. If we skip the kernel configuration, we accept all the default values, move on to the next section of the install, and will not have another chance to change them until we reboot. We can start the kernel configuration in either of two modes, CLI (Command Line Interface), or Visual Mode (Menu Driven Interface). The Visual Mode is by far the easiest to use and understand.

Of these three options, we want to go into Start kernel configuration in Visual Mode. But, before we do that, we need to have a basic understanding of IRQ and Ports and devices.

At this point you don’t need to know what they are so much, as you need to know that each device (hardware component) has one. You need to know what each device’s IRQ and Port number is. This is why you need the documentation on each piece of hardware in your computer. Some of the devices will have IRQ’s and Ports that you can set. These should be set before you begin your installation of FreeBSD. This may require jumper settings on the actual hardware itself, or use of a vendor supplied configuration utility to set it. The PCI devices won’t need to be set up by you in the kernel configuration; This utility only affects ISA devices.

Once you know what each device in your computer is and what it’s IRQ and Port # is, if it has one, then proceed into the visual kernel configuration.

```
--Active-Drivers-----21 Conflicts-----De  
Storage:                (Collapsed)  
Network:                 (Collapsed)  
Communication:          (Collapsed)  
Input:                   (Collapsed)  
Multimedia:              (Collapsed)  
PCI:                     (Collapsed)
```

```
Miscellaneous:
--Inactive Drivers-----
Storage:
Network:
Communication:          (Collapsed)
Input:                  (Collapsed)
Multimedia:
PCI:
Miscellaneous:
-----
```

This screen lets you modify or remove ISA devices from the boot scan. FreeBSD probes each device that it knows about when it boots up. All the devices in the “Active Devices” section will be probed, anything that is deleted from the “Active Devices” section will be moved into the “Inactive Devices” section and not probed. If you delete a device you need, you can go into the “Inactive Devices” section and re-activate it.

At the top of the screen you see “21 Conflicts” in Bold Print. This can be somewhat misleading, because there are not actually 21 physical conflicts. These are only *potential* conflicts. FreeBSD is telling us that if all the devices that it has been set to probe are present that there “would” be 21 Conflicts. FreeBSD is set up to probe all the devices it knows about and look for them at the most common IRQs and Ports. Since several pieces of hardware are commonly found at the same IRQ, it reports a potential conflict. For example, FreeBSD is probing for several different network cards at the same address, and since you only have one network card, the IRQ conflict is really non-existent.

Our job now is to look through the different sections of device drivers and de-activate or delete the ones that aren’t in our system. For each device that we do find that belongs in our system, we need to make sure that the IRQ and Port values are correct.

It is not absolutely necessary to delete all of the unused devices, because when FreeBSD probes for them and does not find them, it ignores them. However, it would be wise to delete any extra devices that do not belong to the system, because they may have long time out values. In other words, FreeBSD waits for a long time to make sure that they really don’t exist. Deleting them will speed up the boot process.

Most likely, all you will have to do is change the settings on a few devices and delete one or two others to speed things up. For example, In my system I have 2 SCSI drives, an NE2000 network card on IRQ 10, and a PS/2 mouse. I have to delete the IDE drivers, change the NE2000 driver (ed1) to IRQ 10, and enable the PS/2 mouse from the disabled list.

It still shows 18 conflicts, however, none of the equipment in conflict exists in my machine, so I can ignore it and continue on. I have resolved the conflicts for the equipment actually in my machine. If, after resolving conflicts for the equipment in your machine, you still have problems, you might try removing “all” the extra device drivers from the kernel configuration menu and checking for a conflict you might have overlooked.

If you have correctly erased “all” the extra devices and set the remaining devices to the correct IRQ and Port #, it should show no potential conflicts, with the exception of the PS/2 mouse. If it shows any remaining, it could be an actual conflict. Any actual conflict will result in both pieces of hardware not working properly. If you do end up with a potential conflict, after deleting ALL of the extra devices, you must move one of the offending devices to a different IRQ or Port (depending on where the conflict is) to avoid an actual conflict.

With this in mind, lets start with storage: With the storage section highlighted, press **ENTER** to

expand the tree. Now you should have lots of SCSI controllers and disk controllers listed. These will cover everything from CD-ROMs to Tape Drives to Floppy disks. They are in alphabetical order. These are only ISA devices, PCI devices are taken care of automatically.

The first four are:

```
--Active-Drivers-----21 Conflicts-----Dev-----IRQ
Adaptec 154x SCSI Controller      CONF          aha0
Adaptec 152x SCSI Controller      aic0         11
Buslogic      SCSI Controller     CONF          bt0
Floppy Disk Controller             fdc0         6
```

The above Buslogic and Adaptec controllers are in conflict trying to use the same Port. Deleting one of them, or changing one of them to an unused Port will eliminate a conflict. Just changing the Port that FreeBSD looks for will not solve your problem unless that is actually the Port that the hardware is set to. If the SCSI controller is set to Port 0x330 and so is the network card, one of them will have to change. Merely pointing FreeBSD to look for the SCSI controller in a different place, will appear to solve the problem, but will only succeed in misleading FreeBSD and hiding the fact that two devices are in conflict. The conflicting IRQ or DMA needs to be changed on the physical device, using a jumper or configuration program supplied by the vendor. Then FreeBSD needs to be pointed to the right place to look for it in this configuration menu.

The “Dev” listing is the file name that you will use to reference the hardware device in FreeBSD. It is the name of the actual device driver. All the devices are kept in a directory called /dev. So to access the floppy disk controller you would reference /dev/fdc0. Normally, however, you don’t access device controllers, just the devices they control. See the section on Accessing the Floppy.

If you don’t have SCSI in the system delete all the SCSI controllers, the same goes for IDE devices. Leave only the devices that actually exist in your system. If you delete an item and decide you need it, just press **TAB** and get it back from the Inactive Drivers section.

To change the IRQ or Port value of a device, press **ENTER** while the device is highlighted. This will open a box at the bottom of the screen allowing you to change the Port address, IRQ Number, Flags, and Memory Address. The Port address is a hexadecimal number. If it is shown in your documentation as a Port value of 330, it is probably 0x330. Use the **TAB** button to move between fields and then press **q** to quit and save the parameters. You can always go back and change them before you quit the kernel configuration editor.

When you are finally done with all of the kernel configurations, it may still show potential conflicts, but if you have configured everything correctly, there should be no actual conflicts.

PS/2 Mouse Users:

If you use a PS\2 Mouse, it will conflict with the Console, showing two conflicts. This is normal.

Be sure you have everything correct. Once you move past this screen, you won’t have another chance to change the devices, unless you reboot. Then press **q** to save these parameters and continue with the Installation.

10. Installing FreeBSD with the boot floppy

- 10.1. *An explanation of install screen options*
- 10.2. *Step 1) Format, fdisk, and partition the disks*
- 10.3. *Step 2) Allocate Filesystem Space*
- 10.4. *Step 3) Select Distributions*
- 10.5. *Step 4) Format and Configure the Media Type*

```

Welcome to FreeBSD! [2.2.6_RELEASE]
This is the main menu of the FreeBSD installation system. Please
select one of the options below by using the arrow keys or typing the
first character of the option name you're interested in. Invoke an
option by pressing [ENTER] or [TAB-ENTER] to exit the installation.

 1 Usage           Quick start - How to use this menu system
 2 Doc             Installation instructions, README, etc.
 3 Keymap          Select keyboard type
 4 Options         View/Set various installation options
 5 Novice          Begin a novice installation (for beginners)
 6 Express         Begin a quick installation (for the impatient)
 7 Custom          Begin a custom installation (for experts)
 8 Fixit           Enter repair mode with CDROM/floppy or start shell
 9 Upgrade         Upgrade an existing system
 c Configure       Do post-install configuration of FreeBSD
 1 Load Config    Load default install configuration
 0 Index           Glossary of functions

                [Select]      Exit Install
                [ Press F1 for Installation Guide ]
```

Welcome to an install of FreeBSD. From here you can install FreeBSD on a new machine, upgrade an existing machine, configure your machine after an installation, and add extra distributions and packages.

OR,

With a special ‘Fixit’ floppy disk or CD-ROM, you can troubleshoot problems in an existing system.

The install process is mostly automated. All the required information is collected before the installation actually starts. The `Novice` and `Express` installation options guide you through the information collection procedure. They let you install only after all necessary information has been gathered. The `Custom` install option gives you the flexibility of installing select sections without performing a complete install.

The install process consists of 4 Steps:

- Partitioning disk space
- Allocating filesystem space
- Selecting distributions to install
- Configuring the medium you will use to install.

Note:

The next section is an explanation of each option on the main menu. If you want to begin installing immediately, *skip* ahead to Step 1.

10.1. An explanation of install screen options

- 10.1.1. *Usage*
- 10.1.2. *Doc*
- 10.1.3. *KeyMap*
- 10.1.4. *Options*
- 10.1.5. *Novice*
- 10.1.6. *Express*
- 10.1.7. *Custom*
- 10.1.8. *Fixit*
- 10.1.9. *Upgrade*
- 10.1.10. *Configure*
- 10.1.11. *Load Configuration*
- 10.1.12. *Index*

10.1.1. Usage

```
usage
HOW TO USE THIS SYSTEM
=====

[press the PageDown key to go to the next screen when you finish
 reading this one]

The following keys are recognized in most of the dialogs you'll
encounter during this installation:

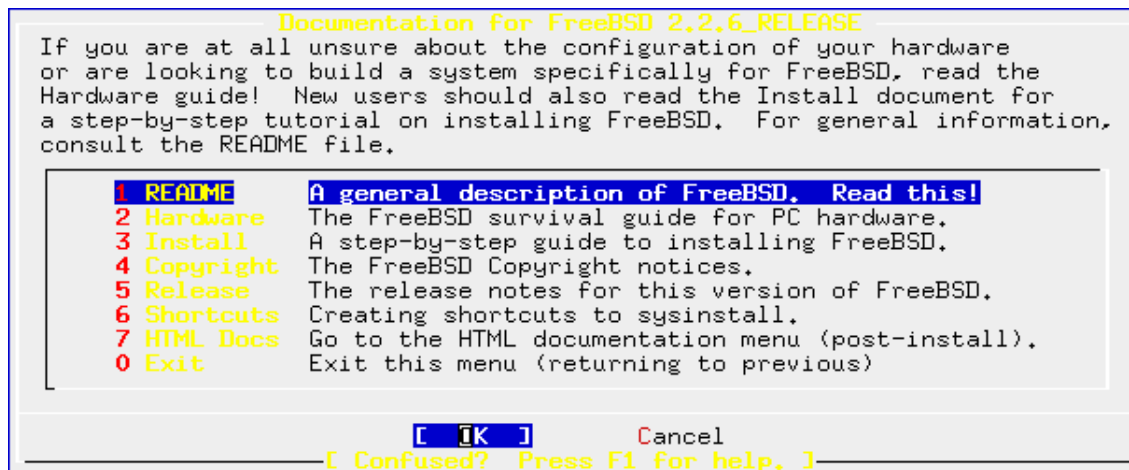
KEY          ACTION
----          -
SPACE        Select or toggle the current item.
RETURN       Finish with a menu or item.
UP ARROW     Move to previous item (or up, in a text display box).
DOWN ARROW   Move to next item (or down, in a text display box).
TAB          Move to next item or group.
RIGHT ARROW  Move to next item or group (same as TAB).
SHIFT-TAB    Move to previous item or group.
LEFT ARROW   Move to previous item or group (same as SHIFT-TAB).
PAGE UP      In text display boxes, scrolls up one page.
PAGE DOWN    In text display boxes, scrolls down one page.
F1           Display associated help text.

If you see small "^(-)" or "v(+)" symbols at the edges of a menu, it
( 32%)
[ OK ]
```

This screen is a tutorial on what each key does when you press it during the install. The **ENTER** key, sometimes called the **RETURN** key, will finish or exit you out of a menu whether or not you have selected anything. This might cause you to skip a menu section, thinking you have selected something.

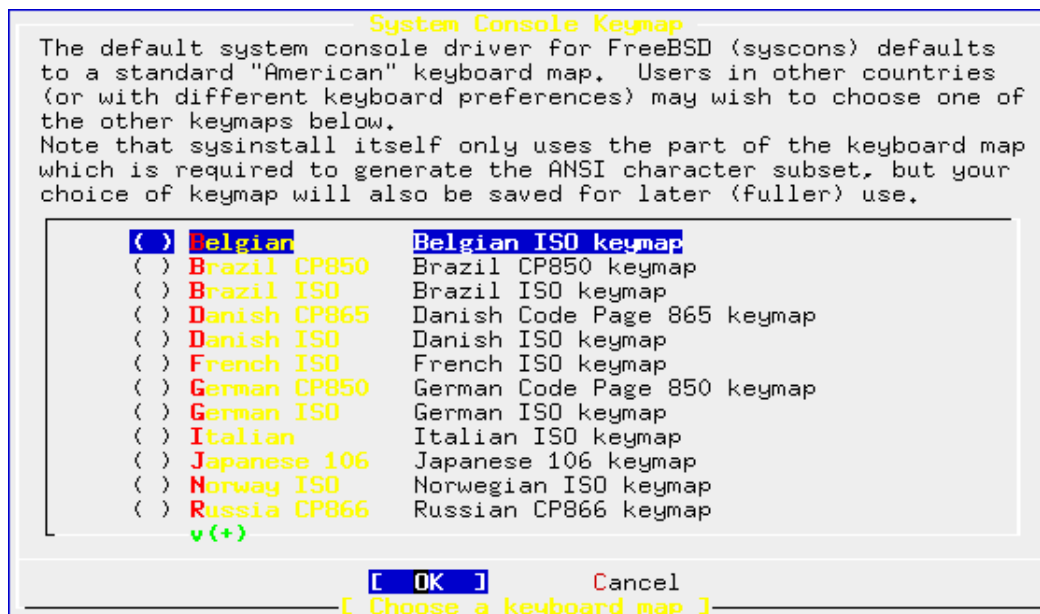
You must press the **SPACE BAR** to select most menu items. Usually, an x will appear next to the selected item.

10.1.2. Doc



FreeBSD comes with several very good sets of documents that are available during the install process. It is a good idea to read all the install documentation before attempting to install for the first time.

10.1.3. KeyMap



If you need special characters to write in the language that you are running FreeBSD on, you need to use the keyboard map for that country. This page lets you select from the available keymaps.

10.1.4. Options

Options Editor

Name	Value	Name	Value
NFS Secure	YES	Package Temp	/usr/tmp
NFS Slow	NO	Gated package	gated-3.5b3
Debugging	NO	PCNFSD package	pcnfsd-93.02.16
No Warnings	NO	Use Defaults	[RESET!]
Yes to All	NO		
FTP username	ftp		
Editor	/usr/bin/ee		
Tape Blocksize	20		
Extract Detail	high		
Release Name	2.2.6_RELEASE		
Install Root	/		
Browser package	lynx-2.7.1		
Browser Exec	/usr/local/bin/lynx		
Media Type	<not yet set>		
Media Timeout	300		

Use SPACE to select/toggle an option, arrow keys to move, ? or F1 for more help. When you're done, type Q to Quit.

NFS server talks only on a secure port

This page lets you tweak with the default settings. For the most part, everything should work fine with the default values. Unless you know what you are doing, you probably shouldn't mess with this section.

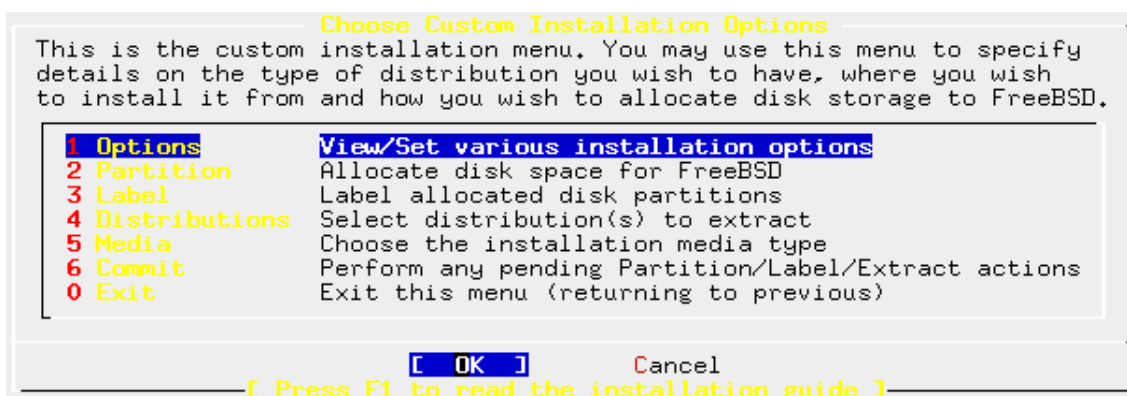
10.1.5. Novice

The `Novice Install` will walk you through each required part of the install process. Before each step, a help screen will appear and explain what is about to happen and what you are expected to do.

10.1.6. Express

Express is just like `Novice`, but without all the help. It walks you through all the important steps and collects the information required to start the installation.

10.1.7. Custom



The `Custom Install` allows you to do a specialized install, or a specific re-install, with out affecting previously installed components. For instance, this would allow you to add an extra hard drive to your system, or add/re-install a distribution. You could also repartition a hard disk or restore the boot-sector/boot-manager to your boot disk.

10.1.8. Fixit

```

Please choose a fixit option
There are three ways of going into "fixit" mode:
- you can use the 2nd FreeBSD CDRom, in which case there will be
  full access to the complete set of FreeBSD commands and utilities,
- you can use the more limited (but perhaps customized) fixit floppy,
- or you can start an Emergency Holographic Shell now, which is
  limited to the subset of commands that is already available right now.

1 CDRom   Use the 2nd "live" CDRom from the distribution
2 Floppy  Use a floppy generated from the fixit image
3 Shell   Start an Emergency Holographic Shell

[ OK ]      Cancel
[ Press F1 for more detailed repair instructions ]
```

If anything should happen to your system that would cause it not to boot properly, the `Fixit` option will give you access to your filesystem. The `Fixit` option is an advanced feature. Knowledge of how to mount filesystems is required to gain access to your hard disks.

10.1.9. Upgrade

```

upgrade
Welcome to the 2.1.x (or 2.0.5) -> 2.2 upgrade procedure!

It must first be said that this upgrade DOES NOT take a particularly
sophisticated approach to the upgrade problem, it being more a
question of providing what seemed "good enough" at the time. A truly
polished upgrade that deals properly with the broad spectrum of
installed 2.0.5 / 2.1.x systems would be nice to have, but until that
gets written what you get is this - the brute-force approach!

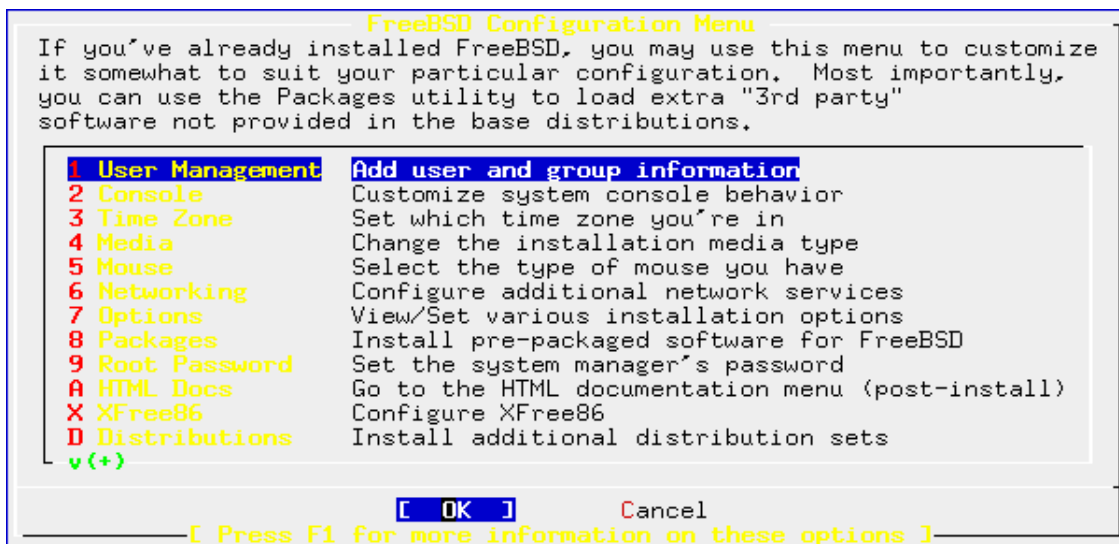
What this upgrade will attempt to do is best summarized thusly:

1. fsck and mount all file systems chosen in the label editor.
2. Ask for a location to preserve your /etc directory into and do so.
3. Extract all selected distributions on top of your existing system.
4. Copy certain obvious files back from the preserved /etc, leaving the
  rest of the /etc file merge up to the user.
5. Drop user in a shell so that they may perform that merge before
  rebooting into the new system.

And that's it! This "upgrade" is not going to hold your hand in all
[ OK ] ( 51%)
```

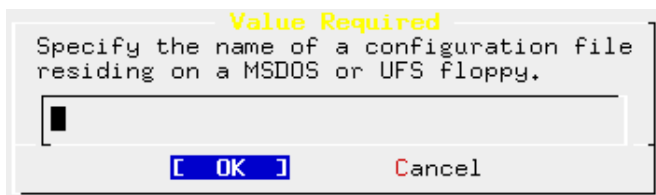
The `Upgrade` option allows you to convert an existing FreeBSD machine to a new version of FreeBSD, without re-formatting your disk drives, or losing all of your data. An upgrade is a lot like re-installing a distribution set. It copies all the distributions you choose over the top of you existing files, replacing the ones with the same name. It does not delete extra files that it finds.

10.1.10. Configure

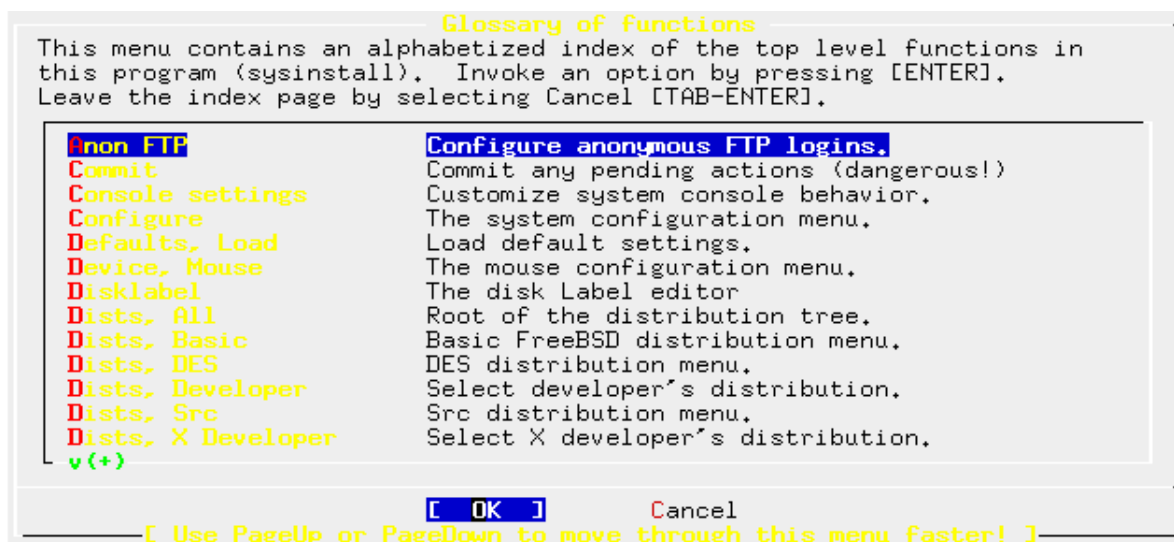


This menu helps you setup your basic networking services, install extra packages (pre-compiled “3rd party” software), add additional distributions and prepare your system for general use.

10.1.11. Load Configuration



10.1.12. Index



This screen could be looked at as a “Custom Install”. Instead of being organized into an install menu structure, all of the install commands have been extracted, alphabetized, and placed on a single menu. This makes it an alphabetized list of every command available during the install process. Pressing **ENTER** at this point, will execute the highlighted option as though you had selected it from an active menu.

10.2. Step 1) Format, fdisk, and partition the disks

- 10.2.1. *A single disk FreeBSD ONLY install*
- 10.2.2. *A single disk Multi-OS install*
- 10.2.3. *A Multi-disk FreeBSD ONLY install*
- 10.2.4. *A Multi-disk Multi-OS install*

Here we are going to prepare our drives for use with FreeBSD. If you already have an operating system installed on your disks, and don't plan on using the whole disk for FreeBSD, you will need to free up a partition to install FreeBSD on. There are several methods of doing this. You can delete your stuff, re-arrange the partitions, and re-install it. (Cleanest, but takes the most work.) Or you can re-partition it using a program that preserves your information. There are several programs that can repartition your drives with out losing your information. A shareware program FIPS will work, or there are several commercial packages, such as "Disk Magic".

`fips.exe` can be found at [ftp.freebsd.org/pub/FreeBSD/tools/fips.exe](ftp://ftp.freebsd.org/pub/FreeBSD/tools/fips.exe).

If you use DOS fdisk to make a new partition, you will be required to delete the existing partition. However, with FIPS and other such programs, you can "shrink" an existing partition, freeing up room for a new partition. FreeBSD can be installed on any available partition, including extended DOS partitions, logical drives, and Non-DOS partitions.

Note:

Once you have a partition available for FreeBSD, it does not matter if it is formatted, the install process will format it for use with FreeBSD. It makes no difference whether DOS or any other OS is installed on it.

Four possible fdisk scenarios are outlined, based on how many hard drives you have available, and whether FreeBSD is the only OS installed.

10.2.1. A single disk FreeBSD ONLY install

This is the simplest install scenario. It is when you only have one disk in the computer and you want to dedicate it to FreeBSD.

In the fdisk editor, delete any partitions that may exist by moving the cursor to the partition and pressing **d**. Then press **a** to use the entire disk as shown in the figure below.

This is an example of a 2.1 Gig SCSI disk. It does not matter whether the disk is SCSI or IDE.

```

Disk name:      sd0                      FDISK Partition Editor
DISK Geometry: 261 cyls/255 heads/63 sectors = 4192965 sectors

  Offset      Size      End      Name      PType      Desc      Subtype      Flags
-----
  0           4194058  4194057  sd0s1     3          freebsd   165          C>

```

The following commands are supported (in upper or lower case):

```

A = Use Entire Disk      B = Bad Block Scan      C = Create Slice
D = Delete Slice        G = Set Drive Geometry  S = Set Bootable
U = Undo All Changes    Q = Finish              W = Write Changes

```

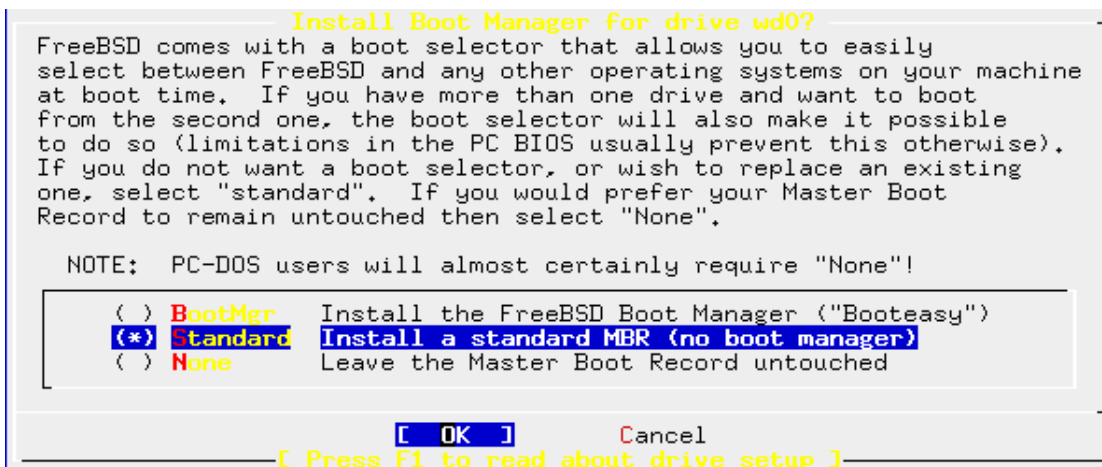
Use F1 or ? to get more help, arrow keys to select.

□

Now, press **q** to quit.

Install the standard MBR (Master Boot Record)

Use the **SPACEBAR** to select standard and press **ENTER** to move on.



Move on to Step Two, “Allocating filesystem space”.

10.2.2. A single disk Multi-OS install

This is one of the more complex scenarios, you don’t have to worry about which drive to boot from like you do with Multi-disk setup-ups, but you need to be careful about which OS you install First. DOS and FreeBSD are friendly to each other; Window95 and WinNT are not friendly to FreeBSD. They both want to try and take over the Boot Sector (MBR). Also, DOS and Win95 usually need to be installed on the first 1026 sectors of the first hard disk to avoid the 530Meg limit imposed by DOS.

Now you should see several partitions. The first one says that is it unused, starts at 0 and goes 63 sectors. This is your boot sector. There is nothing you can, or want to do with this at this time. Then

you should see a section starting at 63 and named `sd0s1` or `wd0s1` depending on whether you have SCSI or IDE disks. `sd0` is the name for a SCSI disk and `s1` is the first partition we made on the drive.

You should see a big section that is unused. This will be used to create our FreeBSD partition.

If there is no partition marked `unused`, then the partition that you have reserved already has information on it, and will need to be deleted and recreated as a FreeBSD partition. Press `d` to delete a highlighted partition.

For example, you have 3 Gb of space on one disk. It is partitioned into a `C:` drive (2Gb), and a `D:` drive(1Gb). The plan is to keep your 'C: drive' and just use the 'D: drive' for FreeBSD. This is what your partition table looks like:

offset	Size	End	Name	PType	Desc	Subtype
0	63	62	-	6	unused	0
63	4193217	4193279	<code>wd0s1</code>	2	<code>fat</code>	6
4193280	2080512	6273791	<code>wd0s2</code>	4	<code>extended</code>	5
6273792	8064	6281855	-	6	unused	0

Here you want to delete the 'extended' partition and re-create it as a FreeBSD partition. Just highlight it and press 'd'. Then press 'c' to create a new one. Accept the default size.

The partition labeled `fat` is a DOS Partition. The partition labeled `extended` is an Extended DOS Partition. (ie not the primary DOS Partition)

The sizes are shown in 512 byte blocks instead of 1K blocks. So you need to cut the size in half. The size '4123217' partition is the 2G disk. The '2080512' size partiton is the 1G drive. The '8064' sized chunk is unallocatable space due to cylinder boundaries not matching up.

To create our FreeBSD partition, highlight an unused partition and press `c` for Create. If you want to use the remaining amount of space for FreeBSD, accept the number that is written in the box, otherwise enter a new number. There are two ways to enter the amount that you want. First you can specify the exact number of sectors that you want the partition to be. Or, you can specify the size in Megabytes by typing in the number followed by an `M` For example, If you want a partition of 1000 Megabytes, you type: `1000M` in the box.

The following graphic shows a 1000M DOS partition, labeled 'fat', as `sd0s1`.

It also shows a 1100M FreeBSD partition.

```

Disk name:      sd0
DISK Partition Editor
DISK Geometry: 261 cyls/255 heads/63 sectors = 4192965 sectors

  Offset      Size      End      Name      PType      Desc      Subtype      Flags
-----
    0          63         62       -         6          unused      0
    63      2040192    2040254  sd0s1     2          fat         6
  2040255    2152710    4192964  sd0s2     3          freebsd    165      C
  4192965     1093     4194057  -         6          unused      0

```

The following commands are supported (in upper or lower case):

```

A = Use Entire Disk      B = Bad Block Scan      C = Create Slice
D = Delete Slice        G = Set Drive Geometry  S = Set Bootable
U = Undo All Changes    Q = Finish              W = Write Changes

```

Use F1 or ? to get more help, arrow keys to select.

Note:

By keeping the partition compatible with other partitions, a small portion of the hard drive might become unusable. The last partition shown on the graphic, starting at offset 4192965, is unusable. This is nothing to worry about, because it is a very small part of the available disk space. It is only 1093 Sectors in size.

When you are done creating your partitions, press **q** to quit and move on to the next part of the install.

Next it is going to ask how you want to boot the disk. Because we are booting several OSes from one disk, we need to install a boot manager.

```

          Install Boot Manager for drive wd0?
FreeBSD comes with a boot selector that allows you to easily
select between FreeBSD and any other operating systems on your machine
at boot time.  If you have more than one drive and want to boot
from the second one, the boot selector will also make it possible
to do so (limitations in the PC BIOS usually prevent this otherwise).
If you do not want a boot selector, or wish to replace an existing
one, select "standard".  If you would prefer your Master Boot
Record to remain untouched then select "None".

NOTE:  PC-DOS users will almost certainly require "None"!

(*) BootMgr  Install the FreeBSD Boot Manager ("Booteasy")
( ) Standard Install a standard MBR (no boot manager)
( ) None     Leave the Master Boot Record untouched

[ OK ]      Cancel
[ Press F1 to read about drive setup ]

```

This screen lets you select the Boot Manager that comes with FreeBSD. At this screen all you have to do is select `BootMgr` and it will install the BootEasy Boot Manager on to your disk.

Since `BootMgr` is the default, just press **ENTER** to go on.

Note:

There are other Boot Managers available, but this is the only one available during the install. If you have a boot manager already installed, such as OSBS or the OS/2 Boot loader, select `NONE` from the menu.

Move on to Step Two, “Allocating filesystem space”.

10.2.3. A Multi-disk FreeBSD ONLY install

This is a very common server-type setup. It is fairly easy to setup and very reliable.

Now we need to partition each of our disks. We will select one disk at a time and partition it, repeating the same process for each disk.

```

Select Drive(s)
Please select the drive, or drives, on which you wish to perform
this operation.  If you are attempting to install a boot partition
on a drive other than the first one or have multiple operating
systems on your machine, you will have the option to install a boot
manager later.  To select a drive, use the arrow keys to move to it
and press [SPACE].  To de-select it, press [SPACE] again.

Select OK or Cancel to leave this menu.

[ ] wd0  wd0
[ ] wd1  wd1

[ OK ]      Cancel
—[ Press F1 for important information regarding disk geometry! ]—
```

Select a disk by moving the cursor to the appropriate disk and pressing the space bar.

Caution:

* (Do NOT press `ENTER` at this point, because this will move us out of the FreeBSD fdisk utility, with out partitioning any disks.

Since this is a FreeBSD only install, we need to delete any partitions that are there and create new ones for FreeBSD. Move your cursor to each partition and press `a` until there is only one partition that says `unused`

Now, just press `a` to use the entire disk.

It will ask you if you want to keep the disk partition compatible with other OS's by making it a true partition. If you are using SCSI disks and are not going to be using any other OS on this system, it is fine to say `no` and dedicate them to FreeBSD. If you are using IDE drives, it might be a good idea to answer `yes` and let them conform to regular partitioning methods.

The following graphic is a 2.1 Gig SCSI disk that has NOT been keep compatible with other OS's, therefore it shows no Boot Sector.

```

Disk name:      sd0
DISK Geometry: 261 cyls/255 heads/63 sectors = 4192965 sectors

  Offset      Size      End      Name      PType      Desc      Subtype      Flags
  -----
  0      4194058      4194057      sd0s1      3      freebsd      165      C>

The following commands are supported (in upper or lower case):

A = Use Entire Disk      B = Bad Block Scan      C = Create Slice
D = Delete Slice        G = Set Drive Geometry  S = Set Bootable
U = Undo All Changes    Q = Finish              W = Write Changes

Use F1 or ? to get more help, arrow keys to select.

[]

```

We have just completed the fdisking the first disk; press **q** to quit and it will return us to the *Select Drives* menu. The disk we just finished will have a **x** in the box and the remaining disks still need to be fdisk'd. Repeat this procedure until all disks have been fdisk'd.

When all disks have been fdisk'd, press **ENTER** at the 'Select Drives' menu.

Move on to Step Two, "Allocating filesystem space".

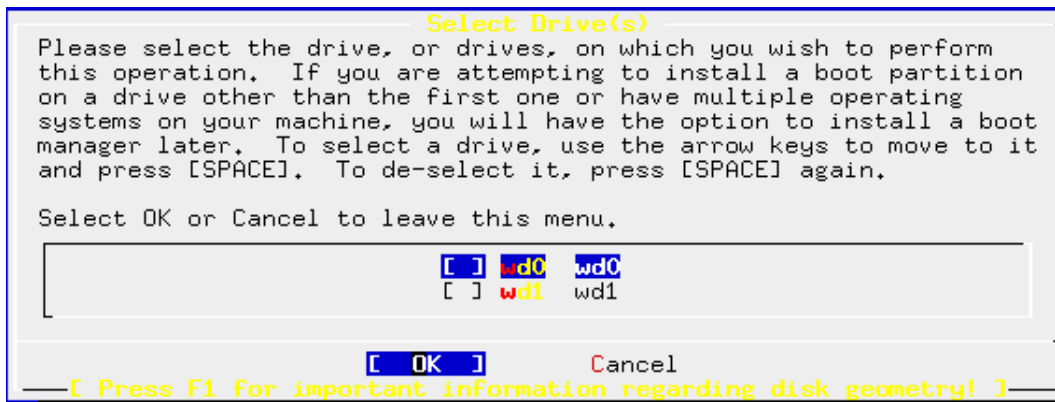
10.2.4. A Multi-disk Multi-OS install

A Multi-disk Multi-OS installation can take on just about any format imaginable. You can have one OS for each disk. Multi-OS's on a on Multiple disks; or any combination of the two. If you are putting more than one OS on a single disk, it will work exactly like the Single disk Multi-OS installation described previously. Of the several options available, one of the easiest, and most sane, is to put one OS per disk. For example, you would put DOS/Win95 on the First disk and FreeBSD on the second disk. I find this to be safer, because sometimes DOS tries to take over the first drive with out telling you and formats it without asking your permission.

One OS per Drive/Two Drive system.

In this example, we have two drives. We are going to put Win95 on the first drive and FreeBSD on the second. Win95 should already be installed and running on the first disk.

We are going to install the Boot Manager on the Win95 disk, because it is the boot disk. To do this, select the first disk, in this case sd0, and install the boot manager. At the *Select Drives* screen, move the cursor to the first disk and press the **SPACE BAR**.



This will bring up the fdisk partitioning editor.

At this screen, you should see one DOS partition. Do NOT delete or change it. just press `q` to quit out of it. Then the Boot Manager screen should come up.

```

Disk name:      wd0      FDISK Partition Editor
DISK Geometry: 524 cyls/32 heads/63 sectors = 1056384 sectors
  
```

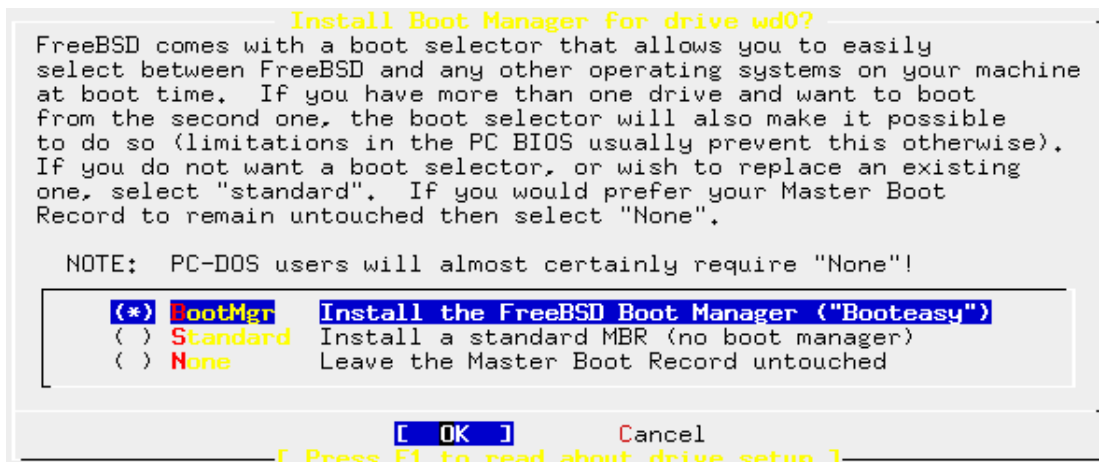
Offset	Size	End	Name	PType	Desc	Subtype	Flags
0	63	62	-	6	unused	0	
63	1056321	1056383	wd0s1	2	fat	6	

The following commands are supported (in upper or lower case):

A = Use Entire Disk	B = Bad Block Scan	C = Create Slice
D = Delete Slice	G = Set Drive Geometry	S = Set Bootable
T = Change Type	U = Undo All Changes	W = Write Changes

Use F1 or ? to get more help, arrow keys to select.

Here we want to select BootMgr; since it is the default, we just need to press **ENTER**.



This will bring us back to the Select Drives screen. Now, just move the cursor to the next disk and

press the **SPACE BAR**. From here it is exactly like a single disk FreeBSD install. Just delete any existing partitions by moving the cursor to them and pressing **d**. When they have been all deleted, press **a** to use the entire disk for FreeBSD. Then press **q** to quit. You will need to install the `BootMgr` on this disk also.

10.3. Step 2) Allocate Filesystem Space

- 10.3.1. *Workstation\Desktop system*
- 10.3.2. *Server (More than 100 People)*
- 10.3.3. *Internet Server (E-mail/Pop3/Web Pages/FTP)*
- 10.3.4. *News Server*

The FreeBSD Filesystem is one big directory structure. There is no noticeable separation of drives, or partitions, just one big filesystem. All additional hard drives must be assimilated into the directory system. This is accomplished by mounting each partition, sometimes called a “slice”, as a sub-directory in the filesystem. A partition can mount to, or attach to, any existing directory, preferably an empty one.

Next, we will enter the disk label editor. This will allow you to distribute your disk space throughout the filesystem. At a minimum, you will need to allocate disk space for a `/` directory and for “Swap Space”.

Space is allocated for a directory by mounting a partition, or “slice” of a partition, to it. If you do not allocate disk space for a directory, it will use disk space from it’s parent directory. Below is an explanation of the standard directories that FreeBSD installs and whether you should consider allocating space for it. Subsections of this chapter describe extra allocations you should consider based on the anticipated use of your system. These are in addition to the basic allocations mentioned below.

- `/`

`/` is your “root” directory, not to be confused with the `/root` directory, which is the home directory of the user “root”. All other directories are sub-directories to `/`.

Like an upside down tree, “root”(`/`), sits at the top and everything branches outward from it. The root directory doesn’t need to be very big, 32-50 Meg. is sufficient. It just needs to have enough room to hold the kernel and various system configuration files.

- `/etc`

The `/etc` directory contains all of your system configuration files. You should not need to mount a partition here, nor do you want to. In the event that you are unable to boot properly, you would have trouble accessing your configuration files.

- `/usr`

The bulk of the system is contained in the `/usr`, or UNIX System Resources directory. Depending on your systems intended use and available disk space, you should reserve at least 100M for the `/usr` directory. Usually you would allocate the most disk space for this partition.

- Swap Space

You must create at least one swap partition. You can have several swap partitions, and it is best to spread them out over all your disks. You don't need to specify a mount point, because the swap partitions are not accessible from the filesystem. Your swap partitions should total at least twice the size of the amount of RAM memory you have installed in your machine.

Note:

A swap partition must not occupy the first sector of a disk. You must put a filesystem partition on it first. It is good to have one swap partition per disk.

- /var

The /var directory holds various system logs and databases. It also holds a lot of transient files, such as queued e-mail and spooled print jobs. Depending on the role of your system, this directory can require several hundred Meg., or only a few Meg. You should mount a partition there of at least 32Meg. to take the strain off of your root directory. Some directories you might want to consider mounting volumes on are /var/mail and /var/log, depending on your anticipated usage.

- /bin and /sbin

These directories contain all the basic binary files that make up the core FreeBSD system. Do NOT mount partitions to these directories. They contain the utilities that allow you to mount filesystems. If you booted improperly, you could not gain access to any of your filesystems.

- /root

This is the Home directory of the user root. Unless you mount a partition to a sub-directory, it uses available disk space from its parent directory. Therefore you want to be careful what you put in the /root directory, because it uses up the available disk space from the / directory. If you find your / filesystem happens to be full, you might want to check this directory for excess files.

- /dev

This directory contains "file" representations of all of the devices that exist in your system. You don't need to mount anything here.

10.3.1. Workstation\Desktop system

Unless you have special plans, on a system like this, just use the defaults. Press **a** to use the defaults. This option will create a 32Meg / partition, a 32 Meg /var partition, calculate a swap partition based on available RAM memory, and allocate the rest as a /usr partition.

10.3.2. Server (More than 100 People)

The more people you have on the system, the more you want to think about making a /home partition. You should decide on a minimum amount of disk space that you are going to allot for each user. Say 3-4 Megs of disk space for each user, and make /home partition that has enough space to

hold all the anticipated user files.

10.3.3. Internet Server (E-mail/Pop3/Web Pages/FTP)

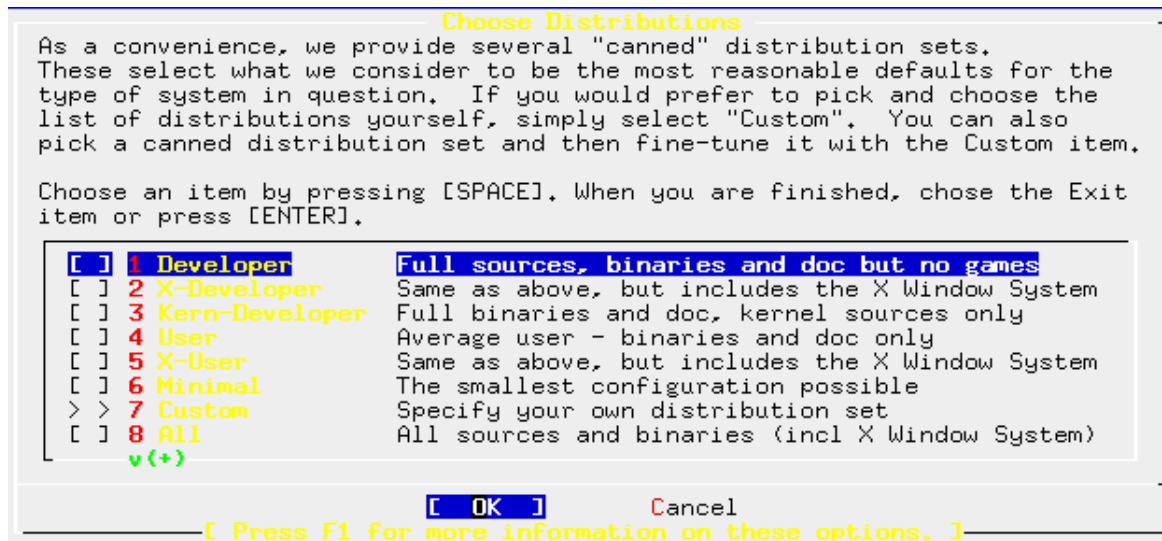
On an e-mail server, you want to add extra space to the directory `/var`. The directories `/var/log` and `/var/mail` receive extra heavy use in Internet servers. If you anticipate a large volume of e-mail, you might want to add a separate `/var/mail` partition.

If this is a print server, `/var/spool` might get heavier than normal usage. However, spooled print jobs don't stay very long.

10.3.4. News Server

A News server is a very advanced project, but I mention it here so you can plan disk space for it. You will want to use several 2.1 Gig SCSI disks, and look into `ccd`. `ccd` can be used to stripe several disks together, and also mirror hard disks.

10.4. Step 3) Select Distributions



A distribution is a collection of files that make up part of the operating system, sort of like separate components that can be added individually. A distribution set is a selection of distributions. Six "typical" distribution sets have been prepared to make it easy for new users to select distributions based on the intended use of the system. A list of the distributions installed by each distribution set is included at the end of this chapter.

You also have the option of selecting your own distribution set. Every distribution set includes the `bin` distribution. The `bin` distribution contains all the files required to make FreeBSD run. Everything else is optional. You'll notice that the `minimal` distribution set only installs the `bin` distribution.

One of the strong points of using an operating system that includes the Source Code is the ability to adjust for special hardware. If you have special hardware, such as a multi-port serial card, or Symmetric Multi-Processors, you will need to be able to re-compile and generate a custom kernel. Installing the kernel source code also allows you to streamline your kernel for optimum

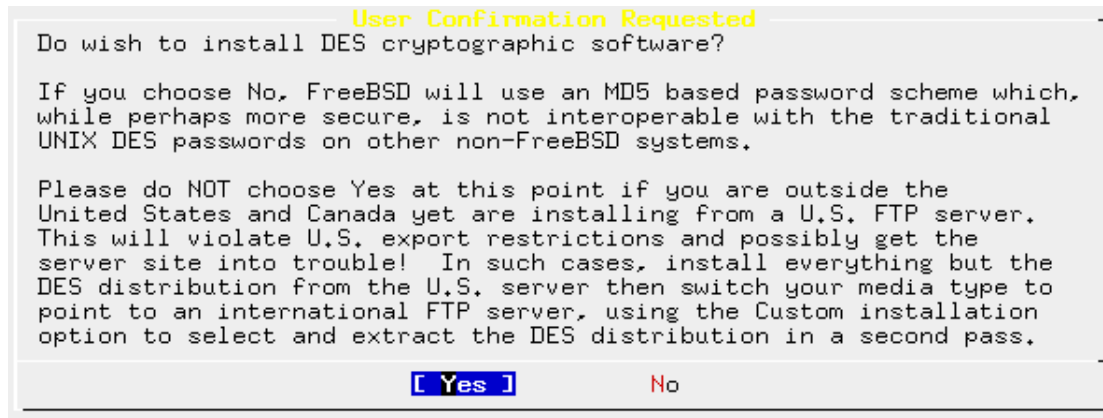
performance, by removing unused device drivers.

I would recommend installing the source code for creating a custom kernel. The `Developer`, `X-Developer`, and `Kern-Developer` distribution sets all include the kernel source code.

If you have an SVGA video card and SVGA monitor, I would recommend installing the X Window System binaries. `XFree86` is the free version of X Window System for FreeBSD. In order to configure X Window System properly, you need to know the Brand of Video Card, the Refresh rates for your monitor, and what kind of mouse you have.

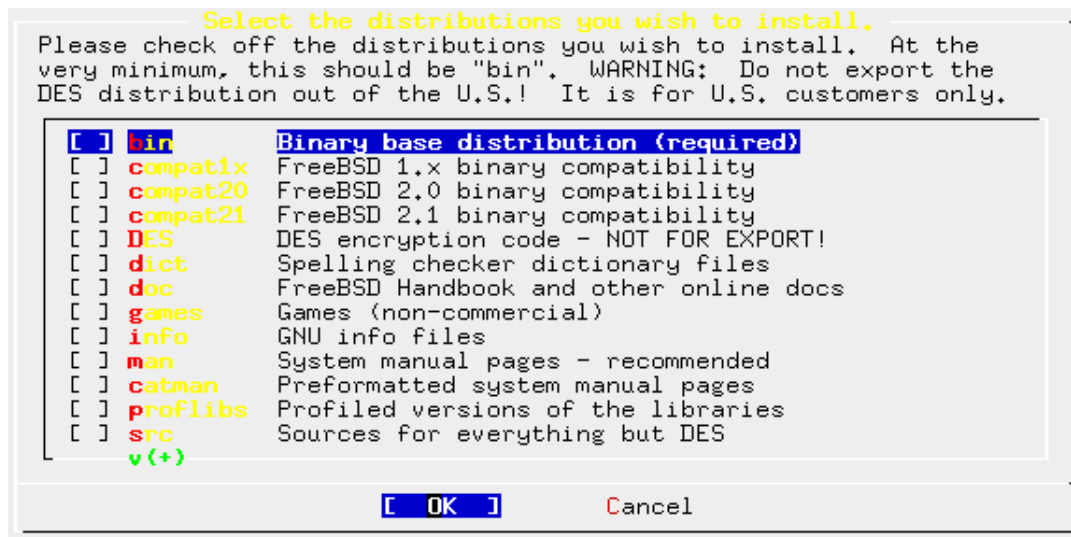
`X-Developer` and `X-User` both include the X-Windows system. I would recommend the `X-Developer` because it includes the kernel sources also.

After you select one of the distribution sets, it ask you if you want to install the DES encryption components.



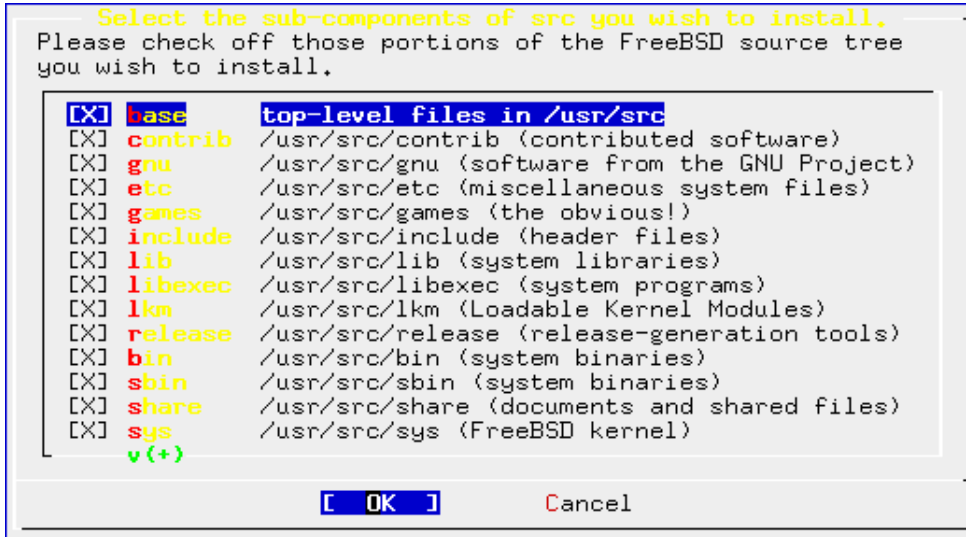
If you are outside of North America (USA or Canada) don't install it over the Internet from a North American site. The copy of DES on the CD-ROM is fine. DES technology originating from the US is not exportable, so you'll have to pick an overseas server to install from (which is the logical way to do it anyhow...)

Selecting `Custom` from the distribution menu allows you to specify which components to install.



This screen also lets you add distributions, or re-install corrupted distributions. If you are adding distributions to an existing system, you probably don't want to select BIN. When installing a new system, I would recommend the following as very necessary: BIN, DOC, MAN, XFree86, and Selected SRC sets.

When you select SRC from the Custom Distribution Menu, it asks you which sets of source code you want to install.

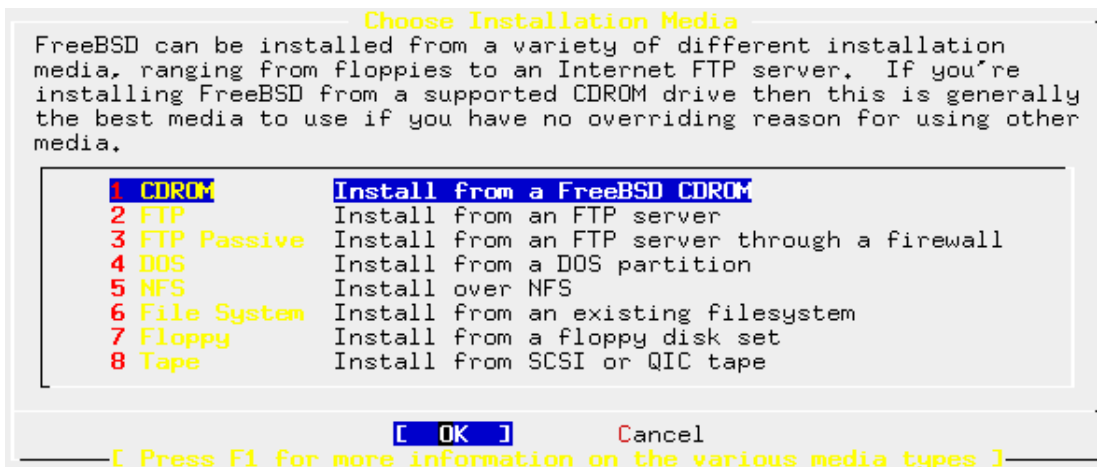


At the very least, I would install sys, the FreeBSD kernel sources. This will allow you to build your own kernel.

10.5. Step 4) Format and Configure the Media Type

10.5.1. *Installing from a FreeBSD CD-ROM*

10.5.2. *Installing from an FTP Site*



10.5.1. Installing from a FreeBSD CD-ROM

To install from a CD-ROM:

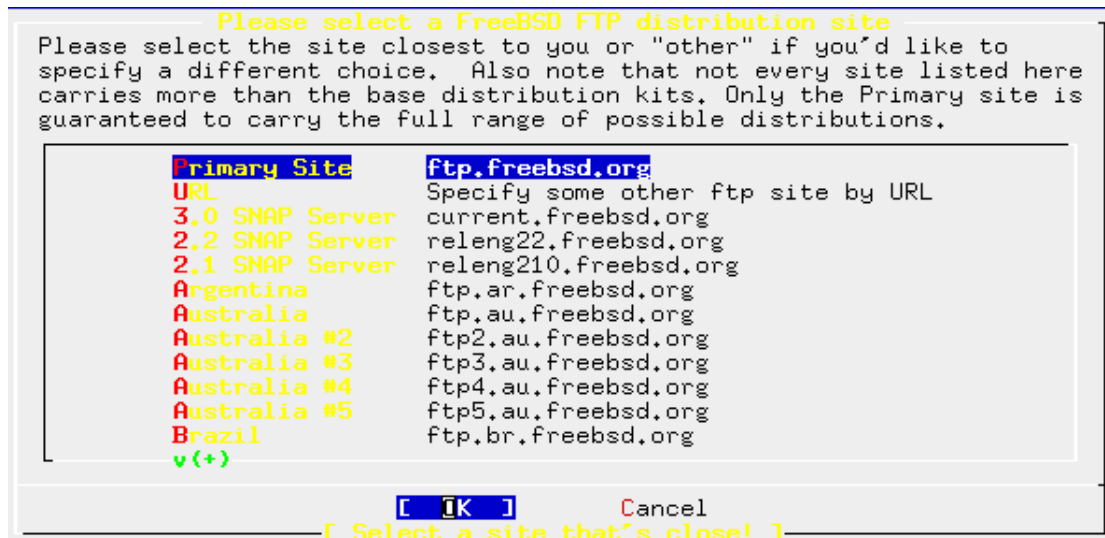
Just stick the CD in to the CD-ROM Drive and sysinstall should find it. As long as your CD-ROM drive is recognized at start up, you should not have problems. Most ATAPI IDE and SCSI CD-ROMs are supported.

The newer installation CD-ROM's for FreeBSD are bootable. If your computer supports booting from a CD-ROM, all you need to do is stick it in the CD-ROM drive and REBOOT the machine.

Pre-FreeBSD-2.2.5 Users:

The IDE/ATAPI CD-ROMs need to be setup as a slave on the Hard drive controller. It will not be recognized properly as a master on the second Hard drive controller This has been fixed in recent FreeBSD releases.

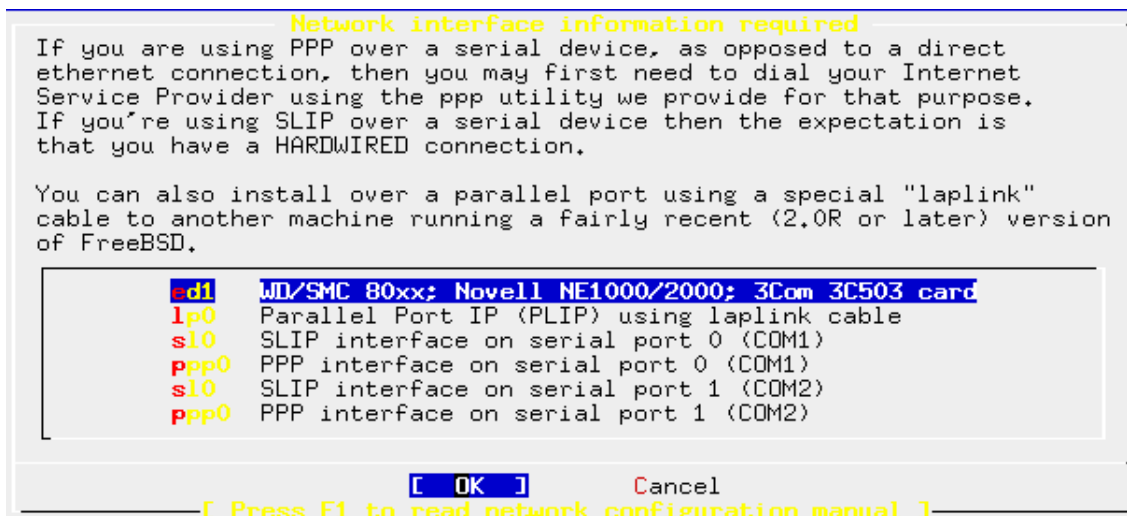
10.5.2. Installing from an FTP Site



Installing from an FTP site is one of the easiest ways to install FreeBSD. It simplifies things greatly, because all you need is a boot disk and an internet connection.

Just Select a site that is close to where you are and press **ENTER**.

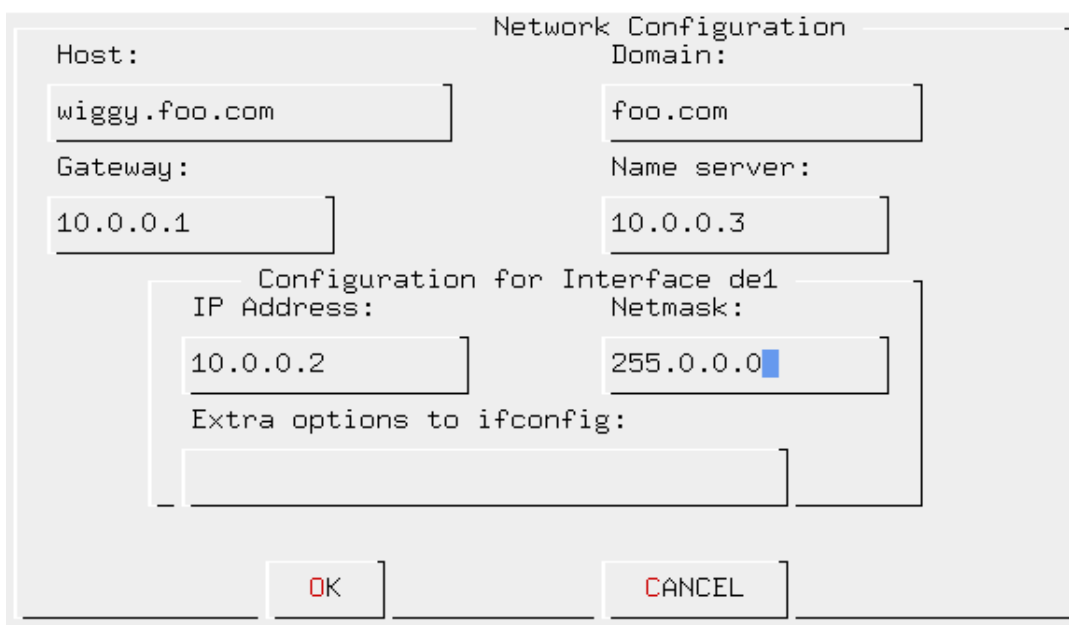
It is possible to install from a local FTP site that you have prepared. The site must include copies of all the distributions you wish to install. This would be helpful if you are planning to install several times from a slow link to the Internet. The distribution could be downloaded ahead of time and installed as many times as needed from local FTP site. See the section on Preparing Distribution Sets for a list of what to download and where to put it.



To install using FTP, your computer must be configured to reach the FTP site. You can do this a variety of ways.

If FreeBSD has detected a device that it could possibly connect, using FTP, protocols to another machine with, it will list them here. This list includes, Network Cards, COM ports, and parallel ports. If you have installed a Network Card, but it is not listed on this list, FreeBSD did not detect it during start up. You may need to check for the device during boot up, and make sure the IRQ and other address settings are correct.

Once you have selected an interface (Network card, modem, etc...), you will need to give your computer an identity on the network.



You have to give your computer a name. In the host field, enter in the name you have given this computer. This field can be anything of your choosing. It should be unique in your domain. In this example, the computer's name is wiggly. You only need to enter the name, the domain name will automatically get added to the host field as soon as you fill in the domain field. The domain name should be given to you by your ISP. A domain name is unique to a particular site or business. Each

computer should have an individual hostname, but a common domain name. Kinda like first and last names in a family.

In the Example, the domain name is `foo.com` If you are configuring this for the Network or Internet, an IP address, gateway address, and nameserver IP address are also necessary. All of these should be provided to you by your ISP (Internet Service Provider).

The Gateway address is an IP address that TCP/IP uses to know how to get out to the internet. It is the address of the router that acts as a `gateway` between you and the internet.

The Name Server address is the IP address of the computer that will translate internet names into IP addresses so that TCP/IP protocols can connect using easy to read names. For example, `freebsd.org` is at IP address `204.216.27.18`. However, that is very hard to remember. The DNS (Domain Name Server) will translate that IP address into something readable to us, and vice-versa.

The IP address is your personal IP address that belongs to the machine you are setting up FreeBSD on. The hostname you just gave it will correspond to this IP address. An entry will need to be setup later in your DNS to make this happen. Do not pick a random IP address. Each IP address has to be unique on the internet. Please get this information from your ISP.

The Netmask is an indication of the relationship between the IP address on this machine and the Gateway IP address. Each `255` that appears in the Netmask means that that field is the same in both Gateway and local IP address. The standard Netmask is `255.255.255.0`. This netmask tells us that the first three sets of numbers on the IP address match with the first three numbers on the Gateway IP address.

```
IP address:    123.123.123.12
Gateway:      123.123.123.1
Netmask:      255.255.255.0
```

This would be a correct scenario. If the numbers differ in the fields, either the netmask would need to be opened up a bit, or a router would need to be installed between the two sets of numbers. A router would create a reachable gateway.

```
IP address:    123.123.111.12
                ^^^(These are Different.)
Gateway:      123.123.123.1
                ^^^(These are Different.)
Netmask:      255.255.255.0
                ^^^(Therefore this mask blocks them.)
```

This would not be a reachable Gateway. If the Netmask was opened up to `255.255.0.0`, then the Gateway would be visible.

You do not need to give any parameters to `Extra options to ifconfig`.

From here you are ready to start installing the actual files. Just select `OK`. If you are in the `Novice` or `Express` installs, it will move you ahead and ask you if you are sure you want to do this. If you are in a custom install, you have to select `Commit` from the install Menu

If you experience any difficulties, copy down any error messages you received during the install process and consult the FAQ and handbook for anything relating to that topic. Then search the mailing list archives. If that reveals nothing, consult the people on the `questions@freebsd.org` mailing list. Be sure to reference the errors you received.

11. Upgrading FreeBSD Versions

Caution:

Just to be safe, backup all of your data before upgrading a system.

```

                                upgrade
Welcome to the 2.1.x (or 2.0.5) -> 2.2 upgrade procedure!

It must first be said that this upgrade DOES NOT take a particularly
sophisticated approach to the upgrade problem, it being more a
question of providing what seemed "good enough" at the time. A truly
polished upgrade that deals properly with the broad spectrum of
installed 2.0.5 / 2.1.x systems would be nice to have, but until that
gets written what you get is this - the brute-force approach!

What this upgrade will attempt to do is best summarized thusly:

1. fsck and mount all file systems chosen in the label editor.
2. Ask for a location to preserve your /etc directory into and do so.
3. Extract all selected distributions on top of your existing system.
4. Copy certain obvious files back from the preserved /etc, leaving the
   rest of the /etc file merge up to the user.
5. Drop user in a shell so that they may perform that merge before
   rebooting into the new system.

And that's it! This "upgrade" is not going to hold your hand in all
                                [ OK ] ( 51%)
```

The first thing an upgrade asks you is where to mount your partitions. It is important to label your partitions the same as they were previously. For example, you have a 100M partition named `sd0s1f`, and a second 100M partition named `sd0s1g`. The partition `sd0s1f` is mounted on `/home`, and `sd0s1g` is mounted on `/var`. If during your upgrade, you switched the disklabels on those two partitions, the upgrade would start copying the files that belong in `/var` on to your users home directories, because they were kept on `sd0s1f`. So, if you delete, or mis-label, a disk partition, you may lose, or screw up, all information stored on that partition.

When you upgrade from a 2.2.1 or earlier -RELEASE, you will be upgrading to the new `rc.conf` files by hand.. You will no longer be using the `/etc/sysconfig` file to configure your server. You will have to convert the information from `/etc/sysconfig` into `/etc/rc.conf`.

FreeBSD Disklabel Editor

Disk: sd0 Partition name: sd0s1 Free: 0 blocks (0MB)

Part	Mount	Size	Newfs	Part	Mount	Size	Newfs
sd0s1a	<none>	62MB	*				
sd0s1b	swap	138MB	SWAP				
sd0s1f	<none>	1846MB	*				

The following commands are valid here (upper or lower case):
C = Create D = Delete M = Mount ptW = Write
N = Newfs Opts T = Newfs Toggle U = Undo Q = Finish
A = Auto Defaults for all!

Use F1 or ? to get more help, arrow keys to select.

Note:

Make sure you know where your volumes are currently mounted before you start an upgrade, so you can remount them in the same place

12. Post install Configuration

FreeBSD Configuration Menu

If you've already installed FreeBSD, you may use this menu to customize it somewhat to suit your particular configuration. Most importantly, you can use the Packages utility to load extra "3rd party" software not provided in the base distributions.

1 User Management	Add user and group information
2 Console	Customize system console behavior
3 Time Zone	Set which time zone you're in
4 Media	Change the installation media type
5 Mouse	Select the type of mouse you have
6 Networking	Configure additional network services
7 Options	View/Set various installation options
8 Packages	Install pre-packaged software for FreeBSD
9 Root Password	Set the system manager's password
A HTML Docs	Go to the HTML documentation menu (post-install)
X XFree86	Configure XFree86
D Distributions	Install additional distribution sets
v (+)	

[OK] Cancel

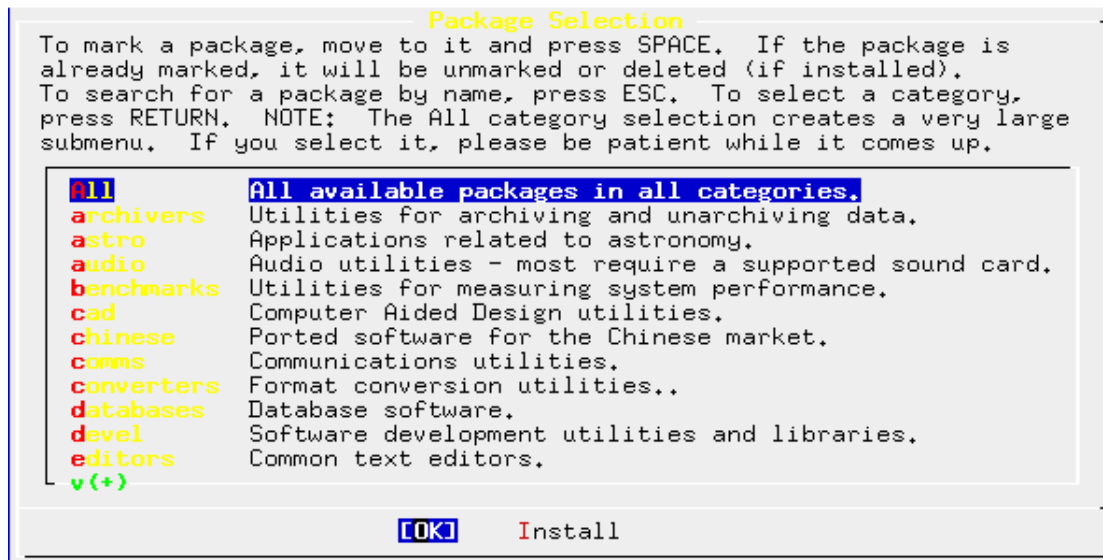
[Press F1 for more information on these options]

During the Post install procedure you get a chance to do a lot of configuration to you system. you can:

- Add Packages
- Set up Users and Groups
- Set up Basic Networking Services

- Install Additional Distributions.
- Configure system wide settings.
- Set the Root Password

13. Adding Packages



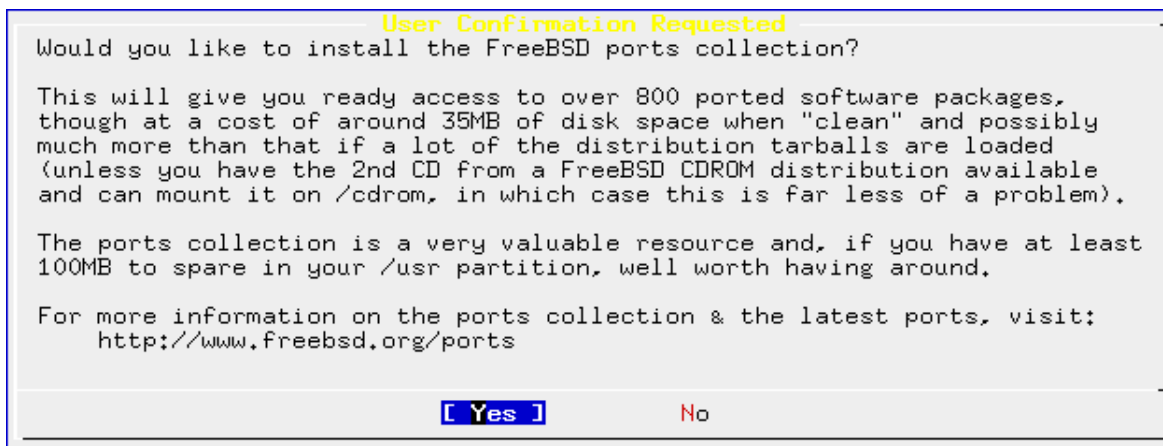
Packages are pre-compiled software that have been assembled for use with FreeBSD. Packages are really just software packages, like you would purchase from a vendor, except they have been packaged by people who work with the FreeBSD project and are available for free. They are the work of people who have either written a software package especially for FreeBSD, or have taken an already existing piece of software and packaged it for installation on FreeBSD. Packages are designed for a specific -RELEASE, in other words, they are compiled on the current version of the operating system and may have unforeseen results if used on an untested version.

Packages are very easy to install. They can be installed several ways.

- From the Sysinstall Packages Menu
- Using the command `pkg_add`
- The hard-way; uncompressing and installing each component by hand.

For more information, see the section on [Adding Software](#)

14. Installing the Ports Collection.



During the install process, it will ask you if you want to install the Ports collection. The Ports collection is a source code distribution of the Package collection. Programs from the Ports collection get compiled for each -RELEASE of FreeBSD and placed in the Package collection. Getting a program from the Ports Collection, instead of the Packages Collection, is kind of like getting an uncooked pizza, it takes more work, but you can pick off the onions if you want.

Part 3:

Running FreeBSD

- 15. *Getting Started With FreeBSD and Unix* *
- 16. *Directories And Files* *
- 17. *System Control* *

15. Getting Started With FreeBSD and Unix

*

- 15.1. *User Names and Passwords -- Logging In*
- 15.2. *The SuperUser - root* *
- 15.3. *The Unix Environment* *
- 15.4. *Changing Passwords* *
- 15.5. *Creating/Adding Users* *
- 15.6. *Unix System Commands*
- 15.7. *Erase and Kill Characters*
- 15.8. *Shutting down FreeBSD* *

15.1. User Names and Passwords -- Logging In

When you boot FreeBSD, you'll see the hardware detection messages scroll by on the screen, followed by some information that indicates what daemons are being started. When everything is said and done, you'll be left with "The Login Screen". It will resemble the following:

```

/dev/rsd1a: clean, 17298 free (106 frags, 2149 blocks, 0.3% fragmentation)
/dev/rsd1s1f: clean, 261451 free (4563 frags, 32111 blocks, 0.5% fragmentation)
/dev/rsd1s1e: clean, 27613 free (85 frags, 3441 blocks, 0.3% fragmentation)
Doing initial network setup: hostname.
ed0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 192.168.0.2 netmask 0xfffff00 broadcast 192.168.0.255
    ether 00:4f:4c:00:70:4f
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    inet 127.0.0.1 netmask 0xff000000
Additional routing options: IP gateway=YES.
recording kernel -c changes
additional daemons: syslogd.
Doing additional network setup: portmap.
Starting final network daemons:.
setting ldconfig path: /usr/lib /usr/lib/compat /usr/X11R6/lib /usr/local/lib
starting standard daemons: inetd cron printer sendmail.
Initial rc.i386 initialization: linux.
rc.i386 configuring syscons: keyrate.
Local package startup: Modula-3 sshd.
starting local daemons:.
Sat Jun 28 21:23:50 EDT 1997

FreeBSD (spokane.quickweb.com) (ttyv0)

login:

```

If you've never used a multi-user operating system before (DOS, Windows, and Macintosh are all single user.) the concept of "logging in" will be new to you. FreeBSD is waiting for a login - a login is simply the name that you supply to FreeBSD to identify yourself to the operating system. FreeBSD keeps track of which names are permitted to log in or access the system, and only allows valid users to have access. When you successfully log in, you'll be presented with an interface to the operating system - the Unix Environment covered later in this chapter. Generally login names reflect a person's real name (for example, on my home system my login name is `mark` on a larger server at the University, my login name is `mmayo` - to distinguish me from all the other Mark's on the system). Login names must be unique on a particular system.

To gain access to the system, you type in your login name. If you are booting FreeBSD for the first time, the only account that is valid is that of the SuperUser: "root". Type `root` at the Login: prompt and press **ENTER**. If you specified a password for root during the installation you'll be asked for it now; otherwise the system will immediately log you in as root. If you are new to Unix, you would be wise indeed to read the section entitled *The SuperUser* presented later in this chapter.

Things to remember:

- FreeBSD login names and passwords are case-sensitive. `root` is different than `root`, `ROOT`, and `ROOT`. Without strong reasons, it is suggested that you use all lowercase for your user names, keeping them to 8 characters.
- The root login does not restrict you in any way. You are god of your OS. With one simple command, you could wipe out everything! Cool huh? For this reason, use the root login only when necessary. Avoid experimenting with commands when you are root.
- System administration is another term you will see often. A system administrator is the actual person who sets up and maintains the FreeBSD (or any other) system. You are the system administrator of your home PC, but your roles (fortunately) are far less consuming than the system administrator of a large public server. System administration generally requires SuperUser privileges.

One of the most useful things about FreeBSD is that you can “log in” from just about anywhere! FreeBSD uses the notions of `ttys` - “terminals” that can attach to a virtual terminal (`ttyv` - such as your screen), a pseudo terminal (`ttyp` - such as a network connections from another Unix or Windows computer), and serial terminals in case you happen to have an old DEC vt100 terminal device lying around :-). Terminals, combined with the concept of logging in make FreeBSD/Unix a dream to administer remotely! Another side-effect of the multi-user `ttys` is that YOU can be logged into your own system as many times as you want. If you’ve just started FreeBSD for the first time, then it may not be obvious too you how to get multiple logins going at the same time. It’s easy, just hit ALT-F2 to switch to “virtual console 2” and you’ll see another login screen. By default, FreeBSD ships with 3 virtual consoles (`ttyv0-2`), reachable with ALT-F1,ALT-F2,ALT-F3. Go ahead and try switching between consoles! Having multiple sessions (logins) going can be extremely useful. For example, say you want to search for the word “foobar” in the entire `/usr/src` source tree! Depending on the speed of your disk and CPU, this could take quite a while - say 2 minutes - so instead of waiting around doing nothing while the command grinds away, you can switch to another console and continue working on whatever you want.

Logging Out!:

Logging Out!

When you’re done doing whatever you were doing it’s time to log out! You can either type `logout` or `exit`. Once you’ve logged out, you’ll see the `Login:` prompt again, indicating your system is ready for more abuse :-)

Note:

The term “account” is synonymous with “login” Some people use the phrase account to encapsulate a user’s entire set of rights and possessions on the system (his login permissions, files, home directory, mail account, etc.), and use the term login to refer to the user’s “account name” At any rate, the two terms can be used interchangeably and their exact meaning varies depending on who you talk to :-). For clarity, we will always append “name” after account or login to indicate we’re only talking about the user’s login name...

15.2. The SuperUser - root *

In FreeBSD (and just about all other UNIX system) the superuser login name is `root`. No mater how large or small the system may be, if you can login as the user `root`, the system is wide open for you to do whatever you please. Obviously, letting anyone log in as `root` is a Bad Thing(tm). Remember, `root` can do *anything*. Only you should know `root`’s password.

To prevent unauthorized access, the `root` login should always have a secure password. By default, in FreeBSD, `root` does not have a password; it’s up to you to change it. The trick is picking a good password. It’s been known for quite some time that people tend to pick passwords that they can easily remember (we’re humans, after all): their birthday, their home town, an item on their desk, and so on. The problem with these passwords is that they are easy to break, either through guessing or by more sophisticated methods of attack.

So picking a “good” password is important if your machine will be connected to a network (a LAN, or the Internet). The best passwords combine a mixture of Upper and Lower case characters,

as well as numbers. A fine example would be: `34DodgesU`, or `Beez1891`. It's even better if there are no "real words" in your password, since one common form of attack is to exhaustively try common combinations of words and numbers against your password entry. Personally, I think a password should be pronounceable, since it minimizes the risk that you might forget it. Try memorizing `s8t4Wx1T43Dc23HiiU2` and you'll see what I mean...

Finally, perhaps the best approach to password privacy is to never write your password down, and to change it every so often. There are system administrators that change their (and root's) password several times per week. Many people recommend a more reasonable time frame of 2 to 3 months.

Also remember that you should only be the SuperUser while you are performing System Administration tasks. For 'normal' activities, it's always a better idea to do them as a mere mortal since mere mortals can't wipe out the entire operating system so easily.

To "become" the SuperUser while you're a normal user, you need the command `su`. `su` stands for "switch user" and you can use it to `su` to any other user on the system. If you just type '`su`' by itself, it is assumed that you want to `su` to root - the SuperUser.

Warning:

In FreeBSD, you *must* be in the `wheel` group to `su` to root.

So go ahead and put yourself in there now. If you have no idea what I'm talking about, just wait until we get to the section "Creating/Adding Users" and it will all be explained.

15.3. The Unix Environment *

15.3.1. *The SHELL*

15.3.2. *The UNIX Philosophy*

15.3.3. *Using the TCSH - a walk through a day on the command-line*

The Unix environment, for the most part, involves two aspects:

- The Shell
- The User Environment

I can't really tell you exactly what the Unix environment is - but I can tell you what most people believe contribute to "it". Really, the Unix environment is just FreeBSD's interface onto itself. Huh? Think of it this way: You need to get things done, and the operating system is willing and ready to perform these tasks. But you need a way to talk to FreeBSD before you can go anywhere. The "Unix Environment" is the collective services, features, and ideologies that represent the interface to the operating system. If you want to work with FreeBSD (or any Unix for that matter), you need to use the Unix environment to get your work done. As we will see, the Unix environment not only "lets" us get our jobs done, but influences how we approach and solve problems. I'll talk about this philosophy of design after a brief introduction to "The Shell" We will end up this section with a hands on walk through of how to use the shell, and the rest of the Unix Environment.

15.3.1. The SHELL

The most visible part of the Unix system -- the part that prompts you for commands and appears to do your bidding is the shell. A shell is a user interface: it talks to the Operating System for you and grants each logged in user system resources (like processor time, disk storage, memory, etc.). The primary purposes of the shell are to provide prompting, command interpretation and execution, and error reporting. You tell the shell what you want done by typing in commands after the prompt. It's very similar to the command interpreter under DOS, but infinitely more powerful and functional. Instead of presenting users with "C:>", most Unix shells present you with "\$" or "%". Naturally, the prompt can be changed - more on that later!

There are many different Unix "Shells" available. By default, most people will be using the "C Shell (csh)" under FreeBSD. You can tell if you are using this shell if your command prompt has an "%" in it. The C Shell is thus called because it is modeled after the C language syntax. For added functionality (including command line editing and an interactive "history" mechanism), you can use "tcsh", which has become quite popular and has very similar syntax to the csh. Also available are the Bourne Shell (sh) and the Bourne Again Shell (bash). bash is the default on Linux systems, whereas tcsh or csh are generally the defaults on BSD systems. Finally, there is also the Korn Shell (ksh) written at Bell Labs - ksh has become a de facto standard on System V based UNIXes. There is a free implementation called "pdksh" (public domain ksh). Many people consider the tcsh the most feature rich, but that doesn't mean it's right for you. The only way to know for sure is to try them all out and see which one turns your crank. tcsh, bash, and pdksh are all available in the FreeBSD Packages Collection.

Note:

Although you spend most of your time in what's known as the "interactive shell", most shells also provide a high-level programming language (scripting language) that is typically used to automate mundane day to day tasks. That being said, Perl is quickly replacing this use of shell languages in many sysadmin circles, and for this reason we will only look at the shell from an interactive stand-point. If you do want to get into shell programming, there are many books available on the subject!

15.3.2. The UNIX Philosophy

Before we get started with how to maneuver around your system with a shell, we need to get several things out of the way first. When the powers that be created Unix, they had a certain idea, or goal, about how it should work. This is loosely referred to as the "Unix philosophy". It can be summed up in the simple phrase: "Small is beautiful". The philosophy has grown from the UNIX community instead of being forced upon it. According to Mike Gancarz, in "The UNIX Philosophy", there are 9 major "tenets" of the Unix philosophy. Several are significant to your understanding of how to make the most out of your FreeBSD system:

1. Small is beautiful
2. Make each program do one thing well
3. Make every program a filter

The idea here is that small programs are simple, and therefore easier to understand. A natural side effect is that a program should only do one thing - if you want more than one thing done, you just connect different individual programs to accomplish your task. That's where filters (also referred to

as pipelines) come into play. Every program is a filter in the sense that it produces an output given an input. The output of one program can be fed into the input of another program, and vice versa. You will see what I mean once we get to some concrete examples in the next section.

15.3.3. Using the TCSH - a walk through a day on the command-line

Personally, I use `tcsh` as my shell, and for this reason we'll focus on this shell in particular as we walk through several examples of how to take full advantage of your shell. `tcsh` is a modified version of the C Shell (`csh`). It's fully backward compatible with `csh`, but it has many new features that make user interaction much easier and more efficient. The biggest areas of improvement are command-line editing, and history navigation. For me, these make life on the command-line much more enjoyable. Everything we'll say about input and output redirection, pipelines, prompts, and job control is identical to the standard `csh`. If you don't have `tcsh` installed on your system (just try typing 'tcsh' to find out) then you can add it from the Package Collection.

Pipes and Filters

`tcsh` provides a mechanism to connect programs together - the filter mentioned above. It is called "the pipe", to represent the directional flow of data "through the pipe". With `tcsh`, the pipe is represented with the "|" character. Before we can use a pipe, we need to know two UNIX commands: `who`, and `wc`. `who` gives a list of each user that is currently logged onto the system, while `wc` counts words. Let's say we want to find out how many users are on the system. Here's how you ask:

```
vinyl % who | wc -l
8
vinyl %
```

Whoah! There's quite a bit going on here. First of all, you should recognize that the "vinyl %" is my interactive prompt (I've made a simple modification to the prompt to let me know what machine I'm on - in this case "vinyl"). It's begging me for commands, and I fed it the command "`who | wc -l`". The answer was 8. Let's take a closer look at what happened. Normally, the command `who` by itself would perform like this:

```
vinyl % who
mark    tty0    Jun 20 18:04    (131.111.238.108)
dlow    tty1    Jun  9 10:53    (204.27.111.42)
chris   tty3    Jun 20 17:46    (208.8.24.42)
gabor   tty4    Jun 19 13:10    (206.222.78.163)
.
.
.
vinyl %
```

Notice that there is one user per line. We want to take this result and do something with it. In this particular case, we want to count the number of lines, which will be the equivalent of counting the number of users currently online! So what we do is "pipe" the result of the `who` command into the `wc` command. `wc` sees the output of the `who` command as its input, and processes it. The final thing to note is that we gave the `wc` command a "command line option"; the `-l` part. "-l" simply tells `wc` to count lines instead of words. The example above demonstrates quite a bit about the UNIX command line. If you're confused, the best way to learn is by picking a few commands and experimenting. One of the most common uses of the pipe on the command line is with the programs "more", or "less". Anytime a program gives an output result that is more than a page long, the best

way to view it one page at a time is to pipe the result into `more`. For example: `last | more`. This lets me see the output of the command `last` (which outputs a list of all the users who have "last" logged onto the system) one page at a time. Quite useful indeed.

Command Completion

Command completion exists in the `tcsh` to make your life easier. With command completion, the shell is able to determine which files you are interested in on it's own. It will become clear what I mean once you see the following example. Remember the the `cd` command "changes directories". Lets say you have the following directories and files in the current working directory:

```
vinyl % ls -F
News/  bin/  code/  games/  mail/  resume.ascii  test/
```

If you want to change directories from the current working directory to the `test` subdirectory, you would enter the command:

```
vinyl % cd test
```

With command completion, you can save on your typing by recognizing that the directory name 'test' is uniquely identified by the its first letter, 't'. So all you need to type is:

```
vinyl % cd t<tab>
```

The `<tab>` is of course the actual tab key found on your keyboard. When you hit the tab key `tcsh` will fill in the rest of the filename for you! This can be a real help with longer file names and makes moving around on the command-line much quicker. Instead of the Tab key, and the standard `csch` uses the ESC key.

Wildcards

Wildcards allow `tcsh` to match more than one file at a time. It supports the three standard file wildcards supported by pretty much all shells:

- * matches any character or any number of characters
- ? matches any single character
- [...] matches any single character contained within the brackets

The * wildcard can be used to do the same thing as command-completion above. If you entered a command like

```
cd t*
```

and only one subdirectory in the current working directory begins with the letter `t`, this command would behave the same as if you had used command-line completion by pressing the Tab key. The * matches any number of characters, so in this case it will match anything behind the leading `t` character, such as 'test' from the example above.

A more common use of wildcards is for working on multiple files however. Often the `ls` is used as follows:

```
vinyl % ls -l *.html
```

This will show you all the files that end with '.html' -- your web pages. The best way to get a feel for wildcards is to just play around with various combinations. Look at the results and figure out how the command behaved. After a very short period of experimentation I'm sure you will have wildcards mastered!

Command History

Coming soon...

Sprucing up TCSH's default behavior

Both `cs`h and `tc`sh read configuration files when they start - `.cshrc` and `.tcshrc` respectively. `tc`sh will also read the `.cshrc` file if no `.tcshrc` file is present. You can customize the behavior of the shell, and set environment variables in the startup files. Remember that you need to use `ls -a` to see files that start with a period!

Skip ahead to the Sample Configuration section for an example of a typical `.tcshrc` and `.cshrc` config file.

15.4. Changing Passwords *

Changing passwords in FreeBSD is quite simple. It is accomplished with the command `passwd`. When you type in the command you get:

```
vinyl % passwd
Changing local password for mark.
Old password:
New password:
Please enter a longer password.
New password:
Please don't use an all-lower case password.
Unusual capitalization, control characters or digits are suggested.
New password:
Retype new password:
passwd: rebuilding the database...
passwd: done
vinyl % █
```

Note that my first two attempts at a new password were not accepted. First, I picked a password that was too small, and then I picked a password that was all lower-case letters. Both passwords are easy to guess or crack, so FreeBSD refused to let me choose such weak passwords. My third attempt was more than 6 characters, and was a mix of upper and lower-case letters - that one worked. After you enter a suitable password you are asked to retype it to make sure you didn't make a mistake. That's it!

If you are root, you can change anyone's password (although you can never learn what someone

else's password is) by typing `passwd fred` (for example). As root, you can also insist on an insecure password by ignoring the warnings. If you're on your home machine, and it's never connected to the Internet, this is probably fine.

The other related command you might want to investigate is `chpass`. It will let you set things for your login like what shell you want to run, your full name, and what your Office and Home phone numbers are!

Finally, if you ever want to make changes to the password file directly, you use the simple command `vipw`. Note that this stands for "vi password" which implies that you can get around in `vi`. (In case you were wondering, `vi` is an excellent text editor that is pretty much the standard on all UNIX Operating Systems - we explain it in full later on in the book.) It's a good idea to see what the password file looks like, and after you're comfortable in `vi` please try this command out. As with all the FreeBSD password utilities, if you do make a change, `vipw` automatically rebuilds the password database for you.

15.5. Creating/Adding Users *

Adding users in FreeBSD is simple, since there is a handy little system command that does all of the work for you: `adduser`. Adding a user to a system involves setting up a few things, and before we step through the `adduser` command, it would be best for you to know exactly what has to change on your FreeBSD system for a user to "exist"

- First of all, FreeBSD must know what users are allowed to login, and what their passwords are so it can validate the login requests. All of this information is stored in the "password file". A plain text version of this file, suitable for human consumption, is located in `/etc/passwd`. Go ahead and take a look at this file (type `cat /etc/passwd`). You should see a bunch of lines that look a lot like:

```
dirk:*:1061:1061:Dirk Jessberger:/home/dirk:/usr/local/bin/tcsh
```

This line contains most of the information needed about the user named "dirk". The line is separated into separate pieces of information by the `:` character. The first part should be obvious - it's the "user name". The next two numbers are the "user id" and the "group id". Every user in FreeBSD belongs to a primary group (and as we will see below, can belong to many secondary groups). The next section is dirk's full name, "Dirk Jessberger". The last two sections say where dirk's "home directory" is (the location on the file system that belongs to him), and what shell FreeBSD should start for him when he logs in.

- Information about what other groups dirk belongs to, as well as the name of his primary group (group pid 1061) is stored in the "group file", which you can view by typing

`cat /etc/group`. You see lines like this:

```
dirk:*:1061:dirk
geeks:*:1023:mark,tburgess,sarahs,jgturner,pwardrop,gabor,karen
```

The format is similar to the password file, and in this case the first section of the line says that the group called "dirk" is known by the "group id" 1061, and the user "dirk" belongs to this group. In other words, the user dirk belongs to his own group: dirk. This is the default - users have their own groups. You'll see in the second line, however, a group called "geeks" with "gid" (group id) 1023, and a whole bunch of people who belongs to the group "geeks"

A little later, when we deal with file permissions, we will see why this type of situation might be desirable.

- Finally, when a user logs in, they need a “home directory”. A home directory is owned by the user, and is “where you put your stuff”. Your home directory also holds your “dot files”, which are config files (such as .cshrc) that tell various programs how to behave. In FreeBSD, user home directories are usually located in /home.

Now that we’ve seen what FreeBSD needs to know in order to host a user, lets take a look at the program `adduser`. Adduser checks the `passwd`, `group`, and `shell` databases (which are built from the text files mentioned above) when adding users to make sure everything is valid. Adduser also creates a `passwd/group` entry for the user, a HOME-directory, dotfiles and sends the new user a welcome message! As you can see, the `adduser` program does pretty much everything for you, and saves a lot of time compared to the old method of manually setting up an account for a new user.

Adduser is an interactive program, meaning that by default you start it up with no arguments (command-line options) and it asks you questions, which you answer, until it has all the info needed to add the user to your system.

```
vinyl# adduser
(c) Copyright 1995 Wolfram Schneider <wosch@cs.tu-berlin.de>
Use option '-silent' if you don't want see all warnings & questions.

Check /etc/shells
Check /etc/master.passwd
User news: illegal shell: '/nonexistent'
User xten: illegal shell: '/nonexistent'
User httpdguuy: illegal shell: '/nonexistent'
User ftp: illegal shell: '/nonexistent'
User majordom: illegal shell: '/nonexistent'
Check /etc/group
Enter your default shell: csh date ksh no sh tcsh [tcsh]:
Your default shell is: tcsh -> /usr/local/bin/tcsh
Enter your default HOME partition: [/home]:
Copy dotfiles from: /usr/share/skel no [/usr/share/skel]:
Send message from file: /etc/adduser.message no
[/etc/adduser.message]:
Use passwords (y/n) [y]:

Ok, let's go.
Don't worry about mistakes, I will give you the chance later to correct any input.
Enter username [a-z0-9]: █
```

15.6. Unix System Commands

Start in Late July.

15.7. Erase and Kill Characters

Start in Late July.

15.8. Shutting down FreeBSD *

FreeBSD is a sophisticated, multi-user, multi-tasking operating system. It’s not DOS. You can’t simply “power off” with the power button on your PC without first telling FreeBSD that you wish

to shutdown. If you want to know why, continue reading. Otherwise, just skip ahead to the next paragraph for the necessary commands. When your FreeBSD system is up and running, it goes through a lot of measures to optimize performance. One of the biggest ways to optimize performance is through the use of “buffers”. When you read or write to the disk, data is moved into and out of buffers. You can think of buffers as buckets. Lets say I have a hose running, and it only reaches so far. Unfortunately, I’m still 5 meters away from my pool, and I need to get water into the pool. Obviously, it’s not very efficient for my to take one glass of water at a time from the hose over to the pool. I’d use a bucket - or more precisely, I’d place the bucket under the hose, and when the bucket fills up I’d dump it into the pool. Get it? The FreeBSD equivalent is: it’s slow to write to the disk, so I want to minimize the trips I have to make to the disk. So when I’m writing a file, I place the information I need into buffers until I have enough to make it worth while to go and visit that slow old hard-disk. Then I can write everything out at once. Since buffers are stored in RAM, if you just cut the power to your computer, there is a chance that some of the information you thought you wrote to the disk never actually made it there! And that would be crappy. As it turns out, FreeBSD (actually, the filesystem FreeBSD uses) “syncs” data from the buffers out to the hard disk every 30 seconds - but even then there is no guarantee that `sync` will move it to the disk right away. If the system is under low load (a measure of how busy the CPU is - usually very low on a workstation, but it could be quite high on a multi-user server) then it’s unlikely that data will be sitting in a buffer for more than a couple of seconds. So, you need to tell FreeBSD to take whatever is sitting buffers and dump it to the disk. Hence, the `shutdown` command. It slices, it dices, it `sync`’s, and it notifies! What more could you ask for?

To shutdown FreeBSD, you use the `shutdown` command! Depending on what arguments you give it, it will behave differently. For example:

`shutdown -h now` Tells FreeBSD to shutdown, *halt* the CPU, and do it now.

`shutdown -r now` Tells FreeBSD to shutdown, and then reboot the system. Do it now.

`reboot` Does the same as `shutdown -r now`.

When you do the shutdown command, a broadcast message will be sent to all users currently logged in that the system is going down. Instead of “now” you could say `shutdown -h +5` which will bring the system down in 5 minutes. Every minute FreeBSD will send a broadcast message to all users warning of the impending shutdown. If you just want to kick everyone off, `shutdown -k now` will do so, and will also prevent anyone else from logging in!

```

root:~102~/home/mark % shutdown -h +3
Shutdown at Sun Jun 29 00:56:51 1997.
shutdown: [pid 236]
root:~103~/home/mark %

*** System shutdown message from mark@spokane.quickweb.com ***
System going down in 3 minutes

root:~103~/home/mark % ps
  PID  TT  STAT      TIME COMMAND
  207  p1  S      0:00.13 -su -m (tcsh)
  228  p1  S+     0:00.14 xterm
  229  p2  Ss     0:00.10 -tcsh (tcsh)
  236  p2  S<     0:00.00 shutdown -h +3
  239  p2  R+     0:00.00 ps
  159  v0  Is+    0:00.01 /usr/libexec/getty Pc ttyv0
  160  v1  Is+    0:00.01 /usr/libexec/getty Pc ttyv1
  161  v2  Is+    0:00.01 /usr/libexec/getty Pc ttyv2
root:~104~/home/mark % █

```

In summary, using the FreeBSD `shutdown` command is not only the “safe” way to stop your system, but also the “friendly way” :-)

16. Directories And Files *

- 16.1. *Reading the Manual Pages* *
- 16.2. *The File System* *
- 16.3. *Viewing Text Files* *
- 16.4. *Moving Files Around* *
- 16.5. *Managing Permissions* *
- 16.6. *Text Editing in System Administration: vi* *
- 16.7. *IN-DEPTH VI.HELP* *

Start in Late July.

16.1. Reading the Manual Pages *

The FreeBSD manual pages (or manpages as they are usually called) describe nearly every user command, administrative command, library call, and file format used on the system, plus some other informative tidbits (like a built-in ASCII table). Each page includes a summary, lists all the options available with the command, and generally goes into detail about the command and refers you to related pages.

Man page information is only available on actual commands, not to aliases you set up in your `.cshrc` file.

To access the Manual Pages, type in `man command` at the Unix Command Prompt with out the quotes. Substitute the word “command” with the command that you are interested in learning about. For Example:

```
/usr/home/chris> man ls
```

The above command will generate a manual page explaining the use of `ls`. While you are viewing

the man page, you can control the screen the same way you would control a `more` session.

Pressing these keys will result in the following:

- "`q`" Will quit the current man session.
- "`b`" Causes Man to go back to the previous screen.
- "`j`" Causes Man to go forward one line.
- "`k`" Causes Man to go back one line.>
- "`<SPACE>`" Causes Man to go forward one page.

Note:

In order to access the online man pages, you must have the `man` distribution installed on your system.

`man` searches all the directories specified in the `MANPATH` environment variable to find the page you requested.

You can adjust the `MANPATH` search string to include custom man page directories so you can add your own man pages. Man Pages are divided up in to Nine sections. The first of these contains all of the user commands.

Almost every install option in the FreeBSD `sysinstall` includes the manual pages. So unless you have selected the custom install and didn't choose the man pages, you should have access to them on your system

16.2. The File System *

When you are new to UNIX, especially when converting from a non-UNIX operating system like DOS or MacOS, figuring out which commands do what you want can be very frustrating. Most of the help features built in to UNIX are designed to help you figure out how to fully use a command AFTER you have discovered it.

One of the first commands that you have to learn is `ls` `ls` is a command to "list" or display all of the files that are present in your current path. In order to comprehend all the things that `ls` can show us, we need to first understand how UNIX treats files.

There four main types of files:

- Regular files
- Directories
- Devices and Special files
- Linked files

Regular files include programs, data files, pictures, binary executables, etc... These account for almost all of the files that will exist on your system.

Directories are files that allow you to contain other files in them. On windows and the Mac systems they are called "folders." All regular files are contained inside a directory. Directories can be contained in side of other directories, these are called sub-directories. Dos users will notice that the directory separator in UNIX is the opposite direction from that in DOS. For more information on directories see the man pages on `pwd` `cd` `mkdir` and `rmdir`

Devices are accessed through special files that are contained in the `/dev` directory. These files allow you to control various parts of the computer, such as your modem or floppy disk. Device drivers access these files to give instructions to things like your mouse or sound card.

Also reference the man pages for: `MAKEDEV`, `mknod`, and

Link files are a lot like pointers in C. Links come in two types: real links and symbolic links. Real links are actually another name for the file. A real link has a few limitations, it can't be linked to a directory and it must exist on the same physical disk and disk partition. However symbolic links, or symlinks as some people call them, don't have this limitation. A link to a file allows you to move the file from it's original place, and still allow programs that depend on it to find it. A Symlink can be thought of as a shortcut to a file. "Shortcuts" in Win95 work exactly the same way. Deleting the symlink doesn't delete the actual file, just the shortcut to it. For more information on links, see the man page for `ln`

Every file has certain properties. These consist of:

X	Kind of File	X	Number of Links		Size	month	day	time	File Name	
v	Read Write	v	Owner	Group						
v	Execute	v								
v	permission	v								
crw-rw-rw-		1	root	wheel	2,	12	Aug	28	1996	zero
brw-r-----		1	root	operator	0,	7	Aug	28	1996	wd0h
brw-r-----		1	root	operator	0,	146	Aug	28	1996	wd0s3
brw-r-----		1	root	operator	0,	682	Aug	28	1996	wd0s4
-r-xr-xr-x		1	bin	bin	45056		Feb	28	04:05	cat
-r-xr-xr-x		1	bin	bin	53248		Feb	28	04:05	chmod
-rw-rw-r--		1	chrisc	bsdbook	416		Mar	24	23:09	.cshrc
-rw-rw-r--		1	chrisc	bsdbook	420		Mar	24	23:09	.login
-rw-rw-r--		1	chrisc	bsdbook	0		Mar	10	13:38	help
drwx-----		2	chrisc	bsdbook	512		Apr	3	01:46	mail
drwxrwxr-x		5	chrisc	bsdbook	512		Mar	24	23:22	public_html
lrwxr-xr-x		1	root	wheel	11		Mar	23	22:56	sys -> usr/src/sys

Files that begin with a "." are hidden files, and are not usually displayed by `ls` and by default they are not acted upon by most other programs.

There are two special files that exist in every directory: `.` and `..` these refer to the "current" and "previous" directories respectively.

16.3. Viewing Text Files *

16.3.1. *CAT:*

16.3.2. *MORE:*

16.3.3. *TAIL:*

16.3.4. HEAD:

Most of the work you do as a system administrator involves text files. Either you are checking a config file, or viewing a log file. To accomplish the job of displaying text files, Unix provides several different tools.

16.3.1. CAT:

`cat` is short for concatenate. If you don't redirect it elsewhere, `cat` will display a file to the standard output, namely the monitor you are looking at. `cat` displays the file as fast as possible, and will not stop for page breaks. For Example:

```
cat myfile
```

will display the contents of "myfile" to the screen. However, if that file happens to be a non-text file, such as a binary file, you would see all of the binary escape codes; it might not be a pretty sight.

If more than one filename is specified, `cat` will display them one after the other, in order, without pausing, as though they were one file. By using the redirection operator `>` you can concatenate two or more files together and create a third file that contains all the information from the other files.

For example:

```
cat file1 file2 file3 > file4
```

In this example, `file4` would contain all the information from files 1,2, and 3. It would not matter if this was a text file or a binary file, because the information would never be shown to the screen.

16.3.2. MORE:

But what if while we are using `cat` we have a file that is too long to fit on one screen, and we really want to read what it says as it scrolls past us at sub-light speed. Over the years people have come up with several programs to make the display pause for page breaks and let us read the page before showing us a new one. One of the first and more popular ways is `more`. Two other commands, `less` and `page` have recently become popular. Currently, `less` is just a hacked version of `more`. Fundamentally `more` works just like `cat` except `more` pauses for page breaks and prompts you to press a key before it continues. However, it isn't always easy to tell which key to press, because it doesn't tell you. It just sits there with a "stdin" in the lower left hand corner. It assumes that we have read the man page, and know that "stdin" stands for "standard input device" ie. the Keyboard. It is waiting for us to press the **SPACE BAR** to go one page forward, the **ENTER** key to go one line forward, **b** to go back a page, or **q** to quit out of it.

By using the pipe operator `|` we can route all the information through `more` and make any program that is displaying text to the screen slow down and pause at page breaks. For example, a `ls -lR /` will display all the files in our computer. This would be very hard to read, and we wouldn't get much out of it. But if we do a `ls -lR / | more` this will show us the directory listings one page at a time and also allow us to stop by pressing **q**

16.3.3. TAIL:

Sometimes, however, we have a real long file and we are only concerned with the last page of

information. It would be silly to use `cat` or `more` and wait all the way through 1500 pages of text just to see the last page. Log files are a prime example of a situation like this. For example, you are checking the log for the web server and you want to see who the last few visitors are, or you want to check the last couple of messages in `/var/log/messages`. In situations like these, it is a good idea to use `tail`.

If we wanted to see the last messages in `/var/log/messages`, type:

```
tail /var/log/messages
```

The last page of the message file will be displayed on your screen.

16.3.4. HEAD:

`head` works just like `tail` except it show only the first page full.

Also reference the `man` pages for: `more` `less` `head` `page` `tail` `cat`

For more information on these commands, see the `man` pages for: `more`, `less`, `head`, `page`, `tail`, and `cat`

16.4. Moving Files Around *

Files, they seem to be everywhere; in FreeBSD this is very true. Except they never seem to be where you want them. There are basically three things that you can do with files: Create them, Use them, and Remove them. The rest of the time is spent moving them around from place to place in your file system. To move files around in a file system, you have to have places to move them to and from. These places are called directories. Directories are really files that contain other files. Files can be placed inside a directory and moved from one directory to another.

Directories can be placed inside each other. A directory contained inside of another directory is called a “sub-directory”.

To move a file or directory, we need to use the `mv`. `mv` requires that you have at least two parameters when moving files. (Some people call these “arguments”.)

The first parameter is the source file, or the name of the file you want to move. The second parameter is the destination, or the place you want to move the file to. If you specify a directory as the destination, it will place that file in the directory specified. If you give it a file name as the destination, it will rename your file. If you move a directory using `mv`, it will rename the directory.

For Example:

```
mv /usr/local/junk.html /usr/local/www/ mv /usr/local/html /usr/local/www mv /usr/local/junk.html
/usr/local/www/index.html mv /usr/local/junk.html /usr/local/www/ mv junk.html junk.htm mv
/usr/local/html/* /usr/local/www/
```

16.5. Managing Permissions *

Security is a big issue in today’s “anyone-can-access-your-site” Internet environment. For the new-to UNIX user and even system administrator, keeping access permissions set properly on files

can be a major chore.

Permissions to a file allow a user to read, write, or execute a particular file based on whether he is the owner of the file, a member of the group that owns the file, or an ordinary user that is trying to use the file. UNIX lets you set the accessibility permissions for each of the three categories. Individual file permissions are set with the command `chmod`. The command `umask` is used to set the default permissions that a file gets when it is created. `chmod` and `umask` use a number and column scheme to represent the particular permissions and the category that they apply to.

There are four columns and four numbers (if you count zero). The columns represent the categories that the permissions apply to and the numbers are the read and write permissions.

The Columns:

column1	column2	column3	column4
Special	Owner	Group	Others

The Numbers:

In Column 1 only:	4	Set User ID On Execution.
	2	Set Group ID On Execution.
	1	Set the Sticky Bit.
	0	Remove all Special options

In Columns 2-4	4	Grant Read Permissions
	2	Grant Write Permissions
	1	Grant Execute Permissions
	0	Remove all permissions from column

If you specify less than four digits when setting permissions it will assume that you are starting from column 4 and work backwards, in other words `chmod 22 file` will set the read permissions of file to “write” for group and others. `chmod 24 file` and `chmod 0024 file` are exactly the same. This however will remove all permissions from the user, a better one to use would be `chmod 644 file` for standard files and `chmod 755 file` for executable files. You can have no more than 4 digits, each corresponding to a column.

You can set both read and write permissions to a column by adding the numbers together. Write(2) + Read(4) = Both Read and Write(6). Therefore to set read and write permissions to “Owner”, “Group”, and “Others”; you would use `chmod 666 file`

The file “bsd.gif” has been set to mode 664 using the command `chmod 664 bsd.gif`

An `ls -l` will display the permissions of all the files in the current directory.

```
>ls -l
total 21
drwxr-xr-x  2 chrisc  bsdbook  512 Mar 21 00:50 articles
-rw-rw-r--  1 chrisc  bsdbook  926 Mar 21 16:01 blueball.gif
-rw-r--r--  1 chrisc  bsdbook 1901 Mar 21 23:50 book.html
-rw-rw-r--  1 chrisc  bsdbook  710 Mar 21 15:57 botbar_raw.gif
-rw-rw-r--  1 chrisc  bsdbook 2088 Mar 21 15:57 bsd.gif
drwxrwxr-x  3 chrisc  bsdbook  512 Mar 22 00:02 cgi-bin
drwxrwxr-x  2 chrisc  bsdbook 2560 Mar 21 15:50 docs
-rw-rw-r--  1 chrisc  bsdbook   0 Mar 22 00:15 file
-rw-rw-r--  1 chrisc  bsdbook 2865 Mar 21 15:47 umask.shtml
^^^^^^^^^^ These are the permissions.
```

The ‘d’ tells us which files are directories. Directories must have execute permissions enabled in

order for a user to change directories to it. A '-' in the first field signifies an ordinary file, in the other fields it signifies a lack of permissions, or a permissions of '0'.

When a file is created the default permissions are set at 666. Umask does just the opposite job of `chmod`. It removes permissions from the default values at creation time based on the number and column scheme. Therefore to have your files set to read and write by "Owner", but read only by "Group" and "Others", you would use a `umask` of 22. The Line

`umask 22`

can be put in your `.login` file and automatically set every time you login.

```
The Default Permissions:      666
Your Umask Values:          22
                             -----
Your New Default Permissions: 644
```

Now every time you create a file it will have the New Default permissions. A `umask` of 66 would give you Default permissions of 600, giving only the owner read and write access to the file. `chmod` can also modify permissions to files using a "first letter" short notation. This style of using `chmod` works exactly the same as the Column and Number Scheme. However, it is easier for new users to remember and use. Consequently it gets a lot of use in everyday tasks, while the Column and Number Scheme gets a lot of use by programs and scripting languages.

```
Permissions Types:
    r      Read
    w      Write
    x      Execute

Affected Area: (column)
    u      User
    g      Group
    o      Others
    a      All

Method Affected:
    +      Add to
    -      Remove from
    =      Set equal to
```

To add execute permissions to all areas of the file "bsd.gif", you would use `chmod a+x bsd.gif`

To remove read permissions from "Others" use `chmod o-r bsd.gif`.

Now you can control the initial permissions of files and modify those permissions later to suit your needs.

16.6. Text Editing in System Administration: vi *

- 16.6.1. *CURSOR MOVEMENT:*
- 16.6.2. *TEXT ENTRY MODE:*
- 16.6.3. *Cut N Paste:*
- 16.6.4. *Special:*
- 16.6.5. *EX MODE or ":" COMMANDS:*
- 16.6.6. *Numbers:*

In FreeBSD there are several text editors available; one of the most powerful and common is `vi`. However, to people who haven't grown up using it, learning `vi` may appear to be the hardest part of System Administration. Used properly, `vi` is one of the best tools for System Administration. When learning `vi` the first thing that must be understood is that `vi` is not a word processor. It is a text editor designed to edit the configuration files that are used by the system; at this job it excels and is (in my opinion) easy to use.

The biggest problem is that `vi` doesn't even pretend to be user-friendly. If you don't know how to use it, there is absolutely nothing to tell you which key to press to accomplish the required task. Generally, all it does is beep at you. To the advanced user, the lack of help menus is a "feature" not a draw-back, because the screen is not cluttered with unused help features.

Because of the lack of help features in `vi`, and the difficulties most people have learning it, other text editors have been created. One of those is `vim`, or "vi improved". `vim` has taken all of the functionality of `vi` and made it easier to use.

A program named `pico` from the University of Washington, is a very easy to use text editor. It provides a lot of help features and opens automatically into "text entry mode". FreeBSD now ships with the `ee` editor as the Default. It is very easy to use, but cluttered with help screens.

Unlike most other editors, `vi` has three modes: Command Mode, Text Entry Mode and EX mode. In Command Mode, each key you press is part of a command. Cursor movement, deleting lines, searching, and such are done in command mode. When you start `vi`, it starts you out in Command Mode and you have to Enter the correct keys to get to Data Entry Mode.

From Data Entry Mode you can enter text just as you would in a normal text editor. There are several commands to start Text Entry Mode, revolving around what you want to happen prior to entering the text. You will remain in Text Entry Mode until you press the `ESC` key, which will return you to Command Mode. To start a new document, type `vi` from the command line followed by the name of the file you wish to create. This will put you into a clean document, provided you have not named an already existing file, and start you off in Command Mode. Now any key entered will be interpreted as a command and `vi` will promptly beep at you if you press the wrong one. At this point press `a` to begin entering text. Now enter in several lines of text; if you don't press return, `vi` will treat this all as one line, with no word wrapping. This is an important feature of `vi` in System Administration. Some configuration files require that certain lines all be entered on one line.

`vi` will also show you all the escape characters contained in the text. This is a very helpful feature if you are working with special file formats. Editors like `pico` will often not display special characters. They will sometimes even remove them.

16.6.1. CURSOR MOVEMENT:

Press the `ESC` key to return to Command mode. Use the `h` and `l` keys to move the cursor left and right. In `vi` all the commands are case sensitive. The following commands will move the cursor in various ways.

`h` One Space Left.

`l` One Space right.

j One Line Down.

k One Line UP.

w Forward/Right One Word.

b Back/Left One Word.

g Goto Bottom of Document.

Many of the other commands can be combined with these motion commands.

16.6.2. TEXT ENTRY MODE:

Now position your Cursor at the end of a word and press **a**. The cursor moved one space forward and is allowing you to append text after that word. Press the **ESC** key and return to Command Mode. Position your cursor at the end of the same word. This time press **i** and type some characters. The letters were inserted in front of the cursor position. The following command will allow you to enter Text Entry Mode in various ways.

a Append after the cursor position.

A Append at the End of the Line.

i Insert before the cursor position.

I Append at the Beginning of the Line.

o Open a new line Below the cursor.

O Open a new line Above the cursor.

16.6.3. Cut N Paste:

y Copies or "Yanks" a line *

d Deletes *

c Changes *

P Pastes Below Cursor

p Pastes Above Cursor

*** Note:**

These need to be mixed with motion commands to make these effective. Pressing the same key affects the current line. (ie. **yy** yank current line)

Position your cursor on a line with text on it and press **dd** This will delete the current line and save

it to a temporary buffer. you can restore it by pressing **p**.

16.6.4. Special:

/ Search

n Find the next match.

. Repeat last Command

u Undo/Redo

The search, repeat, and “find next” commands allow you to do a search-and-replace in a very controlled manner. For example, you have a document that has the word `me` in it several times. You want to change it to the word “you” To do this you would search for “me” by typing `/me`. Your cursor will jump to the nearest word containing `me`. Then type `cwyou` and press the `ESC` key. This will change the word to `you`. Now you are set up to search for and replace all the other “me” words in the document. To find the next occurrence of “me” press `n` to change it to “you” press `.`; if you don’t want to change it, press `n` to go on to the next.

`/me` will match any word containing `me`, words like: `reames`, `mess`, and `mean`. Use `/\ me\` to find only the word `me`.

16.6.5. EX MODE or “:” COMMANDS:

w < filename > Write the file

q Quit `vi`

w! < filename > Over Write an existing file/force write

q! Quit without saving/force quit

r < filename > read in a file

!ls execute the shell command `ls`

123 goto line 123

When you press the `:` key, a `:` will appear in the lower left hand corner of the screen. This is a prompt that will accept a limited number of commands. It will let you execute shell commands, read in files, or quit `vi`. When you are done with `vi`, press `:wq` and you will save your file and quit out of `vi`.

To read in or write to a file, you need to supply a file name. If you supplied a file name when starting `vi`, it will use it by default. Giving a filename at this point would override the one you supplied at startup.

16.6.6. Numbers:

In command mode, if you enter a number, it will repeat the next command that many times. For

example, if you are deleting lines, typing `10dd` will delete 10 lines. Motion commands may be used also; `10dk` will delete 11 lines above and including the one the cursor is on.

16.7. IN-DEPTH VI.HELP *

16.7.1. *Introduction*

16.7.2. *(Part I/a) An explanation of the syntax of vi commands.*

16.7.3. *(Part I/b) Quick help and covers some of the important(in my mind) vi*

16.7.4. *(Part I/c) a listing of all the vi commands and their function.*

16.7.5. *(part II/a) The syntax of how addresses work in ex. commands*

16.7.6. *(part II/b) A quick list of almost all of ex commands and their abbreviations.*

16.7.7. *(Part II/c) A listing of almost all of the standard ex commands and their function.*

16.7.8. *(part III) A quick explanation of regular expressions as they are used in vi/ex.*

In-depth help for learning the vi editor.

16.7.1. Introduction

vi has basically three modes,

- 1) command or normal mode,
- 2) insert or input or append mode
- 3) ex or command-line mode.

vi starts in command mode where you can issue commands, move and/or cut/copy text. The `:` (colon) puts you in ex mode where you can use powerful ex commands like `g`, `s`, `v`, `!` etc. certain commands will put you in insert mode, eg. `i`, `I`, `a`, `A`, `o`, `O`. When in insert mode you can type in text. `R` will put you in a special case of insert mode, that is overwrite mode `^[` (escape key) will put you in command mode.

16.7.2. (Part I/a) An explanation of the syntax of vi commands.

SYNTAX OF VI COMMANDS

most commands in vi have the following form (exceptions are `:m`, `M`, `n`, `N`, `p`, `P`, `u`, `U`)

`[n]` operator

eg.

`5x` delete 5 characters

`3ANO!` will insert `NO!` three times at the end of the line if you hit **ESC** after typing the `!`

`2fa` forward to second `a` on line

`3Tc` backward to after the third `c`

`20j` move cursor down 20 lines

`5H` move cursor 5 lines from top of screen

`5rx` replace next 5 characters with `x`

`5$` move to end of 5th line down (line 1 is current)

`3_` move to first non-blank character 3 lines up (line 1 is current)

editing commands(`c`,`d`,`y`,`>`,`<`,`!`) have the following general form

`[n]` operator `[m]` object

an object in vi would be a character or a word or a line or a sentence etc.

basic operators for editing are

`c` begin a change

`d` begin a delete

`y` begin a yank

if the current line is the object of the operator then the object is the same as the operator: `cc`, `dd`, `yy`.

if both `n` and `m` are specified the effect is `n * m`, that is `n` times `m`.

eg.

```

cw      change word
c2W    change next two WORDS (includes punctuation) (could also be 2cW)
2cc    change two lines
c$     change to end of line
c)     change to end of sentence
5dd    delete next five lines
d5G    delete to line 5
dG     delete to end of file
de     delete to end of word
d^     delete to beginning of line
d30|   delete to column 30 on line
yy     yank current line
y'z    yank to line marked with z
y]]    yank up to next section
yL     yank to bottom of screen
y2fq   yank to second q on line forward

```

commands in vi are not echoed to the screen except the following:

```

/      search forward for pattern
?      search backward for pattern
:      invoke an ex command
!      invoke a unix (OS) command that takes as its input an object in the
      buffer and replaces it with output from the command
      eg. !G!Gsort -fd

```

the above commands are completed by pressing the return key
a buffer in vi would refer to the file as it's kept in memory

```

<, >, and ! can also be combined with a movement command like c, d, and y
>>    indent current line one shiftwidth
5<<    outdent 5 lines starting with current
>>%   indent to matching parentheses
!Gsort -fd  sort from current line to end of file

```

16.7.3. (Part I/b)Quick help and covers some of the important(in my mind) vi

QUICK HELP (commands and command combinations to get you statrted in vi)

MOVEMENT

Character:

```

h      cursor left
j      cursor down
k      cursor up
l      cursor right
space  cursor right

```

Line:

```

[n]G   to line n
0, $   first, last position on line
+, -   first character on next, previous line
^, _   first character on current line (other than tab or space)

```

Screen:

```

^F, ^B scroll forward, backward one screen
^D, ^U scroll forward, backward half screen
^E, ^Y show one more line at bottom, top of screen

```

Marking position:

```

mx     mark current position with x
`x     move cursor to mark x
'x     move cursor to first non-whitespace character containing mark x

```

Miscellaneous movement:

```

fa     forward to character a

```

Fd backward to character d
 tg forward to character before g
 Tw backward to character after w
 w beginning of next word
 W beginning of next WORD (punctuation is part of word)
 b back one word
 B back one WORD (punctuation is part of word)
 e end of word
 E end of WORD (punctuation is part of word)
 z**ENTER** position line with cursor at top of screen
 z. position line with cursor at middle of screen
 z- position line with cursor at bottom of screen
), (next, previous sentence
]], [[next, previous section
 }, { next, previous paragraph
 (the above will depend on your settings for what constitutes a "section"
 and "paragraph")

EDITING TEXT:

Inserting text:

a append after cursor
 A append at end of line (same as \$a)
 i insert before cursor
 I insert at beginning of line (same as _i)
 o open line below cursor
 O open line above cursor
 ^[terminate insert mode (escape)

Deleting and changing text:

cw change word
 C change line starting with current character (same as c\$)
 cc change current line
 2cc change 2 lines
 dH delete to top of screen
 dd delete current line
 dj delete current line and one below it
 d/pat delete to pat(tern)
 d'z delete to character marked with z
 d'z delete to line marked with z
 de delete to end of word
 d% delete to matching bracket
 P put text previously yanked or deleted before/above cursor
 p put text previously yanked or deleted after/below cursor
 4s substitute 4 characters
 S substitute entire line
 u undo last change
 . repeat last change
 ~ reverse case
 ll~ reverse case of next ll characters (on current line)

Copying and moving:

yy copy current line to (unnamed) buffer
 "xyy copy current line to buffer x
 "xp put contents of buffer x below cursor
 ye yank to end of word
 "xyG yank from current line to end of file into buffer x
 y2k yank current line and two above it
 y'x yank to line with mark x
 y`x yank to character with mark x
 3>> indent next three lines one shiftwidth
 >>34G indent current line to line 34 one shiftwidth

Save and exit:

:x quit vi, write file if changes were made
 :w file save copy to file

```

:w >> file    append copy to file
:q!          quit vi, do not save changes
:e!         return to file before changes were made, after last write
:e#         edit alternate file

```

Interacting with OS:

```

:!command    run command
:r file      read in contents of file after cursor
:r !command  read in output of command after cursor
:n,m! command run command on lines in file from n to m and replace with ou
!object command send buffer object to command, replace with output (eg.
           !G sort -fd will sort from the current line to end of file)
ncommand    send n lines to command, replace with output

```

16.7.4. (Part I/c) a listing of all the vi commands and their function.

VI COMMAND KEYS IN ALPHABETICAL ORDER

(^A stands for control-a ^i stands for
-i)

```

a          append after cursor (can be preceded by a number)
A          append at end of line (same as $a) (can be preceded by a number)
^A        unused
b          back up one word (can be preceded by a number)
B          back up one WORD (including punctuation) (can be preceded by a number)
^B        scroll up one screen (can be preceded by a number)
c          begin a change (combine with movement command)
          (can be preceded by a number)
C          change line from cursor to end of line (same as c$)
          (can be preceded by a number)
^C        unused; in insert > end insert mode
d          begin a delete (combine with movement command)
          (can be preceded by a number)
D          delete from cursor to end of line (same as d$)
          (can be preceded by a number)
^D        scroll down a half screen (can be preceded by a number);
          in insert > back up on shiftwidth
e          goto end of word (can be preceded by a number)
E          goto end of WORD (including punctuation) (can be preceded by a number)
^E        show one more line at bottom of screen (can be preceded by a number)
f          forward to next typed character (can be preceded by a number)
F          backward to next typed character (can be preceded by a number)
^F        scroll down one screen (can be preceded by a number)
g          unused
G          goto specified line or end of file (can be preceded by a number)
^G        print file info on status line (bottom of screen)
h          cursor left (can be preceded by a number)
H          goto top of screen (can be preceded by a number)
^H        left cursor; in insert > backspace
i          insert before cursor (can be preceded by a number)
I          insert at beginning of line (same as _i)
          (can be preceded by a number)
^I        unused; in insert > tab key
j          cursor down (can be preceded by a number)
J          join two lines (can be preceded by a number)
^J        down arrow cursor key; in insert > same as enter
k          cursor up (can be preceded by a number)
K          unused
^K        unused
l          cursor right (can be preceded by a number)
L          goto bottom of screen (can be preceded by a number)
^L        redraw screen usually
m          mark current cursor position with character typed (a-z)

```

M goto middle of screen
^M to beginning of next line (can be preceded by a number);
in insert > same as enter key
n repeat last search command
N repeat last search command in reverse direction
^N down arrow cursor key
o open a new line below the cursor (can be preceded by a number)
O open a new line above the cursor (can be preceded by a number)
^O unused
p put yanked or deleted text after or below cursor
P put yanked or deleted text before or above cursor
^P up arrow cursor key
q unused
Q quit vi, invoke ex (not very useful) return with vi
^Q unused (some terminals, stop data flow)
r change character under cursor to character typed
(can be preceded by a number)
R replace characters (overwrite mode) (can be preceded by a number)
^R redraw screen usually
s substitute characters under cursor to ones typed
(can be preceded by a number)
S change entire line (can be preceded by a number)
^S unused (on some terminals, resume data flow)
t forward to before next character typed (can be preceded by a number)
T backward to after next character typed (can be preceded by a number)
^T goto previous tag; in insert > move right one shiftwidth
u undo last change
U restore current line
^U scroll screen up a half screen (can be preceded by a number);
in insert > delete back to start of insert (depends on terminal
settings)
v unused
V unused
^V unused; in insert > quote next character
w forward one word (can be preceded by a number)
W forward one WORD (including punctuation) (can be preceded by a number)
^W unused; in insert > back up to beginning of word (depends on
terminal settings)
x delete character under cursor (can be preceded by a number)
X delete back one character (can be preceded by a number)
^X unused
y begin a yank (combine with movement command)
(can be preceded by a number)
Y yank current line (same as yy) (map it to y\$ to be consistent with
D and C) (can be preceded by a number)
^Y show one more line at top of screen (can be preceded by a number)
zENTER reposition line with cursor to top of screen
z. reposition line with cursor to middle of screen
z- reposition line with cursor to bottom of screen
(the three z commands above can be preceded by a number)
ZZ quit vi, write file if changes were made
^Z suspend vi on systems that have job control (depends on terminal
settings)

^@ unused
^[terminate insert mode (escape key)
^\
I don't think it does anything
^] goto tag under cursor
^^ edit alternate file
^_
unused?
^?
unused? delete key?

space cursor right (can be preceded by a number)
! filter program through external filter, requires an object to work c
(combine with movement command) (press enter to execute)

```

"      use register for next delete, yank or put, registers are (a-zA-Z0-9)
#      unused?
$      goto end of line (can be preceded by a number)
%      goto matching bracket ( ) [ ] { }
&      repeat last substitute
'      to first non-blank character on line with mark (a-z)
(      beginning of next sentence
)      beginning of current sentence (sentence is delimited by ., !, ? and
      followed by at least one space)
*      unused
+      down one line to first non-blank character (j_)
      (can be preceded by a number)
,      repeat f, F, t, T commands in reverse direction
-      up one line to first non-blank character (k_)
      (can be preceded by a number)
.      repeat previous command
/      search forward for text entered (repeat previous search if no
      argument supplied) (press enter to execute)
0      goto beginning of line
:      enter an ex command (press enter to execute)
;      repeat f, F, t, T commands
<      begin a shift left (combine with movement command)
      (can be preceded by a number)
=      unused (unless in lisp mode in which case it formats to standard
      lisp formatting)
>      begin a shift right (combine with movement command)
      (can be preceded by a number)
?      search backward for text entered (repeat previous search backward
      if no argument supplied) (press enter to execute)
@      execute a register (0-9a-z".)
[[     beginning of current section (as delimited by the sect= option)
\      unused
]]     beginning of next section (as delimited by the sect= option)
^      goto first non-blank character on line
_      goto first non-blank character on line (can be preceded by a number)
`      goto mark (a-z)
{      beginning of current paragraph (as delimited by a blank line or the
      p= option)
n|     column n of current line
}      beginning of next paragraph (as delimited by a blank line or the
      p= option)
~      change case of character under cursor (can be preceded by a number)

```

16.7.5. (part II/a) The syntax of how addresses work in ex. commands

```

EX EDITOR
address symbols
0      beginning of file
$      end of file
1,$    all lines in file
%      stands for filename, hence it means the entire file (1,$)
#      stands for alternate file
x,y    lines x through y
x;y    lines x through y, current line reset to x
.      current line
n      absolute line number n
x-n    n lines before x
x+n    n lines after x
+[n]   n lines ahead (default is one)
-[n]   n lines back (default is one)
'x     line marked with x
''     previous mark
/pattern/ forward to line matching pattern

```

?pattern? backward to line matching pattern

16.7.6. (part II/b) A quick list of almost all of ex commands and their abbreviations.

QUICK LIST OF EX COMMANDS AND THEIR ABBREVIATIONS					
abbreviate	ab	move	m	tag	ta
append	a	next	n	unabbreviate	una
args	ar	number	# nu	undo	u
change	c	open	o	unmap	unm
chdir	cd chd	preserve	pre	version	ve
copy	t co	print	p	visual	vi
delete	d	put	pu	write	w
edit	e	quit	q	xit	x
file	f	read	r	yank	ya
global	g v	recover	rec	(window)	z
insert	i	rewind	rew	(escape to OS)	!
join	j	set	se	(lshift)	<
list	l	shell	sh	(rshift)	>
map		source	so	(line number)	=
mark	k ma	substitute	s & ~	(execute buffer)	* @

16.7.7. (Part II/c) A listing of almost all of the standard ex commands and their function.

EX COMMANDS

(anything in [] is optional , anything in {} is the rest of the name of the command which is also optional)
the default for address is the current line, except for g, g!, v, w; for these the default is the entire file

ab{breviate} [string text]

define string when typed to be translated into text.
if string and text not specified - list all abbreviations
abbreviations are usually put into your .exerc file
eg. :ab Xvi vi is the best editor
:ab character character
:ab the the
:ab have have

[address]a{ppend} [!]
text

append text at address. ! switches autoindent.

ar{gs}
print filename arguments.

cd [path]
without path change to home directory, with path change to path

[address]c{hange} [!]
text

replace lines with text. ! switches autoindent.

[address]co{py}destination
copy lines from address to destination.
eg. :1,10co50 copy lines 1 to to 10 to below line 50

[address]d{elete} [buffer][count]

delete lines in address if buffer is specified save lines to buffer.
count specifies the number of lines to delete starting with address.
eg. `:/include/,/main/d` delete lines between include and main
including include and main
`:/include+,/main/-d` as above but not including include nor main
`:3d` delete line 3
`:d3` delete 3 lines starting with current
`:$d` a delete to end of file from current line into buffer a
`:d a3` delete next three lines starting with current into buffer a

`e{dit}[!] [+n] [file]`
begin editing file. ! will discard changes to current file.
n specifies line to begin editing file (default 1).
eg. `:e file` edit file
`:e #` edit previous file

`f{ile} [filename]`
change name of file to filename. without argument print current
filename.
eg. `:f %.new` appends .new to current filename, renaming file to
file.new

`[address]g{lobal}[!]/pattern/[commands]`
execute commands on lines that contain pattern, if address is specified
within the address range. ! execute on lines not containing pattern.
without commands print matching lines.
eg. `:g/\#include/d` delete all lines that have include directives
`:g!/\#define/p` print lines that are not define statements
`:g/^\/*/p` print lines that start with /*
`:g/^[^I]*$/d` delete empty lines as well as lines with only tabs
or spaces
`:g/strcmp/d5` delete lines that have strcmp in them as well
as the following 4 lines

`[address]i{nsert}[!]`
text
.
insert text at line before address. ! switches autoindent.

`[address]j{oin}[!][count]`
join lines in specified range. ! preserves white space.
eg. `:1,5j!` join lines 1 to 5, preserve white space

`[address]k char`
synonym for mark, character can follow k with no intervening space.

`[address]l{ist}[count]`
print lines so that tabs show as ^I and end of lines are marked with \$.

`map[!] [char commands]`
define a keyboard macro for named char that is a synonym for commands.
! will create a mapping for input mode. with no arguments print mapped
keys.
eg. `:map ^N :n^M` (to get a control character type ^V followed by the
control character)

`[address]ma{rk} char`
mark lines with char. char is a single lowercase letter.
eg. `:ma z` mark line with z
`:'z` return to line with mark z

`[address]m{ove} destination`
move lines specified by address to destination.
eg. `./include/m /string/` move lines from current to include line below
line with string

```

n{ext}[!] [[+commands] filelist]
    edit next file from argument list. if filelist is provided, replace
    argument list with filelist. ! will discard changes to current file.

[address]nu{umber}[count]
    print lines specified by address, precede each line by its line number.
    count specifies the number of lines to print.

[address]o{pen}[/pattern/]
    enter vi's open mode with lines specified by address, or at lines
    containing pattern.

pre{serve}
    save current buffer as if the system had crashed

[address]p{rint}[count]
    print lines specified by address. count is the number of lines to print.
    eg. :304;+5p    print five lines starting with 100, reset current line t
                100

[address]pu{t} [buffer]
    restore lines that were previously deleted or yanked and put them after
    current line. put line from buffer if specified.

q{uit}[!]
    quit. ! discard changes to file.

[address]r{ead} file
    copy text from file on line below specified by address.
    eg. :0r data    read in file "data" at top of file

[address]r{ead} !command
    read the output of command into file after line specified by address.
    eg. :$r !ls -aFC    run "ls" and read in its output at end of file

rec{over} [file]
    recover file from system save area.

rew{ind}[!]
    rewind argument list to first argument. ! discards changes to current
    file.

se{t} parm1 parm2
    set value to an option with each parm. if no parm is supplied print all
    changed options. for boolean options parm is phrased as option or
    nooption, other options are option=value. all will print all available
    options.
    set commands are usually put into your .exerc
    eg. :se autowrite tabstop=4 autoindent shiftwidth=4 wrapmargin=5
        :se all    print all available options

sh{ell}
    create a new shell. resume editing when shell exits.

so{urce} file
    read and execute ex commands from file
    eg. so ~/old.exrc

[address]s{ubstitute}[/pattern/replacement/][options][count]
    replace pattern with replacement. if pattern and replacement are omitted
    repeat last substitution. count specifies the number of lines on which t
    substitute starting with address.
    options
    c    prompt for confirmation
    g    substitute all instances on line

```

p print last line on which substitution was made
eg. :%s/[hH]ello/Hi/g replace hello or Hello with hi, all occurrences
:.,\$s/[uU][nN][iI][xX]/\U&/10 upcase unix on next 10 lines
:%s/\<./\u&/g turn the first letter of all words to uppercase
:1,/main/s/int/long/g substitute occurrences of int before main with
long

[address]t destination
synonym for copy

ta{g} tag
switch to file containing tag.
eg. :!ctags *.c run ctags on all .c files in directory
:ta func switch to file containing func, put cursor on it

una{abbreviate} word
remove word from list of abbreviations.

u{ndo}
undo changes made by last editing command.

unm{ap}{!} char
remove char from keyboard macros. ! remove macros for input mode.

[address]v/pattern/[commands]
synonym for global!
eg. :v/././.-j join empty lines to have only single empty line
between lines of text

ve{rsion}
version of editor.

[address]vi [type][count]
enter visual mode at line specified by address. exit with Q. count
specifies initial window size.
- place line at bottom of window
. place line in center of window
^ print previous window

vi [+n] file
begin editing file in visual mode at line n.

[address]w{rite}{!} [[>>] file]
write lines specified by address to file. >> is used to append to file.
with no address write all of the file. ! forces the write.
eg. :1,25w new_file write lines 1 to 25 to "new_file"
:50,\$w >> new_file append line 50 to end of file to new_file

[address]w !command
write lines specified by address to command.

wq{!}
write lines specified by address to file and quit. ! forces the write

x{it}
exit file. save changes.

[address]ya{nk} [buffer][count]
place lines specified by address in buffer char. count is the number of
lines to yank starting with address.
eg. :20,100ya z yank lines 20 to 100 into buffer z

[address]z[type][count]
print a window of text. count is the number of lines to display starting
with address.

```

type
+ place line at top of window(default)
- place line at bottom of window
. place line in center of window
^ print previous window
= place line in center of window. leave line as current line.

[address]![command]
execute command. if address is specified apply lines from address as in
to command, and replace lines with output.
eg. :!ls -aFC          run ls, will not read output into file
    :0!ls -aFC         run ls, read in its output to beginning of file
    :11,35!sort -fd    sort lines from 11 to 35
    :%!spell -b        run the spellchecker on entire file

[address]=
print line number of matching address. with no address print line number
of last line.

[address]<[count]
[address]>[count]
shift lines left(<) or right(>). count specifies the number of lines to
shift starting with address.
eg. :1,9> indent lines 1 through 9 one shiftwidth

address
print line specified by address.

return
print next line.

[address]&[options][count]
repeat previous substitution. count specifies the number of lines to
substitute on starting with address.
eg. :s/msdos/UNIX      substitute msdos with UNIX
    :g/OS/&             redo substitutions on all lines with "OS"

[address]~[count]
replace previous regular expression with the previous replacement from
substitute.

* [buffer]
@ [buffer]
execute named buffer

[address]#[count]
synonym for number

```

The following can be used as addresses in ex, as well as in commands :g,
:s, :v, /, ?

16.7.8. (part III) A quick explanation of regular expressions as they are used in vi/ex.

```

REGULAR EXPRESSION SEARCH AND SUBSTITUTE
. match any single character except newline
* match any number (including none) of the single character immediate]
preceding
^ match beginning of line if at start of expression
$ match end of line if at end of expression
[ ] match anything enclosed in [ ]
[^ ] match anything not enclosed in [ ]
\( \) store pattern for later replay

```

```

\<      match following characters at beginning of word
\>    match preceding characters at end of word
\<     escape character following (needed for eg. \<. to match a .)
\

```

Examples

```

dig      matches dig
^dig     matches dig at beginning of line
dig$    matches dig at end of line
^dig$   matches dig as the only word on line
^$      matches the empty line
^..*$   matches a line with at least one character
.*      matches any string of characters including none
^[ ^I]*$ as above but line can also contain spaces and/or tabs(^I)
[dD]ig  matches dig or Dig
[aA][nN] matches an, aN, An, AN
d[aeiou]g second letter is a vowel
d[^aeiou]g second letter is not a vowel
d.g     second letter is anything
^....$  matches a line with exactly four characters
^\.     matches any line beginning with a dot
^\. [a-z] same with a lowercase letter following
^[^\.] matches any line that does not begin with a dot
;$      matches a line ending with a semicolon
digs*   matches dig, digs, digss, digsss etc.
[a-z][a-z]* matches one or more lowercase letters
[a-zA-Z] matches any character
[^0-9a-zA-Z] matches any symbol (not a letter or number)
\

```

17. System Control *

17.1. *Process Control **

17.2. *Accessing The Floppy Drive **

17.1. Process Control *

Whenever you execute a command at the UNIX prompt, FreeBSD assigns it a Process Identification number or PID. This is a number between 1 and 65535. This number will be used to

identify and control the process the entire time that it is running. When the program terminates, the Number will be recycled into the stack and reassigned when the kernel has cycled completely through all 65535 numbers.

If the same program is run again, it will be given a different PID and FreeBSD will track it through this new number. If two copies of the same program are started at the same time, two individual PID numbers will be assigned; one to each.

`ps` will give you a listing of all the process you have running.

```
>ps
PID   TT    STAT   TIME       COMMAND
12346 p0    Is+    0:00.57    -tcsh (tcsh)
9871  p0    R+     0:00.57    ps
```

`ps -a` will give you a listing of all of the process running on the system.

To stop a process, you can use the `kill` command. Select the PID of the process you want to stop. Then type

```
kill -KILL PID.
```

For Example to kill process #12346 you would type:

```
>kill -KILL 12346
```

Process # -1 refers to every running process that you own. If you have several process running, you can kill all of them, (except your current shell) by typing:

```
kill -HUP -1
```

As root, you can use this command to kill, or restart, every process in the system.

If you don't want to use the PID, you can use the program `killall`. `killall` requires that you specify the name of the program running as a command line argument. Then it kills every instance of that program that it finds, providing you have permission to kill that program.

For instance:

```
killall make
```

would kill all `make` processes that you have running.

In addition to a priority, each process is assigned a "Niceness" value. The default value is 0. The users can give a `nice` value of up to +20, making processes run slower; freeing up processor time for more important processes. The "nicer" a program is, the more it allows other programs to run. Root can give values down to -20, making important processes run faster. All child processes, processes started by a program, inherit the `nice` value from the parent program. If you give a `nice` value of -20 to a `make world` all processes spawned by that program will have the `nice` value of -20. ie. all compiler processes generated will run at a `nice` of -20.

To give a program a `nice` value, precede your command with `nice`. If you wanted to start `ppp` with

a nice value of +10 you would use:

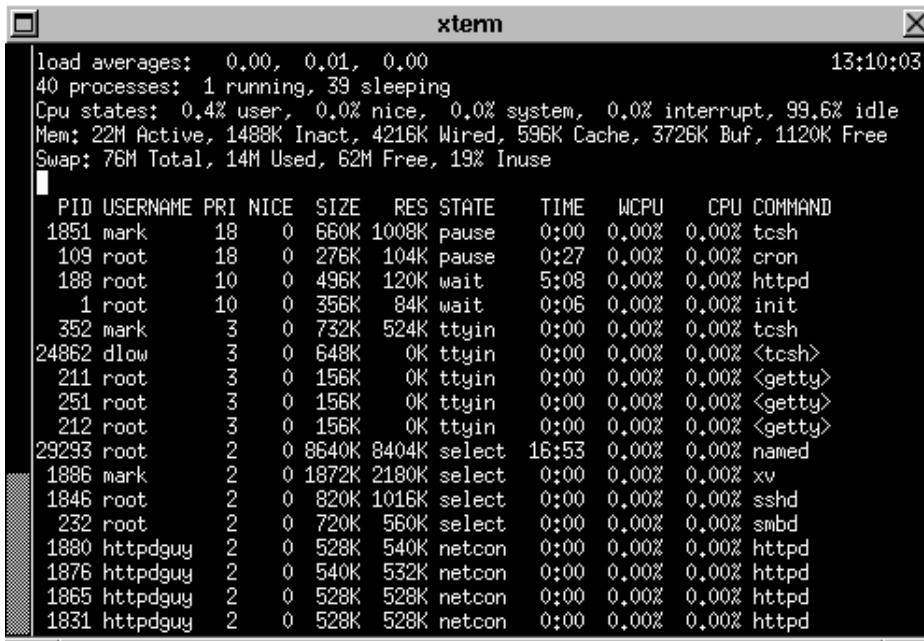
```
> nice +10 ppp
```

If you want to change the nice value of a process that is already running, you would use `renice` followed by the process ID. To change the niceness value of process #1234 from the default of Zero to -10, you would use:

```
> renice -10 1234
```

'Nicing' or 'renicing' to +10 is the most common use of the (re)nice command. It's really just a way of telling FreeBSD that you aren't too concerned about the process finishing immediately, and it's okay to let other processes use more CPU time than your process. Usually, if you are going to run a program that you know will take a long time to run (say 5 minutes or greater) it is considered 'polite' to `nice +10` the program. Your program will still finish very quickly. In fact, if the system isn't very busy there will be almost no difference, since no other processes are battling for the CPU!. Unless the system is very overloaded, a nice value of +10 will NOT result in a slow down. When you raise the niceness value of a running program, your program isn't put on hold until others are done, it just receives a smaller percentage of the CPU time. A nice value of +20 indicates that the application in question should ONLY run when the CPU is completely free. Generally, the only time a process is run with a nice value of +20 is when you don't want it to slow down any other aspect of the system.

`top` is a process monitoring tool that helps you keep track of system resources. It displays the Active Processes, CPU Utilization, Swap Space Usage, and Memory Available. It also allows you to `renice` and `kill` processes. If you watch the CPU utilization, you will notice that it sits at about 90% IDLE most of the time. (Active Servers won't have so many Free Cycles.) Many system administrators find ways to utilize these idle CPU Cycles with creative applications of `nice`, `renice`, and `top`. Processor intensive, time consuming processes can be set to run in the background and only use up IDLE CPU Cycles, allowing your program to run and not infringe on regular operations.



```
xterm
load averages:  0.00,  0.01,  0.00                               13:10:03
40 processes:  1 running, 39 sleeping
Cpu states:  0.4% user,  0.0% nice,  0.0% system,  0.0% interrupt, 99.6% idle
Mem: 22M Active, 1488K Inact, 4216K Wired, 596K Cache, 3726K Buf, 1120K Free
Swap: 76M Total, 14M Used, 62M Free, 19% Inuse

  PID USERNAME PRI NICE  SIZE  RES STATE   TIME  WCPU   CPU COMMAND
  1851 mark      18  0  660K 1008K pause   0:00  0.00%  0.00% tcsh
  109 root       18  0  276K  104K pause   0:27  0.00%  0.00% cron
  188 root       10  0  496K  120K wait    5:08  0.00%  0.00% httpd
    1 root       10  0  356K   84K wait    0:06  0.00%  0.00% init
  352 mark       3  0  732K  524K ttyin  0:00  0.00%  0.00% tcsh
24862 dlow       3  0  648K   0K ttyin  0:00  0.00%  0.00% <tcsh>
  211 root       3  0  156K   0K ttyin  0:00  0.00%  0.00% <getty>
  251 root       3  0  156K   0K ttyin  0:00  0.00%  0.00% <getty>
  212 root       3  0  156K   0K ttyin  0:00  0.00%  0.00% <getty>
29293 root       2  0  8640K 8404K select 16:53  0.00%  0.00% named
1886 mark       2  0  1872K 2180K select  0:00  0.00%  0.00% xv
 1846 root       2  0  820K 1016K select  0:00  0.00%  0.00% sshd
  232 root       2  0  720K  560K select  0:00  0.00%  0.00% smbd
 1880 httpdguy   2  0  528K  540K netcon  0:00  0.00%  0.00% httpd
 1876 httpdguy  2  0  540K  532K netcon  0:00  0.00%  0.00% httpd
 1865 httpdguy  2  0  528K  528K netcon  0:00  0.00%  0.00% httpd
 1831 httpdguy  2  0  528K  528K netcon  0:00  0.00%  0.00% httpd
```

17.2. Accessing The Floppy Drive *

FreeBSD looks at file systems in “slices” or partitions. Each file system is assigned to a device. The Floppy Drive is assigned the device “fd”. The devices are also numbered, starting with 0. The first floppy disk is fd0. Filesystem devices are further divided up into slices. Each slice is assigned a letter. The letter is appended to the device name and number. Traditionally, slices a through h are available. Some slices have special meanings:

- a) This refers to the boot sector of the disk only.
- b) This refers to the swap space on the disk if there is any.
- c) This references the Whole Disk. Usually what you want.
- e-h) This and other such letters would refer to non-bootsector slices if the disk happened to be divided up further.

Access to devices is granted through file representations of those devices contained in the `/dev/` directory. Each slice has its own file representation. Therefore to access the whole floppy disk you would reference the file: `/dev/fd0c`. To actually get at the contents of the disk, you have to mount it into the file system under a directory. If you have a FreeBSD formatted Floppy disk, you can just use the `mount` command without any special parameters. You will have to specify a directory that exists on the system that you have write permission to, and have permission to access the device file you are trying to mount. By default, general users don't have access to the floppy disk device files.

To mount the floppy to a directory called `/mnt` you would type:

```
mount /dev/fd0c /mnt
```

Then `cd` to `/mnt` and work with the files that are there.

If you have a MSDOS disk that you want access to, you will use the command:

```
mount_msdos /dev/fd0c /mnt
```

To check and see if it mounted you can `cd` to the directory and do an `ls`, or type `df` and `df` will give you a list of all the file systems that are currently mounted and where they are mounted to. It will also show all the amount of space used.

Mounting a MSDOS floppy would give you access to the floppy just like a regular directory on the system, however it will automatically truncate the file names of any files copied to that directory to the dos 8.3 notation. (ie. `FreeBSD.File.long.name` will become `FreeBSD.Fil`)

When you are done with the floppy disk, you will have to unmount it from the file system. To do this type:

```
umount /dev/fd0c
```

OR

```
umount /mnt
```

Either one will successfully unmount the file system. You cannot unmount a file system while you

are in the directory that the filesystem is mounted to. This will give you a device busy error. You will have to `cd` out of that directory before unmounting it.

Now when you `cd` to `/mnt` it should be empty.

Part 4:

Customizing your FreeBSD System

18. *Configuring your .cshrc File*
19. *Setting up More Virtual Terminals **
20. *Adding a Hard Disk*
21. *System Configuration File Options*
22. *Adding and Installing Software*

18. Configuring your .cshrc File

This is the Standard .cshrc files that comes with your system.

```
#csh .cshrc file

alias h          history 25
alias j          jobs -l
alias la         ls -a
alias lf         ls -FA
alias ll         ls -lA
alias su         su -m

setenv EDITOR   vi
setenv EXINIT   'set autoindent'
setenv PAGER    more

set path = (~ /bin /usr/{bin,games} /usr/local/bin /usr/X11R6/bin)

if ($?prompt) then
    # An interactive shell -- set some stuff up
    set filec
    set history = 1000
    set ignoreeof
    set mail = (/var/mail/$USER)
    set mch = `hostname -s`
    set prompt = "${mch:q}: {\!} "
    umask 2
endif
```

The `path` statement tells your shell where to look for programs when you try to execute them. It starts with the first directory listed and looks there for any match to the command you have just entered. If it finds it, it executes that program.

If the directory is not listed in the `path` statement, it will not look there for executable files. Not even in your current directory. So even if you are in the same directory as the file you want to execute, it will not find it if that directory is not listed in your `path` statement.

If you want to execute a command in your current path, you have to precede the command with

“.”. For example, you are in the `/usr/local/www/cgi-bin` directory and want to test out a program you have written. The program is `getdata.pl`, and you have just tried to run it by typing `getdata.pl`, however instead of the HTML output you expect, you just get the error, `getdata.pl: Command not found`. This is because you did not include the “.” directory in your `path` statement. Remember that in UNIX, the “.” represents the current directory. To get this file to execute you would need to type:

```
./getdata.pl
```

Now you should see the HTML output you expect, unless you messed up in your program and then you will get error messages pertaining to the language you wrote the program in.

A TCSH Example

As we mentioned earlier, the `tcsh` has a few extra features over the base `csch` that make life on the command line a little easier. Here is an example `.cshrc` file that takes advantage of `tcsh`'s goodies:

```
#tcsh .tcshrc file

alias h      history 25
alias j      jobs -l
alias la     ls -a
alias lf     ls -FA
alias ll     ls -lF
#alias su    su -m -- bad for su'ing to root..

setenv EDITOR vim
setenv PAGER less

set path = ( ~/bin /usr/local/bin /bin /usr/{bin,games} /usr/sbin /sbin /usr

setenv IRCNICK ringzero
setenv IRCSERVER irc2.magic.ca
setenv NNTPSERVER news.sentex.ca

if ($?prompt) then
    # An interactive shell -- set some stuff up
    set filec
    set ignore = .o           # ignore object files for filec
    set showmatch             # for programming
    set history = 100
    set savehist = 75         # tcsh version of history..
    set ignoreeof
    set mail = (/var/mail/$USER)
    set prompt="vinyl:{%h}%~ %% " # tcsh version of above!

    set watch = (1 gabor any mike any adrian any pwardrop any)

    umask 2
endif
```

The first things to note are the `setenv` lines. In this case, they are setting some environment variable that are used by my news reader software and IRC client software. The next point of interest is the `set ignore = .o`, which makes the file completion (`set filec`) ignore files that end in “.o”. This is nice if you’re a C programmer since you never would want to edit or manually do much with a .o file - you want the “.c” file! The `showmatch` option is another addition to the file completion mechanism of `tcsh`. Try it out and see what it does for yourself! The `set prompt` option sets up a very friendly prompt for your shell. It will show you what directory you are in all the time. Here’s what it looks like:

```
mark:{123}~ % cd /usr/src
mark:{124}/usr/src %
```

Note that the tilde character (“~”) indicates your home directory. Now that you’ve seen some of what the `tcsh` and `csh` options are, the best way to learn more about them is to `man tcsh`! All of the features of both shells (such as the “watch” feature of `tcsh`) are documented in the man pages. Go ahead and take a look and experiment to see what fun tricks you prefer!

Note:

If you change your `.cshrc` file, you can re-read the changes by issuing a **source .cshrc!**

19. Setting up More Virtual Terminals *

A Virtual Terminal is a Terminal built into the Main Server Console. It allows you to have several screens open at once, however only one screen is visible at a time. By Default there are four Virtual terminals setup, but only three enabled for use.. To switch between them hold down the **ALT** key and press **F2** or **F3** (You start out at **F1**. This will switch you to the second or third Virtual terminal. Now you should have a login screen like the one you just left. There is nothing different about logging in here as opposed to the virtual terminal on **F1**. To get back to your original screen, hold down the **ALT** key and press **F1**

To add more Virtual terminals, you need SuperUser access to the `/dev` directory. Next you need to run the program `MAKEDEV`. `MAKEDEV` is a shell script, so you have to use the `sh` shell interpreter to execute it. At the command prompt type:

```
cd /dev
sh MAKEDEV vty 16
```

Note:

(Type `MAKEDEV` in all CAPS, Capitalization makes a big difference.)

This will create 16 Virtual terminals. They will be accessible with the **ALT + F?** keys. But Before you can use them, you have to enable them in the `/etc/ttys` file. Change directories to the `/etc/` directory and use `vi` to edit the `/etc/ttys` file.

```
cd /etc
vi ttys
```

```

#
# @(#)ttys      5.1 (Berkeley) 4/17/89
#
# name  getty          type  status  comments
#
# This entry needed for asking password when init goes to single-user mode
# If you want to be asked for password, change "secure" to "insecure" here
console none          unknown off secure
#
ttyv0  "/usr/libexec/getty Pc"      cons25  on  secure
# Virtual terminals
ttyv1  "/usr/libexec/getty Pc"      cons25  on  secure
ttyv2  "/usr/libexec/getty Pc"      cons25  on  secure
ttyv3  "/usr/libexec/getty Pc"      cons25  off secure
# Serial terminals
ttyd0  "/usr/libexec/getty std,9600"  unknown off secure
ttyd1  "/usr/libexec/getty std,9600"  unknown off secure
ttyd2  "/usr/libexec/getty std,9600"  unknown off secure
ttyd3  "/usr/libexec/getty std,9600"  unknown off secure
# Pseudo terminals
ttyp0  none          network
ttyp1  none          network
ttyp2  none          network

```

If you are running The X Windows System, you need to leave one of the Virtual Terminals turned off. By default, ttyv4 is turned off.

In the section labeled Virtual Terminals, you need to add a few lines. If you just made 16 virtual terminal devices, you now have 16 virtual terminals to configure. You need to have 16 lines that deal with virtual terminals in the `/etc/ttys` file; right now you only have 4. You can cut and paste the line that says:

```
ttyv2 "/usr/libexec/getty Pc" cons25 on secure
```

Move the cursor to that line and press **yy** then move the cursor down one line by pressing **j** and press **12p** to paste the line 12 times. Now you have 12 copies of `ttyv2`, you need to change this, numbering them in HEX. Start numbering them at 4, then 5,6,7,8,9,a,b,c,d,e,f.

When you are done, you should have `ttyv1 - ttyvf` configured in the `/etc/ttys` file.

In `vi` you can do this easily by positioning the cursor over the number you wish to change and pressing **r** the next key you press will replace that digit, press 4 to start with. Then press **j** to move down and press **r** followed by the next number, repeat as necessary.

when you have finished editing, press **ZZ** to quit and exit `vi` Now you will have to restart the `init` process. You must have SuperUser access. Type: `kill -HUP 1`

Now you are ready to use your Virtual Terminals.

20. Adding a Hard Disk

The hard drive needs to be physically installed. Make sure the cable is on properly, pin 1 is the closest to the power plug.(ie. the red stripe goes toward the power plug). And make sure that there is power to the Drive.

Now reboot to FreeBSD. During Boot up, FreeBSD should detect the drive. If it doesn't, reboot using the `"-c"` option. At the boot prompt and go in to the visual config and make sure that the

appropriate controller is enabled. If you have built a custom kernel and have disabled the extra IDE drive, or are adding a first SCSI drive to a kernel that has SCSI disabled, this is most likely the problem. If the kernel has the controller merely *disabled*, not removed, you can enable it in the visual config. Otherwise you will have to rebuild the kernel adding the appropriate controllers and devices.

If you missed the boot up section, type `dmesg` at the prompt after you have logged in. This will display your boot-up messages again.

```
/usr/> dmesg | more
```

If you are installing SCSI, be sure the drives are found by the SCSI BIOS and that the proper SCSI termination is enabled.

Once everything has been found, you are ready to start the FreeBSD configuration.

```
*****
```

The first thing you will need to do is to clear out whatever might have been on the disk previously. Most hard-drive manufacturers ship new drives already partitioned and formatted to the DOS FAT filesystem. For the remainder of this example, we'll be using a hard-drive that was detected as `sd1`. In our case, it is the second SCSI device on the chain (with SCSI ID 1). Here's how to get rid of that FAT garbage:

```
dd if=/dev/zero of=/dev/rsd1 count=100
```

Now you need to prepare a "disk label" on the drive. We'll assume in this example that you plan on using the entire drive under FreeBSD:

```
disklabel -Brw sd1 auto
```

* substitute `rsd1` and `sd1` for the disk that you are adding, don't do this to a disk you are using. This basically kills everything on the disk, and prepares it for use by FreeBSD. If this works, you shouldn't need to use `fdisk`.

If you are planning to use the entire disk as one partition under FreeBSD, then you can use the single default partition just created by disk label, otherwise you will need to use the disklabel editor and create the smaller partitions. You might want to add a swap partition to the disk, for example.

To do get in to the disk label editor for IDE disk 1, type:

```
disklabel -e wd0
```

```
# /dev/rwd0c:
type: ESDI
disk: wd0s1
label:
flags:
bytes/sector: 512
sectors/track: 51
tracks/cylinder: 13
sectors/cylinder: 663
cylinders: 722
sectors/unit: 479298
rpm: 3600
```

```

interleave: 1
trackskew: 0
cylinderskew: 0
headswitch: 0 # milliseconds
track-to-track seek: 0 # milliseconds
drivedata: 0

```

```

8 partitions:
#      size  offset  fstype  [fsize bsize bps/cpg]
a:    65536    0    4.2BSD    0     0     0 # (Cyl.  0 - 98*)
b:    85936  65536    swap                # (Cyl.  98*- 228*)
c:   479298    0   unused    0     0     0 # (Cyl.  0 - 722*)
e:    61440 151472    4.2BSD    0     0     0 # (Cyl. 228*- 321*)
f:   266386 212912    4.2BSD    0     0     0 # (Cyl. 321*- 722*)

```

Now comes the tricky part, getting the numbers right. The easiest way to allocate a partition is to select the number of cylinders you want to allocate to the partition and calculate from there. The size is the size of the partition in sectors. The offset is the last sector of the previous partition. Partition a starts at sector 0. Partitions need to start and end on cylinder boundaries. To find the cylinder boundary, multiply the tracks/cylinder # by the sectors/cylinder # and then multiply that by the number of cylinders you want to allocate. (Sectors * Tracks * Cylinders = Size_of_Partition) To find the offset, add the size of the previous partition to the offset of the previous partition. (Size + Offset = Offset_Of_Next_Partition)

The fstype is the type of filesystem you are putting on that partition. 4.2BSD is the filesystem type for a normal FreeBSD filesystem. swap is the filesystem type for a swap file. FAT is the filesystem type for DOS partitions. Partition a is reserved for the boot sector. Leaving it blank implies that you don't have/need a boot sector.

Partition b is reserved for the swap file. It cannot start with an offset of 0. There must be a partition that starts before it. Partition e can start at offset 0 and be any size, then the swap partition can be placed starting at the offset where partition e ended.

Partition c should not be changed. Start making your filesystems on partition e

Now you need to create a file system on the drive, you use the `newfs` command to do this. If you are using the whole disk, use the `c` partition automatically created by `disklabel`.

```
newfs /dev/rsd1c
```

The `c` partition always represents the entire device - so in this case we've just created a filesystem that spans the entire drive.

If you have separated the disk into smaller parts, you will need to use `newfs` to create a file system on each partition, except the swap partition.

Finally, you want to `mount` the drive. We'll assume here that you've created a "mount point" of `/d2` (using `mkdir`).

```
mount /dev/sd1c /d2
```

If all went well, you should now see something like this:

```
#vinyl % df -k
```

Filesystem	1K-blocks	Used	Avail	Capacity	Mounted on
/dev/sd0a	31775	13530	15703	46%	/
/dev/sd0s1f	1913091	1086207	673837	62%	/usr
/dev/sd0s1e	29727	12663	14686	46%	/var
procfs	4	4	0	100%	/proc
/dev/sd1c	4108717	616594	3163426	16%	/d2

Now you will need to create an entry in the `/etc/fstab` file so that it will get mounted each time you reboot. Otherwise you will have to execute the `mount` each time by hand. A `fstab` entry will need to be made for each partition you created, including the swap partition.

Also reference the `man` pages for: `fstab` `mount` `disklabel` `dmesg` `dd` and `newfs`

21. System Configuration File Options

- 21.1. *Important initial Boot-time options*
- 21.2. *Network configuration sub-section*
- 21.3. *Network daemon (miscellaneous) & NFS options:*
- 21.4. *Network Time Services options:*
- 21.5. *Network Information Services (NIS) options:*
- 21.6. *Network routing options:*
- 21.7. *System console options*
- 21.8. *Miscellaneous administrative options*
- 21.9. *Allow local configuration override at the very end here*

Note:

All arguments must be in double or single quotes. Multiple entries are separated by spaces.

21.1. Important initial Boot-time options

swapfile="NO"

Set to name of swapfile if aux swapfile desired.

apm_enable="NO"

Set to YES if you want APM enabled.

pccard_enable="NO"

Set to YES if you want to configure PCCARD devices.

pccard_mem="DEFAULT"

If `pccard_enable=YES`, this is card memory address.

pccard_ifconfig="NO"

Specialized pccard ethernet configuration (or NO).

```
local_startup="/usr/local/etc/rc.d /usr/X11R6/etc/rc.d"
```

Local Startup Directories. During startup, FreeBSD searches certain directories and executes any programs in them. Sort of like the “startup” folders in Windows. The `local_startup` option lets you specify which directories to search during startup. Multiple directories, separated by spaces, can be listed. They will be searched in the order listed and every file will be executed. This is an alternate approach to using `/etc/rc.local` to start programs.

21.2. Network configuration sub-section

Basic network options:

```
hostname="myname.my.domain"
```

Set this!

```
nisdomainname="NO"
```

Set to NIS domain if using NIS (or NO).

```
firewall="NO"
```

firewall type (see `/etc/rc.firewall`) or NO.

```
tcp_extensions="YES"
```

Allow RFC1323 & RFC1544 extensions (or NO).

```
network_interfaces="lo0"
```

List of network interfaces (lo0 is loopback).

```
ifconfig_lo0="inet 127.0.0.1"
```

default loopback device configuration.

```
ifconfig_lo0_alias0="inet 127.0.0.254 netmask 0xffffffff"
```

Sample alias entry.

21.3. Network daemon (miscellaneous) & NFS options:

```
syslogd_enable="YES"
```

Run syslog daemon (or NO).

```
syslogd_flags=""
```

Flags to syslogd (if enabled).

inetd_enable="YES"

Run the network daemon dispatcher (or NO).

inetd_flags=""

Optional flags to inetd.

named_enable="NO"

Run named, the DNS server (or NO).

named_flags="-b /etc/namedb/named.boot"

Flags to named (if enabled).

kerberos_server_enable="NO"

Run a kerberos master server (or NO).

rwhod_enable="NO"

Run the rwho daemon (or NO).

amd_enable="NO"

Run amd service with \$amd_flags (or NO).

amd_flags="-a /net -c 1800 -k i386 -d my.domain -l syslog /host /etc/amd.map"

Example

nfs_client_enable="NO"

This host is an NFS client (or NO).

nfs_client_flags="-n 4"

Flags to nfsiod (if enabled).

nfs_server_enable="NO"

This host is an NFS server (or NO).

nfs_server_flags="-u -t 4"

Flags to nfsd (if enabled).

weak_mountd_authentication="NO"

Running PCNFSD / other non-root nfsd (or NO).

nfs_reserved_port_only="NO"

Provide NFS only on secure port (or NO).

rpc_lockd_enable="NO"

Run NFS rpc.lockd (*broken!*) if nfs_server.

rpc_statd_enable="YES"

Run NFS rpc.statd if nfs_server (or NO).

portmap_enable="YES"

Run the portmapper service (or NO).

portmap_flags=""

Flags to portmap (if enabled).

xtend_enable="NO"

Run the X-10 power controller daemon.

xtend_flags=""

Flags to xtend (if enabled).

21.4. Network Time Services options:

timed_enable="NO"

Run the time daemon (or NO).

timed_flags=""

Flags to timed (if enabled).

ntpddate_enable="NO"

Run the ntpdate to sync time (or NO).

ntpddate_flags=""

Flags to ntpdate (if enabled).

xntpd_enable="NO"

Run xntpd Network Time Protocol (or NO).

xntpd_flags=""

Flags to xntpd (if enabled).

tickadj_enable="NO"

Run tickadj (or NO).

tickadj_flags="-Aq"

Flags to tickadj (if enabled).

21.5. Network Information Services (NIS) options:

nis_client_enable="NO"

We're an NIS client (or NO).

nis_client_flags=""

Flags to ypbind (if enabled).

nis_ypset_enable="NO"

Run ypset at boot time (or NO).

nis_ypset_flags=""

Flags to ypset (if enabled).

nis_server_enable="NO"

We're an NIS server (or NO).

nis_server_flags=""

Flags to ypserv (if enabled).

nis_ypxfrd_enable="NO"

Run rpc.ypxfrd at boot time (or NO).

nis_ypxfrd_flags=""

Flags to rpc.ypxfrd (if enabled).

nis_yppasswdd_enable="NO"

Run `rpc.yppasswdd` at boot time (or NO).

`nis_yppasswdd_flags=""`

Flags to `rpc.yppasswdd` (if enabled).

21.6. Network routing options:

`defaultrouter="NO"`

Set to default gateway (or NO).

`static_routes=""`

Set to static route list (or leave empty).

`gateway_enable="NO"`

Set to YES if this host will be a gateway.

`router_enable="YES"`

Set to YES to enable a routing daemon.

`router="routed"`

Name of routing daemon to use if enabled.

`router_flags="-q"`

Flags for routing daemon.

`mrouterd_enable="NO"`

Do multicast routing (see `/etc/mrouterd.conf`).

`ipxgateway_enable="NO"`

Set to YES to enable IPX routing.

`ipxrouted_enable="NO"`

Set to YES to run the IPX routing daemon.

`ipxrouted_flags=""`

Flags for IPX routing daemon.

`arproxy_all=""`

replaces obsolete kernel option ARP_PROXY_ALL.

21.7. System console options

keymap="NO"

keymap in /usr/share/syscons/keymaps/* (or NO).

keyrate="NO"

keyboard rate to: slow, normal, fast (or NO).

keybell="NO"

bell to duration.pitch or normal or visual (or NO).

keychange="NO"

function keys default values (or NO).

cursor="NO"

cursor type {normal|blink|destructive} (or NO).

scrnmap="NO"

screen map in /usr/share/syscons/scrnmaps/* (or NO).

font8x16="NO"

font 8x16 from /usr/share/syscons/fonts/* (or NO).

font8x14="NO"

font 8x14 from /usr/share/syscons/fonts/* (or NO).

font8x8="NO"

font 8x8 from /usr/share/syscons/fonts/* (or NO).

blanktime="NO"

blank time (in seconds) or "NO" to turn it off.

saver="NO"

screen saver: blank/daemon/green/snake/star/NO.

moused_type="NO"

See man page for rc.conf(8) for available settings.

moused_port="/dev/cuaa0"

Set to your mouse port (required if mousetype set).

moused_flags=""

Any additional flags to moused.

21.8. Miscellaneous administrative options

cron_enable="YES"

Run the periodic job daemon.

lpd_enable="YES"

Run the line printer daemon.

lpd_flags=""

Flags to lpd (if enabled).

sendmail_enable="YES"

Run the sendmail daemon (or NO).

sendmail_flags="-bd -q30m"

-bd is pretty mandatory.

savecore_enable="NO"

Save kernel crashdumps for debugging (or NO).

dumpdev="NO"

Device name to crashdump to (if enabled).

check_quotas="NO"

Check quotas (or NO).

accounting_enable="NO"

Turn on process accounting (or NO).

ibcs2_enable="NO"

Ibcs2 (SCO) emulation loaded at startup (or NO).

linux_enable="NO"

Linux emulation loaded at startup (or NO).

rand_irqs="NO"

Stir the entropy pool (like "5 11" or NO).

21.9. Allow local configuration override at the very end here

```
if [ -f /etc/rc.conf.local ]; then
    . /etc/rc.conf.local
fi
```

22. Adding and Installing Software

Adding/installing software in FreeBSD is easy. FreeBSD maintains a set of "packages" that are pre-built, ready to run binaries of almost all the popular programs out there. If you have the CD set, they are all on CD#1, which you can browse directly (there's a directory called packages..), or use the "/stand/sysinstall" tool, go to Post Configuration, and choose Packages. You'll get a nice little screen with all the packages categorized with a short description.

Often, you might want to `ftp` to `ftp.freebsd.org` to get the latest version of a package - `ftp://ftp.freebsd.org/pub/FreeBSD/packages-stable` is probably where you want to look. If you download a package, it will have a `.tgz` ending. You don't need to manually `untar/expand` this file. Just use the command `pkg_add`.

Ex: I've downloaded the packages called `spaz-1.32.tgz`

- 1. `su` to root (you need to be root to install packages)
- 2. `pkg_add spaz-1.32.tgz`

That's it! Now the package is installed and setup on your system. If you're using `csh` or `tcsh`, you should type `rehash` for your shell to rescan your system bin directories...

The package is registered in the `/var/db/pkg` directory - you can `cd` to there and get a directory listing to see what packages you have installed on your system. If you no longer want a package, you can remove it with the `pkg_delete` command: `pkg_delete spaz-1.32.tgz` (from the `/var/db/pkg` directory)

The FreeBSD ports system works exactly the same, with port registration making it easy to remove things you don't want afterwards.. The difference is that the ports are in "Source" form, and need to be compiled. If you have an Internet connection up, and have the ports tree installed (you were asked if you wanted it installed during the installation of FreeBSD) you can `cd /usr/ports` and take a look around.

Say you wanted to install the port from the `/usr/ports/net/spaz` directory -

- 1. `su root`
- 2. `cd /usr/ports/net/spaz`
- 3. `make`

Note:

(-> at this point the make program will actually download the port for you!

- 4. `make install` (if the make above succeeded fine, you can "make install" to actually install the fruits of your labour to the system bin/lib directories where it's now ready to use)
- 5. `make clean` (if you want to save space and clean up the object files and such that were created from the compile)

Once again, you use the `pkg_delete` program in `/var/db/pkg` to remove the ports that you have installed.

Part 5:

Setting up your Network

- 23. *Configuring Network Interfaces*
- 24. *Introduction to TCP/IP*
- 25. *Inetd, The "Super" Server*
- 26. *File Transfer Protocol*
- 27. *Email*
- 28. *POP3 Email*
- 29. *Windows File Sharing; SAMBA*
- 30. *Web Servers*
- 31. *NFS - Network File System*
- 32. *NIS - Network Information System*
- 33. *Setting up A Domain Name Server.*
- 34. *Router / GateWay*
- 35. *Firewall*
- 36. *IP Network Address Translation*

23. Configuring Network Interfaces

A network interface is a device that allows you to communicate with your network. These devices could be a network card, or a modem, or a variety of different pieces of equipment that allows communication. These devices need to be configured so that the network knows who you are. These interfaces will be configured to use TCP/IP networking protocols. This is usually setup in the `/etc/rc.conf`, or `/etc/sysconfig` if you have a 2.2.1 or older system.

24. Introduction to TCP/IP

24.1. IP - Internet Protocol

24.2. TCP - Transmission Control Protocol

TCP/IP is the protocol used for network communications by FreeBSD. Of course, TCP/IP is also the protocol used to communicate over the Internet, meaning FreeBSD can be used on the Internet “out of the box”. This chapter will attempt to give you a brief overview of the TCP/IP protocol, and what you need to understand in order to configure your FreeBSD machine for use on a TCP/IP network.

While we are going to give you enough of an introduction to TCP/IP in this book to get you up and running, if you’re in the position of maintaining a FreeBSD network (or any TCP/IP network for that matter) you’ll almost certainly need a better understanding of the TCP/IP protocol. TCP/IP is a large enough protocol that one could devote an entire book to it, and indeed many people have! An excellent book on the subject (IMHO) is *TCP/IP Network Administration* by Craig Hunt. Published by O’Reilly and Associates. This book gives a more in depth overview of TCP/IP and covers many of the common Unix services that we touch upon in this book. While the material is somewhat out of date when it concerns configuration files and so on, you’ll find that the concepts are still perfectly conveyed. If you’re interested in the guts of the protocol itself, Richard Stevens has three entire volumes written about TCP/IP starting with the theoretical and moving towards more practical knowledge. *TCP/IP Illustrated* is published by Addison Wesley.

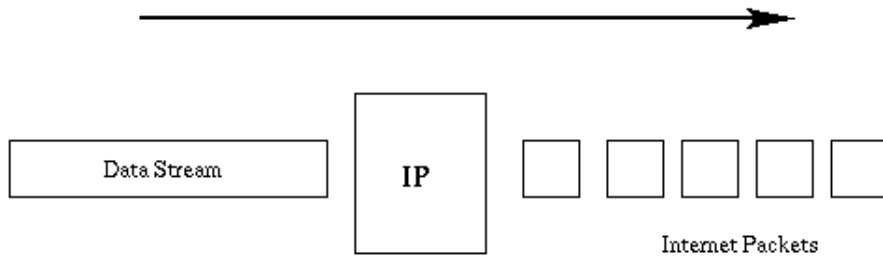
Note:

Although FreeBSD can also use the Apple-talk and IPX/SPX (Novell) protocols, this isn’t the norm and for this reason we’ll focus on TCP/IP as the protocol of choice for FreeBSD. The Applications section of this book will look at using FreeBSD as a replacement Apple and Novell File/Print server - we’ll discuss how to turn on the appropriate protocols there.

The first thing you need to know about TCP/IP is that it is actually a *suite* of protocols. Together, this suite of protocols lets us send data from one computer to another -- a source to a destination -- without having to specifically know the exact path the data will take on its journey. There are three important components that you need to know: IP, TCP, and UDP.

24.1. IP - Internet Protocol

Essentially, IP makes packets. What’s a packet? A packet is a small “unit”, or “package” of data. The easiest way to understand what a packet is, and how IP makes packets, is to look at an example. Let’s say you have a 200K file you wish to send over the Internet to a friend. You can think of this file as a “stream” of data - the file has a beginning, and a sequential set of bytes (200 thousand of them in this case) all the way up to the end of the file. Any continuous set of data like this is called a stream, due to the directional nature of the data. The most important thing to realize about a data stream is that the order that the bytes are in is critical. What IP does is break this continuous stream of data up into discrete packages (“packets”) that can be delivered independently. IP chops this data stream up into packets, wraps each packet to make it suitable for delivery over the Internet, and labels the packet with a source and destination address.



24.2. TCP - Transmission Control Protocol

What I didn't tell you before about IP is that, oddly enough, IP is not reliable. IP doesn't guarantee that a packet will reach its destination, or that a collection of packets that represent a single coherent data stream will arrive in the correct sequential order. That's where TCP comes in. TCP puts the packets in their correct order and makes sure all the packets arrived at the destination. Without TCP, IP would be nearly useless! I guess that's why they call the suite TCP/IP!! :-)

Hidden in the address information of each packet is a sequence number. IP increments the sequence number as it sends each packet out, and TCP uses these sequence numbers to reassemble the packets in the correct order when they arrive at the destination. TCP also uses the sequence number to determine if any packets are missing. If a particular packet does not show up (it was incorrectly routed, etc.) TCP requests the packet from the source again. It will keep trying until it has all the packets that make up a data stream. Hence, TCP makes the TCP/IP packet delivery mechanism "reliable".

TCP adds another layer of functionality that IP itself doesn't provide: ports. IP numbers specify individual computers on the network, whereas TCP port numbers specify services on that particular computer. Port numbers allow you to have many services running at once, and have a mechanism for making sure requests get to the correct service. The example you are perhaps most familiar with is a web server. WWW servers typically run on port 80. The standard notation for representing a IP number and Port number combination is ip:port - for example, 192.168.1.100:80 . TCP lets us send a packet intended for the web server, and a packet intended for the telnet server (port 23) to the same host IP number simultaneously. Great! For the record, port numbers range from 0 - 65000. All ports below 1024 are restricted, meaning that only root has the privilege of starting a service which listens on a port less than 1024. Anything above 1024 is fair game for any user process to use - no root privileges are required.

25. Inetd, The "Super" Server

As we discovered earlier, a FreeBSD system can run many programs simultaneously - such is the case for a FreeBSD network server; it will run a program for each service you wish to provide. The problem is, each service eats up a chunk of memory - and memory is always in demand (not to mention expensive!). The solution is to only start a service when it has a pending request, and to terminate it when a reply is sent to the client making the request. So how does a service know when to wake up?

That's where `inetd` comes in. It's called the Internet "super-server" since it controls many of the Internet services available on a typical Unix host. `inetd` monitors the port numbers of all requests that come to a server. When a request comes in, `inetd` looks in `/etc/services` for the name of the

service. As we can see below, the services file is a simple mapping of port numbers to service names:

```
mark:~112~/home/mark/FreeBSD % tail /etc/services
sd          9876/udp   #Session Director
amanda     10080/udp  #Dump server control
amandaidx  10082/tcp  #Amanda indexing
amidxtape  10083/tcp  #Amanda tape indexing
isode-dua  17007/tcp
isode-dua  17007/udp
biimenu    18000/tcp  #Beckman Instruments, Inc.
biimenu    18000/udp  #Beckman Instruments, Inc.
dbbrowse   47557/tcp  #Databeam Corporation
dbbrowse   47557/udp  #Databeam Corporation
mark:~113~/home/mark/FreeBSD %
```

Next, after `inetd` has the name of the service to start, it looks in its configuration file, `/etc/inetd.conf` to get setup information. The config file looks like this:

```
mark:~114~/home/mark/FreeBSD % head /etc/inetd.conf
#
# Internet server configuration database
#
#      @(#)inetd.conf  5.4 (Berkeley) 6/30/90
#
ftp      stream  tcp    nowait  root    /usr/libexec/ftpd      ftpd -l
telnet   stream  tcp    nowait  root    /usr/libexec/telnetd   telnetd
shell    stream  tcp    nowait  root    /usr/libexec/rshd      rshd
login    stream  tcp    nowait  root    /usr/libexec/rlogind   rlogind
finger   stream  tcp    nowait  nobody  /usr/libexec/fingerd   fingerd -s
mark:~115~/home/mark/FreeBSD %
```

The general format of the file is as follows:

- service name
- socket type
- protocol
- { wait|nowait } [/max-child]
- user
- server program
- server program arguments

Let's take a closer look at one of the lines and see what this all means.

```
telnet stream tcp nowait root /usr/libexec/telnetd telnetd
```

The first field, `telnet`, is the name of the service. This name must match exactly the name found in `/etc/services`. The second and third fields, `stream tcp`, describes the kind of connection the service will make. In this case, `tcp` will handle the packet ordering so that the downloaded information appears to be a steady stream of error-free data. The only other option for this field,

`dgram udp`, specifies that UDP, not TCP, will handle the packet ordering (the so called “unreliable” datagram form of TCP/IP).

The fourth field, `nowait` tells the service to spawn itself for each service request that comes to the server - even if one instance of the program is already running. The other option, `wait` says to run the programs sequentially, waiting for the first program to finish before starting up another instance.

The fifth field, `root`, specifies the user ID used to run the service. The majority of programs started from `inetd` run as the user “root” or “nobody”. The remaining fields tell `inetd` where to find the program, and any arguments that are required (in the case of `telnet`, no arguments are present).

Sidebar:

The disadvantage of running programs through `inetd` is that it isn't too efficient for services that are constantly being run, perhaps once a second, or even hundreds of times per second. Why is that you ask? Well, although running a program through `inetd` generally reduces the amount of memory used, it introduces a penalty in that it must “fork” the program each time a request comes in. “fork” simply means that `inetd` starts a new process for the service. We won't get into the technical details of the overhead required to fork a process. Let's just say that there is an overhead, and under a busy system is enough to cause a considerable slow down. For this reason, some services are run as “daemons” instead of through `inetd`. How you decide to run a service depends entirely on what you feel is important. If performance is critical, run the program as a daemon. If you want to reduce memory usage, run the program through the `inetd` super-server. Typically, FTP, finger, talk, POP3, and `telnetd` are run through the `inetd` server, but things such as web servers and the mail delivery server (`sendmail`, usually) are run as daemons. Web servers like the NCSA and Apache servers even go through the trouble of “pre-forking” several copies of themselves to eliminate the overhead of forking under large numbers of requests! If you install a new service, the documentation that comes with the program will generally give tips on which approach is best for that particular application.

26. File Transfer Protocol

Not Yet Scheduled

27. Email

Not Yet Scheduled

28. POP3 Email

Not Yet Scheduled

29. Windows File Sharing; SAMBA

if you are installing from the ports it installs everything to `/usr/local/samba/` The binary files are stored in the `bin` directory. The config file is located in `lib/smb.conf`

if you are installing from a package it installs stuff to `/usr/local/sbin` and the config file is in `/usr/local/etc/smb.conf` You will need to read the man page on `smb.conf` to understand how to create the `smb.conf` configuration file. The Documentation on `smb.conf` is extremely well written and comprehensive. Everything you could possibly put in the configuration file is included in that man page. To read it, type:

```
man smb.conf
```

Use `testparm` to check your config file. `testparm` will also tell you where Samba is looking for your default config file.

No matter which way you installed samba, you will need to modify your `/etc/rc.local` file to start Samba at boot up. To do this, include these lines in your `rc.local` file:

```
/usr/local/sbin/smbd -D /usr/local/sbin/nmbd -D
```

30. Web Servers

Not Yet Scheduled

31. NFS - Network File System

Coming soon... NFS lets client Unix machines mount remote file systems as if they were local.

32. NIS - Network Information System

Coming soon.. NIS lets you distribute user, group, and other information to Unix clients from a centralized server. Typically used in conjunction with NFS

33. Setting up A Domain Name Server.

First off you have to have a valid domain name to use. The place that your domain is registered with needs to point at the IP address of your primary DNS Machine. This should all be taken care of by INTERNIC or the DNS one level above you.

If you are setting up a sub-domain, you will need to create a mapping in your zone pointing to the primary DNS for that domain.

First:

In the `/etc/sysconfig` file, or in the new `/etc/rc.conf`, Add the line:

```
namedflags="-b /etc/namedb/named.boot"
```

This will start the "Name Daemon" at boot up and set it to read the file `/etc/namedb/named.boot`

as the configuration file.

Now you need to `cd` to the `/etc/namedb/` directory.

Now type:

```
sh make-localhost
```

This will create the `localhost.rev` in that directory. This file is necessary for all local traffic to be properly dealt with.

Now create or edit the file `named.boot`

```
+++++NAMED.BOOT EXAMPLE ++++++
;semi colons comment out statments.
; sortlist 128.3.0.0
; The sort list gives higher priority to certain domains in the case
;of multi-homed hosts.
directory      /etc/namedb
; this denotes the directory that named should look to find all of the
; source files.
; type      domain                source host/file                backup file
cache      .                        named.root
; named keeps a cache of recently looked up hostname in the file mentioned.
primary    0.0.127.IN-ADDR.ARPA    localhost.rev
; this is the local host entry needs to be there. Usually automatic.
primary    Berkeley.EDU            your.domain.zone
primary    32.128.IN-ADDR.ARPA     your.domain.rev
; These lines are a pair. They represent the primary domain you control
; The first one is your domain and the second is the reverse lookup table.
; You need to have each DNS entry entered in to both files.
; You will need a pair of primary lines for each primary domain that
; you administer.
secondary  Berkeley.EDU            128.32.130.11 128.32.133.1    ucbhosts.bak
secondary  32.128.IN-ADDR.ARPA     128.32.130.11 128.32.133.1    ucbhosts.rev
; These lines are a pair also. They represent the domains that you are
; interested in knowing about if their DNS goes down. Or you may just be
; the back up DNS for them.
; Instead of source files, you specify the host that is the primary DNS
; for that domain. You must also specify that filename that named will
; store the temporary table in.
; You need a pair of these lines for each of the Domains that you are
; a secondary DNS for.
+++++End Example Named.boot File+++++
```

If you had the domain “my.domain.com”, the IP Address range 10.20.40.0 - 10.20.40.255,

and wanted to be the primary DNS for it, you would put this line in your `named.boot` file:

```
primary my.domain.com my.domain.com.zone
```

```
primary 40.20.10.IN-ADDR.ARPA my.domain.com.rev
```

Now you have to make the primary source file. Create a file called `/etc/named/your.domain.zone`. Substitute your actual domain name. It must match the filename specified in the primary entry of `named.boot`

Here is an example of such a file.

```
IN      soa      bbcc.ctc.edu.  root.bbcc.ctc.edu. (
                28      ;serial
                10800   ;refresh every 3 hours
                900     ;retry every 15 minutes
                604800  ;expire after a week
                86400   ;minimum of a day
                )
                IN      NS      bbcc.ctc.edu.
                IN      NS      ctc.ctc.edu.
                IN      NS      bb.cc.wa.us.
bbcc.ctc.edu.  IN      A      134.39.180.254
mail          IN      CNAME   bbcc.ctc.edu.
www          IN      CNAME   bbcc.ctc.edu.
irc          IN      CNAME   bbcc.ctc.edu.
bigbend.ctc.edu.  IN    CNAME   bbcc.ctc.edu.
athena       IN      A      134.39.180.5
            IN      HINFO   intel 586-133 winnt
proto       IN      A      134.39.180.6
aries       IN      CNAME   bb.cc.wa.us.
sal         IN      A      134.39.180.8
dialup3     IN      A      134.39.180.252
;end of file.
```

5) Now you have to make the primary reverse lookup file. Create a file called `/etc/named/your.domain.rev`

Substitute your actual domain name. It must match the filename specified in the primary entry of `named.boot`

Here is an example of such a file.

```
IN      soa      bbcc.ctc.edu.  root.bbcc.ctc.edu. (
                28      ;serial
                10800   ;refresh every 3 hours
                900     ;retry every 15 minutes
                604800  ;expire after a week
                86400   ;minimum of a day
                )
8       IN      PTR      sal.bbcc.ctc.edu.
252    IN      PTR      dialup.bbcc.ctc.edu.
```

34. Router / GateWay

Not Yet Scheduled

35. Firewall

Not Yet Scheduled

36. IP Network Address Translation

- 36.1. *1) Loading the Kernel Module*
- 36.2. *2) Setting up the NAT Rules*
- 36.3. *3) Loading the NAT Rules:*
- 36.4. *4) Enable Routing between interfaces.*
- 36.5. *5) Static Routes to Subnet Ranges*
- 36.6. *6) Make sure that you have your interfaces configured.*

After you have installed IpFilter: You will need to change three files:

```
/etc/rc.local  
/etc/sysconfig  
/etc/natrules
```

This was tested using ipfilter 3.1.4 and FreeBSD 2.1.6-RELEASE

36.1. 1) Loading the Kernel Module

If you are using a Kernel Loadable Module you need to edit your `/etc/rc.local` file and load the module at boot time.

use the line: `modload /lkm/if_ip1.o`

If you are not loading a kernel module, skip this step.

36.2. 2) Setting up the NAT Rules

Make a file called `/etc/natrules` put in the rules that you need for your system. If you want to use the whole 10 Network. Try:

```
map fxp0 10.0.0.0/8 -> 208.8.0.1/32 portmap tcp/udp 10000:65000
```

Here is an explanation of each part of the command:

`map` starts the command.

`fxp0` is the interface with the real internet address.

`10.0.0.0` is the subnet you want to use.

`/8` is the subnet mask. ie `255.0.0.0`

`208.8.0.1` is the real IP address that you use.

/32 is the subnet mask 255.255.255.255, ie only use this IP address.

```
portmap tcp/udp 10000:65000
```

tells it to use the ports to redirect the tcp/udp calls through The one line should work for the whole network.

36.3. 3) Loading the NAT Rules:

The NAT Rules will need to be loaded every time the computer reboots. In your `/etc/rc.local` put the line: `ipnat -f /etc/natrules`

To check and see if it is loaded, as root type: `ipnat -ls`

36.4. 4) Enable Routing between interfaces.

Tell the kernel to route these addresses. In the `/etc/rc.conf` put the line:

```
Gateway=YES
```

Or configure it by had by putting this line in the `/etc/rc.local` file :

```
sysctl -w net.inet.ip.forwarding=1
```

36.5. 5) Static Routes to Subnet Ranges

Now you have to add a static routes for the subnet ranges. Edit your `/etc/rc.conf`, or on an older system, your `/etc/sysconfig` to add them at bootup.

```
static_routes="foo" route_foo="10.0.0.0 -netmask 0xf0000000 -interface 10.0.0.1"
```

36.6. 6) Make sure that you have your interfaces configured.

I have two Intel Ether Express Pro B cards. One is on 208.8.0.1 The other is on 10.0.0.1 You need to configure these in the `/etc/sysconfig`

```
network_interfaces="fxp0 fxp1"  
ifconfig_fxp0="inet 208.8.0.1 netmask 255.255.255.0"  
ifconfig_fxp1="inet 10.0.0.1 netmask 255.0.0.0"
```

Note:

When using ftp from a client computer on the virtual network, you will need to use passive mode. Otherwise, it will time out trying to get a directory listing.

Setting up the X Window System

- 37. *Installing the X binaries*
- 38. *Configuring X for your Hardware*
- 39. *Choosing a X window manager*
- 40. *Installing X programs*
- 41. *Tips and Tricks for X*

37. Installing the X binaries

Before X can be run, the X Window System needs to be installed. There are many different X Window Systems that you can get, however only XFree86 is available for Free. XFree86 is available on the FreeBSD CD, from the FTP site or directly from www.xfree86.org. From here on out, I will refer to XFree86 as X or the X Window System. Installing the X Window System can require a significant amount of disk space.

The X Window System can be installed during a regular FreeBSD install, or done after the system is up and running. X is included in several of the Distribution Sets provided in the FreeBSD install. These include

- X User
- X Developer
- Everything

If you have installed one of these Distribution Sets, the X System is on your hard drive and you can skip ahead to configuration.

During a Custom Install, you can select the X Window System as part of the normal install procedure. If you don't install a Distribution Set that contains the X Window System, you have a second chance to install it during the post-install configuration menu. To install X from the post-install configuration menu, select `Distributions` and chose the `X Window Distribution`.

If your system is already running, you can install X from the ports collection. You will need to be root to do this:

- `cd /usr/ports/x11/XFree86`
- then type `make install`.

The program will be downloaded from the FTP site and then compiled and installed. This may take a while depending on the speed of your computer.

If neither of these methods are appealing, you can download the source code directly from www.xfree86.org and compile and install it yourself. Good Luck.

38. Configuring X for your Hardware

- 38.1. *xf86config*
- 38.2. *XF86Setup*

Before you can configure X to use your hardware, you need to know what your hardware is. There are three things you absolutely have to know (or at least be able to guess correctly):

- The type of mouse and where its connected. (ie. PS/2 or Serial on COM1)
- The brand of video card you have. (Needs to be a supported type.)
- The horizontal and vertical sync limits of your monitor.(Found in your monitor manual.)

If you know these three things, setting up X is easy. The rest is just a matter of personal taste and geographic/language requirements. Once you have obtained this data, you will need to run either the text based config program (*xf86config*), or the graphical config program (*XF86Setup*). You do not need to run both.

38.1. *xf86config*

xf86config is located in `/usr/X11R6/bin/`. If that is part of your path, you can just type ***xf86config*** and begin the configuration. Otherwise, you can type `/usr/X11R6/bin/xf86config`. You need to be root to do this.

The first thing it is going to ask you is which kind of mouse you have attached to the computer:

First specify a mouse protocol type. Choose one from the following list:

1. Microsoft compatible (2-button protocol)
2. Mouse Systems (3-button protocol)
3. Bus Mouse
4. PS/2 Mouse
5. Logitech Mouse (serial, old type, Logitech protocol)
6. Logitech MouseMan (Microsoft compatible)
7. MM Series
8. MM HitTablet
9. Microsoft IntelliMouse

Select 1 - 9 based on which type of mouse you have. If you select a mouse that has 3 buttons, it will ask you if you want to enable the center button, its called `ChordMiddle`:

Please answer the following question with either 'y' or 'n'.
Do you want to enable ChordMiddle?

If you don't enable the Middle Button, it will ask you if you want to emulate the middle button. With the 3 button emulation enabled, if you press both mouse buttons simultaneously, X will pretend that you have pressed the center mouse button.

Please answer the following question with either 'y' or 'n'.
Do you want to enable Emulate3Buttons?

Next it wants to know where the mouse is plugged into.

Now give the full device name that the mouse is connected to, for example `/dev/tty00`. Just pressing enter will use the default, `/dev/mouse`.

Mouse device:

A mouse is usually plugged into either COM1, COM2, or the PS/2 Port.

- /dev/ttyd0 (COM1)
- /dev/ttyd1 (COM2)
- /dev/psm0 (PS/2 Port)

Now it will ask you if you need to be able to re-map keyboard keys. If you don't know, just press **ENTER**

Please answer the following question with either 'y' or 'n'.
Do you want to use XKB?

This is for special language support, if you speak only English, just press **ENTER**.

If you want your keyboard to generate non-ASCII characters in X, because you want to be able to enter language-specific characters, you can set the left Alt key to Meta, and the right Alt key to ModeShift.

Please answer the following question with either 'y' or 'n'.
Do you want to enable these bindings for the Alt keys?

These are the Horizontal Refresh rates found in your Monitor manual. #2 is pretty safe if you don't know what your monitor does.

```
hsync in kHz; monitor type with characteristic modes
1 31.5; Standard VGA, 640x480 @ 60 Hz
2 31.5 - 35.1; Super VGA, 800x600 @ 56 Hz
3 31.5, 35.5; 8514 Compatible, 1024x768 @ 87 Hz interlaced (no 800x600)
4 31.5, 35.15, 35.5; Super VGA, 1024x768 @ 87 Hz interlaced, 800x600 @ 56
5 31.5 - 37.9; Extended Super VGA, 800x600 @ 60 Hz, 640x480 @ 72 Hz
6 31.5 - 48.5; Non-Interlaced SVGA, 1024x768 @ 60 Hz, 800x600 @ 72 Hz
7 31.5 - 57.0; High Frequency SVGA, 1024x768 @ 70 Hz
8 31.5 - 64.3; Monitor that can do 1280x1024 @ 60 Hz
9 31.5 - 79.0; Monitor that can do 1280x1024 @ 74 Hz
10 31.5 - 82.0; Monitor that can do 1280x1024 @ 76 Hz
11 Enter your own horizontal sync range
```

Enter your choice (1-11):

You need to enter the Vertical refresh rate of your monitor. #2 is a very common SVGA setting.

```
1 50-70
2 50-90
3 50-100
4 40-150
5 Enter your own vertical sync range
```

Enter your choice:

Answer y to this next question and find your card in the card database.

Do you want to look at the card database?

On the Left is the name of the card, and on the right is the card chipset.

```
0 2 the Max MAXColor S3 Trio64V+ S3 Trio64V+
```

1	928Movie	S3 928
2	AGX (generic)	AGX-014/15/16
3	ALG-5434(E)	CL-GD5434
4	ASUS PCI-AV264CT	ATI-Mach64
5	ASUS PCI-V264CT	ATI-Mach64
6	ASUS Video Magic PCI V864	S3 864
7	ASUS Video Magic PCI VT64	S3 Trio64
8	ATI 3D Pro Turbo	ATI-Mach64
9	ATI 3D Xpression	ATI-Mach64
10	ATI 3D Xpression+ PC2TV	ATI-Mach64
11	ATI 8514 Ultra (no VGA)	ATI-Mach8
12	ATI All-in-Wonder	ATI-Mach64
13	ATI Graphics Pro Turbo	ATI-Mach64
14	ATI Graphics Pro Turbo 1600	ATI-Mach64
15	ATI Graphics Ultra	ATI-Mach8
16	ATI Graphics Ultra Pro	ATI-Mach32
17	ATI Graphics Xpression with 68875 RAMDAC	ATI-Mach64

Enter a number to choose the corresponding card definition.
Press enter for the next page, q to continue configuration.

The term X server used in this next section makes more sense if you think of it as “video card driver” instead. The driver suggested by the card database definition will always perform better. However if the card is not in the database, choose the next best card driver. #5 will contain the X server suggested by the card definition.

- 1 The XF86_Mono server. This a monochrome server that should work on any VGA-compatible card, in 640x480 (more on some SVGA chipsets).
- 2 The XF86_VGA16 server. This is a 16-color VGA server that should work on any VGA-compatible card.
- 3 The XF86_SVGA server. This is a 256 color SVGA server that supports a number of SVGA chipsets. On some chipsets it is accelerated or supports higher color depths.
- 4 The accelerated servers. These include XF86_S3, XF86_Mach32, XF86_Mach8, XF86_8514, XF86_P9000, XF86_AGX, XF86_W32, XF86_Mach64, XF86_I128 and XF86_S3V.

These four server types correspond to the four different "Screen" sections in XF86Config (vga2, vga16, svga, accel).

- 5 Choose the server from the card definition, XF86_S3.

Which one of these screen types do you intend to run by default (1-5)?

If you don't know about symbolic links, just say 'y'

Please answer the following question with either 'y' or 'n'.
Do you want me to set the symbolic link?

If you don't know, just press **ENTER**. It is automatically autodetected.

How much video memory do you have on your video card:

- 1 256K
- 2 512K
- 3 1024K
- 4 2048K
- 5 4096K
- 6 Other

Enter your choice:

Unless the card definition tells you what to put, just press 'q'.

1	ATT 20C490 (S3 and AGX servers, ARK driver)	att20c490
2	ATT 20C498/21C498/22C498 (S3, autodetected)	att20c498
3	ATT 20C409/20C499 (S3, autodetected)	att20c409
4	ATT 20C505 (S3)	att20c505
5	BrookTree BT481 (AGX)	bt481
6	BrookTree BT482 (AGX)	bt482
7	BrookTree BT485/9485 (S3)	bt485
8	Sierra SC15025 (S3, AGX)	sc15025
9	S3 GenDAC (86C708) (autodetected)	s3gendac
10	S3 SDAC (86C716) (autodetected)	s3_sdac
11	STG-1700 (S3, autodetected)	stg1700
12	STG-1703 (S3, autodetected)	stg1703

The card definition has Ramdac "bt485".

Enter a number to choose the corresponding RAMDAC.
Press enter for the next page, q to quit without selection of a RAMDAC.

Unless the card definitions tells you which to pick, just press **ENTER**.

1	Chrontel 8391	ch8391
2	ICD2061A and compatibles (ICS9161A, DCS2824)	icd2061a
3	ICS2595	ics2595
4	ICS5342 (similar to SDAC, but not completely compatible)	ics5342
5	ICS5341	ics5341
6	S3 GenDAC (86C708) and ICS5300 (autodetected)	s3gendac
7	S3 SDAC (86C716)	s3_sdac
8	STG 1703 (autodetected)	stg1703
9	Sierra SC11412	sc11412
10	TI 3025 (autodetected)	ti3025
11	TI 3026 (autodetected)	ti3026
12	IBM RGB 51x/52x (autodetected)	ibm_rgb5xx

The card definition has Clockchip "icd2595"

Just press enter if you don't want a Clockchip setting.
What Clockchip setting do you want (1-12)?

Unless the card definition says Not to, go ahead and press **y**.

You must be root to be able to run X -probeonly now.

The card definition says to NOT probe clocks.
Do you want me to run 'X -probeonly' now?

These are the list of resolutions that you can switch between. The defaults usually work. Unless you want to change these, just press **ENTER**.

Currently it is set to:

"640x480" "800x600" "1024x768" "1280x1024" for 8bpp
"640x480" "800x600" "1024x768" "1280x1024" for 16bpp
"640x480" "800x600" "1024x768" "1280x1024" for 24bpp
"640x480" "800x600" "1024x768" for 32bpp

Note that 16, 24 and 32bpp are only supported on a few configurations.
Modes that cannot be supported due to monitor or clock constraints will be automatically skipped by the server.

- 1 Change the modes for 8pp (256 colors)
- 2 Change the modes for 16bpp (32K/64K colors)
- 3 Change the modes for 24bpp (24-bit color, packed pixel)
- 4 Change the modes for 32bpp (24-bit color)
- 5 The modes are OK, continue.

Enter your choice:

You need to press **y** here to save your configuration.

I am going to write the XF86Config file now. Make sure you don't accidentally overwrite a previously configured one.

Shall I write it to /etc/XF86Config?

38.2. XF86Setup

39. Choosing a X window manager

- 39.1. *twm*
- 39.2. *fvwm*
- 39.3. *fvwm95*
- 39.4. *enlightenment*
- 39.5. *windowmaker*
- 39.6. *kde*

39.1. twm

The window manager is setup by default. Its not very user friendly and has no s upport for virtual desktops.

39.2. fvwm

fvwm has nine virtual desktops, a window 3.1 look and feel, and is currently use d for the new Novell 5 server desktop.

39.3. fvwm95

Looks like Windows 95 with all the fvwm toolbars. Very stable and has nine virt ual desktops.

39.4. enlightenment

A reallly cool looking desktop. Has support for pluggable desktop themes, such as “aliens”. Supports four virtual desktops. Not the most stable yet.

39.5. windowmaker

Allows you to create and delete virtual desktops on demand.

39.6. kde

kde is a complete desktop environment. It comes complete with all its own utilities and programs. Therefore its HUGE, but versatile and easy to setup.

To install `cd /usr/ports/x11/kde make install`

40. Installing X programs

Netscape

Staroffice

41. Tips and Tricks for X

- `ctrl + alt + backspace`

Close X and return to the shell

- `ctrl + alt + keypad plus`

Change screen resolutions to the next one in the list.

- `ctrl + alt + keypad minus`

Change screen resolutions to the previous one in the list.

- `ctrl + alt + F1`

Switch out of X to `ttyv0`

- `alt + F4`

Switch back to X from the `tty`'s (provided X is on `ttyv4`)

Part 7:

System Administration

42. *Setting up Printing Services*

43. *User Administration*

44. *System Administration*

42. Setting up Printing Services

There are three steps to setting up printing services:

1. Edit the `/etc/printcap` and make an entry for the printer.
2. Create a spool directory for `lpd` to store print jobs in.
3. Start the line printer daemon `lpd`

Information about what printers are available to print from is contained in the file `/etc/printcap`. FreeBSD can print to printers connected to any valid port on the server, both parallel and serial. It can also print to any TCP/IP capable printer, such as those attached to other FreeBSD Boxes, and those attached to network printing devices, like Jet Direct cards.

Here is the standard `/etc/printcap` file:

```
#      @(#)printcap      5.3 (Berkeley) 6/30/90
# $Id: printcap,v 1.5 1996/10/13 16:52:33 joerg Exp $

#
# This enables a simple local "raw" printer, hooked up to the first
# parallel port.  No kind of filtering is done, so everything you
# pass to the "lpr" command will be printed unmodified.
#
# Remember, for further print queues you're going to add, you have to
# chose different spool directories (the "sd" capability below),
# otherwise you will greatly confuse lpd.
#
# For some advanced printing, have a look at the "apsfilter" package.
# It plugs into the lpd system, allowing you to print a variety of
# different file types by converting everything to PostScript(tm)
# format.  If you don't have a PostScript(tm) printer, don't panic,
# but do also install the "ghostscript" package.
#
# Do also refer to section 7 (Printing) of the handbook.  A local copy
# can be found under /usr/share/doc/handbook/handbook.{html,latin1}.
#
lp|local line printer:\
    :lp=/dev/lpt0:sd=/var/spool/output/lpd:lf=/var/log/lpd-errs:
#
# Sample remote printer.  The physical printer is on machine "lphost".
# NB: you cannot perform any kind of local filtering directly.  If
# you need local filters (e.g. LF -> CR-LF conversion for HP printers),
# create a filter script that recursively calls lpd with another -P
# argument after filtering.
#
#remote|sample remote printer:\
    :rm=lphost:sd=/var/spool/output/lphost:lf=/var/log/lpd-errs:
```

The line

```
lp|local line printer:\
    :lp=/dev/lpt0:sd=/var/spool/output/lpd:lf=/var/log/lpd-errs:
```

from the above file, enables the default generic printer attached to `/dev/lpt0`. (Same as LPT1 on DOS systems)

For a printer to print, it needs a spool directory, or a place to temporarily store the file it is printing. This is sometimes called a queue; a print queue. When a print job is sent, but cannot be printed, it will be placed in this directory as a file. As soon as the printer is available, it will be sent to the printer. The spool directory in the example above is `/var/spool/output/lpd`

Note:

Post script printers are just like regular printers. Nothing special is required to set them up. Setting up a Non-post script printer to accept post script documents requires the `a2ps` or `apsfilter` package from the `ports/packages` collection

Also reference the `man` pages for: `lpd` `lpr` `lpc` `pr`

43. User Administration

43.1. *Deleting Old Users*

43.2. *Moving Home Directories*

43.3. *vipw; chpass;*

43.4. *Managing Groups; /etc/group*

43.1. Deleting Old Users

Deleting users in FreeBSD is super simple thanks to the handy program `rmuser`. All you do is type `rmuser username`, where `username` is the name of the user who is about to be nuked.

`rmuser` basically does four things for you:

1. Removes the user's home directory, as it is defined in the password file. Usually, this will be `/home/username`.
2. Removes the user's mail spool. This is located in `/var/mail/username`
3. Removes the user from any groups they are present in. Groups are stored in `/etc/group`.
4. Removes the user from the password file, `/etc/passwd`

It's good to know the steps that are involved in cleansing a user from your system. If you're a system administrator, it will often be necessary to temporarily remove users from a system by preventing them from logging in. You can do this by simply editing the password file with `vipw` and placing a `#` at the front of their encrypted password. When you want to reinstate the user you just remove the `#` sign!

43.2. Moving Home Directories

Not Yet Scheduled

43.3. vipw; chpass;

Not Yet Scheduled

43.4. Managing Groups; /etc/group

Not Yet Scheduled

44. System Administration

44.1. *booting -c or -s*

44.2. *File Systems; mount; fsck; fdisk; umount*

44.3. *MAKEDEV; tar; cron;*

44.1. booting -c or -s

Not Yet Scheduled

44.2. File Systems; mount; fsck; fdisk; umount

Not Yet Scheduled

44.3. MAKEDEV; tar; cron;

Not Yet Scheduled

Part 8:

Practical FreeBSD Applications

45. *FreeBSD As A Dialup Server*

46. *FreeBSD as a LAN Server*

45. FreeBSD As A Dialup Server

45.1. *Installing a Modem **

45.2. *Configuring the modems*

45.3. *Multiport modem boards.**

45.4. *pppd*

45.1. Installing a Modem *

45.1.1. *Pre-Install*

45.1.2. *Install Hardware.*

45.1.3. *Configuring a Modem IRQ/COM Port and Device Driver*

45.1.1. Pre-Install

Before you install the modem in your computer, you need to know a few things:

1. The COM port the modem is currently on.

There are 4 COM ports available on a standard Computer. The Mouse will usually take up the first one; unless you have a PS/2 Mouse.

2. How to Change the COM ports on the Modem.

Most Modems come with Jumpers on the board that you can change, however some of the new ones are Plug N Play.

3. What the IRQ setting currently is.

There are 15 IRQ setting available, but most of them are already in use. Commonly you can assign IRQ's 3,4,5,9, and 10.

4. How to Change the IRQ setting.

As with COM Ports, most Modems come with Jumpers on the board that you can change, however some of the new ones are Plug N Play.

5. What Speed the Modem is.

Usually it will be a 14,400 Bps, 28,800 Bps, or 33,600 Bps. There are a few new ones that run at 56,700 Bps or higher.

6. What COM ports are available in your Computer.

7. What IRQ's are available in your Computer.

Now you want to select an IRQ and a COM port that is available in your computer and set the modem jumpers to match it.

45.1.2. Install Hardware.

Now we need to install the Modem in your Computer.

1. Shutdown FreeBSD; type: `shutdown -h now` as root.
2. Turn off Computer and unplug it. (just to Be safe)
3. Remove Cover
4. Install Modem; Make sure the modem is securely seated in slot and the retaining screw is tight.
5. Plug Phone Cord into Modem.
6. Replace cover; plug it in and turn it on.

Now your Modem is installed.

45.1.3. Configuring a Modem IRQ/COM Port and Device Driver

Turn on your Computer and Boot to FreeBSD with the `-c` Option. If your computer is already on, you will need to reboot it. At start-up, you will see the Boot: prompt; type `-c` and press **ENTER**.

```
>> FreeBSD BOOT @ 0x10000: 639/31744 K of Memory
Usage: [[[0:]fd](0,a)]/kernel][-abcCdghrsv]
Use 1:sd(0,a)/kernel to boot sd0 if it is BIOS drive 1
Use ? for file list or Enter for defaults
Boot: -c
```

Now it should start booting and take you to a `config>` prompt. Here you need to type **visual**

45.2. Configuring the modems

Not Yet Scheduled

45.3. Multiport modem boards.*

BocaBoard 16 Port

In order to use a Multi port board, you need to modify your kernel. If you have never built your own custom kernel, see the directions on Creating Custom Kernels in the Basic FreeBSD programming section.

In your Custom Kernel Config File, add the following line at the top of the document:

```
options          COM_MULTIPORT          #Multi-Port Support
```

Next you need to add support for the extra COM ports. Each new COM port needs to have a "device" entry in the Kernel Config File or the Kernel will not be able to recognize it as a valid piece of hardware. BOCA Multi-port boards use a shared IRQ.

You will need to specify the Port memory starting address and the IRQ of the card using the jumpers found on the card. Follow the instructions provided by BOCA for the card that you use.

A BOCA 8 port board will not work because it doesn't have the modem control capabilities. You will also need to buy db25 to rj45/10 connectors to connect from the BOCA breakout box to the external modems.

You will need to add the following to your Kernel Config File, assuming that your starting port address is 100h and the IRQ is 12.

```

device      sio0    at isa? port "IO_COM1" tty irq 4 vector siointr
#device     sio1    at isa? port "IO_COM2" tty irq 3 vector siointr
#device     sio2    at isa? disable port "IO_COM3" tty irq 5 vector siointr
#device     sio3    at isa? disable port "IO_COM4" tty irq 9 vector siointr

device sio1  at isa? port 0x100    tty    flags 0x1005
device sio2  at isa? port 0x108    tty    flags 0x1005
device sio3  at isa? port 0x110    tty    flags 0x1005
device sio4  at isa? port 0x118    tty    flags 0x1005

device sio5  at isa? port 0x120    tty    flags 0x1005
device sio6  at isa? port 0x128    tty    flags 0x1005
device sio7  at isa? port 0x130    tty    flags 0x1005
device sio8  at isa? port 0x138    tty    flags 0x1005

device sio9  at isa? port 0x140    tty    flags 0x1005
device sio10 at isa? port 0x148    tty    flags 0x1005
device sio11 at isa? port 0x150    tty    flags 0x1005
device sio12 at isa? port 0x158    tty    flags 0x1005

device sio13 at isa? port 0x160    tty    flags 0x1005
device sio14 at isa? port 0x168    tty    flags 0x1005
device sio15 at isa? port 0x170    tty    flags 0x1005
device sio16 at isa? port 0x178    tty    flags 0x1005 irq 12 vector siointr

```

This is an example of a BOCA 16 port Board Sharing IRQ 12, with a starting address of 100H. Adjust your settings as required. You'll notice that I commented out sio1-3 from the File, but left sio0. I am using a mouse on COM1 and am not using any of the other Serial ports. You can start the BOCA sio numbering at the first free COM Port.

Save the file and rebuild your kernel.

Now we need to edit `/etc/ttys` and add the lines to activate all the extra lines.

Next we create the required devices in `/dev`

Next we edit `/etc/modems` so it initializes all the new Devices at startup.

Last we reboot.

45.4. pppd

Not Yet Scheduled

46. FreeBSD as a LAN Server

46.1. PC Imaging Software (Pcrdist and Samba)

46.1. PC Imaging Software (Pcrdist and Samba)

Not Yet Scheduled