FreeBSD Overview
Comparison with Linux

ccTLD Workshop

June 20, 2006
Samoa

Hervey Allen

Some Practical Matters

- When we install please use the root password supplied in class.
- During the workshop *please do not change the root password.*
- Please do ask questions! Lots of questions! Really -we mean this.
- If you don't understand something be sure you ask for help! This is how you learn.
- Questions?

Outline

- The World of FreeBSD
- FreeBSD 6.1 installation
- FreeBSD disk paritioning
- FreeBSD directory structure (`man hier`)
- How FreeBSD boots (`man boot`)
- Configuring a network interface
- Shutdown and restart the server – runlevels

Outline continued

- How to install software:
  - packages
  - ports
  - source
  - cvs
  - portsnap
- Summary
- More resources

Linux != UNIX

The World of FreeBSD

Start here: http://www.freebsd.org/

- RELEASE (6.1 and 5.5 legacy)
- STABLE ('beta' code)
- CURRENT ('alpha' code)
- Ports
- Packages
- Documentation Project
  - FreeBSD Handbook

Installing FreeBSD (6.1)

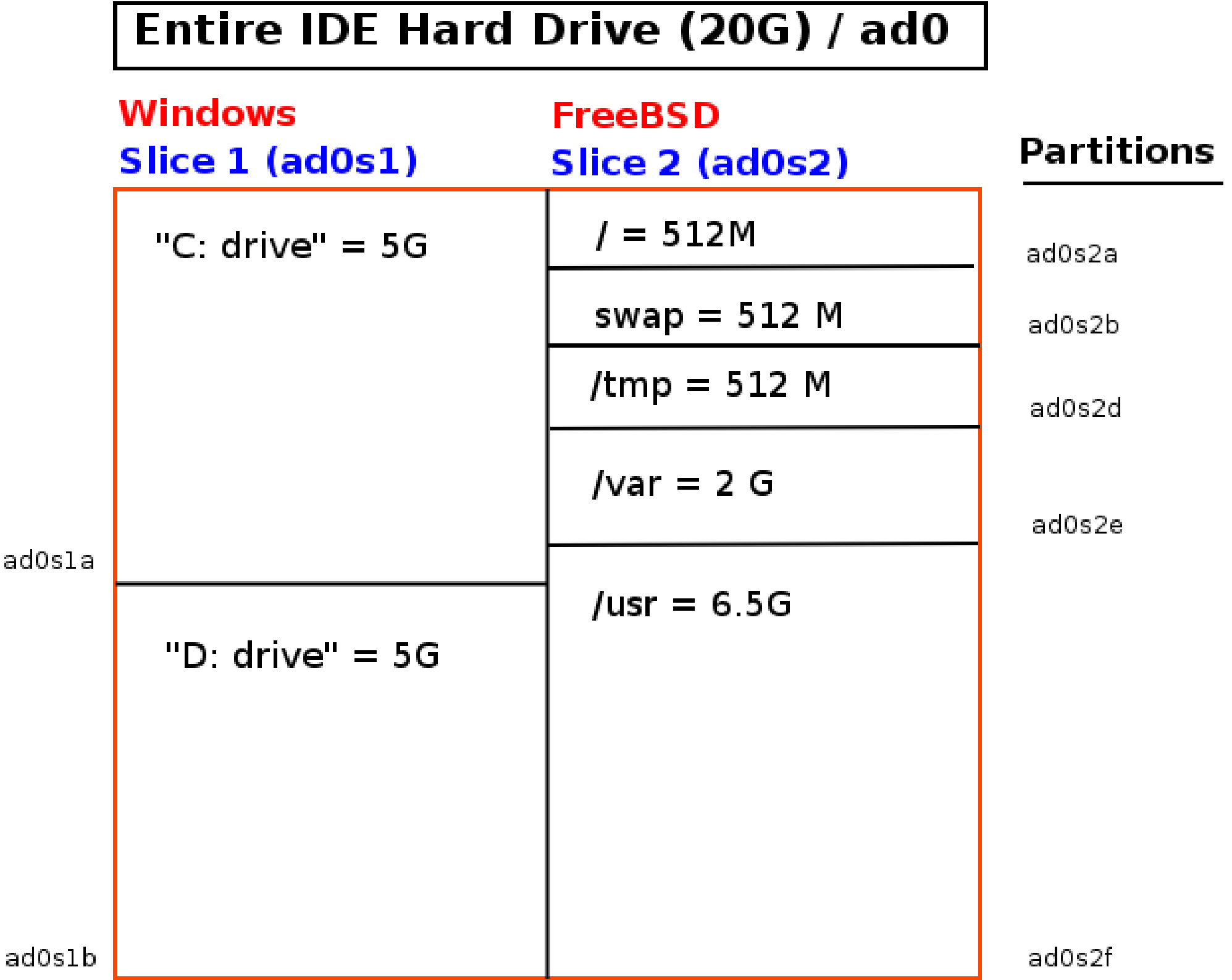
- How can you install? (FreeBSD Handbook section 2.2.6)
  - A CDROM or DVD
  - Floppy disks (including preconfigued install)
  - An FTP site, going through a firewall, or using an HTTP proxy, as necessary
  - An NFS server
  - A DOS partition on the same computer
  - A SCSI or QIC tape
  - A dedicated parallel or serial connection

FreeBSD Disk Organization

If you wish to understand how FreeBSD organizes and views disks then read section 3.5 of the FreeBSD handbook for an excellent and succinct description.

If you come to disk partitioning from a Windows perspective you will find that UNIX (FreeBSD, Linux, Solaris, etc.) *partitions* data very effectively and easily.

In FreeBSD a “slice” is what you may consider to be a “partition” under Windows.


Entire IDE Hard Drive (20G) / ad0
Windows
Slice 1 (ad0s1)
FreeBSD
Slice 2 (ad0s2)
Partitions
"C: drive" = 5G
"D: drive" = 5G
/ = 512M
swap = 512 M
/tmp = 512 M
/var = 2 G
/usr = 6.5G
ad0s2a
ad0s2b
ad0s2d
ad0s2e
ad0s2f
ad0s1a
ad0s1b

FreeBSD Partition Schemes

Partition	Usage
a	Root partition (/)
b	swap partition
c	Not used for filesystems.
d/e/f	/tmp, /usr, /var, etc...

View partition information using “`df -h`” and “`swapinfo`”

FreeBSD Disk Slices

Sample Output to view disk slices from "`fdisk -s`"

```
/dev/ad0: 77520 cyl 16 hd 63 sec
Part          Start          Size Type Flags
   1:            63       8385867 0x0b 0x80
   2:       8385930       8385930 0xa5 0x00
   3:      16771860        208845 0x83 0x00
   4:      16980705      61159455 0x0f 0x00
```

This is a 40GB disk with 3 operating systems spread across four slices. The operating systems include

Windows 2000 (1), FreeBSD (2), DOS swap slice for Windows 2000 (3) and Linux (4).

FreeBSD Partitions in a Slice

You can see more detailed information about your disk slices by just typing "`fdisk`"

To see the partitions in a FreeBSD slice use "`disklabel /dev/DEV`":

```
# /dev/ad1s1:
8 partitions:
#      size        offset      fstype      [fsize bsize bps/cpg]
  a:   524288      0           4.2BSD      2048 16384 32776
  b:   2045568     524288      swap
  c:   122865057   0           unused      0    0      # "raw" part, don't edit
  d:   524288      2569856     4.2BSD      2048 16384 32776
  e:   524288      3094144     4.2BSD      2048 16384 32776
  f:   119246625   3618432     4.2BSD      2048 16384 28552
```

FreeBSD Partitions in a Slice cont.

To view slice partition information in a more "human" readable format use "df -h". This can, however, be misleading. For example:

```
Filesystem     Size    Used   Avail Capacity  Mounted on
/dev/ad1s1a    248M     35M    193M    15%    /
devfs          1.0K    1.0K      0B   100%    /dev
/dev/ad1s1e    248M    526K    227M     0%    /tmp
/dev/ad1s1f     55G    2.7G     48G     5%    /usr
/dev/ad1s1d    248M     42M    186M    18%    /var
/dev/ad1s2      55G     15G     38G    28%    /data
/dev/da0s1     500M    226M    274M    45%    /mnt/flash
```

Use "`swapinfo`" to see the swap partition:

```
Device          1K-blocks     Used    Avail Capacity
/dev/ad1s1b       1022784      124  1022660     0%
```

FreeBSD Directory Structure

Repeat after me:

"The command '`man hier`' is your friend."

So, why is your FreeBSD disk slice split in to partitions? Largely to separate important file systems from each other. These file systems are usually represented by specific directories.

Why not just run with everything in one place? That is, everything under root (/).

- Note: FreeBSD can optimize layout of files based on the use for the file system.

A Few FreeBSD Directories

- Structure of partitions/directories:
  - / (“root”)
  - /usr
  - /var
  - swap
- Two important directories:
  - /tmp or /var/tmp
  - /usr/home

“/” Root

The root partition is where critical system files live, including the programs necessary to boot the system in to “single user” mode.

The idea is that this part of the system does not grow or change, but rather stays isolated from the rest of the operating system.

If you give enough room to /usr and /var, then “/” can be quite small (around 512MB should be safe for now).

The one directory that may grow is /tmp, particularly if you run Linux binaries that use /tmp.

/usr

Is used for system software like user tools, compilers, XWindows, and local repositories under the /usr/local hierarchy.

If one has to expand this partition for additional software, then having it separate makes this possible.

FreeBSD maps user directories to /usr/home.

/var

This is where files and directories that consistently change are kept. For example, web server logs, email directories, print spools, temporary files, etc.

On a server it is a good idea to have /var in a separate partition to avoid having it fill your other file systems by accident.

swap

Swap is where virtual memory lives. Swap is it's own file system.

You can run without swap, and your PC may run faster, but this is dangerous if you run out of memory.

There are several opinions about what is the optimal swap size. This can depend on what type of services you run (databases need more swap). The general rule of thumb is that swap size should be somewhere between your RAM and twice your server's RAM.

/u

Optional file system methodology to consider:

- Can make life easier if you create “/u” at initial build.
- Data is stored in /u, and you can symlink /home to /u/home.
- You can reinstall FreeBSD (the OS) from scratch and leave /u alone.
- Keep backups of /etc/ and /usr/local/etc/ as well as any other directories you need. Consider tar'ing these to somewhere on /u.

How FreeBSD Boots

The init process:

- Refer to Chapter 12 of the Handbook for more information.
- After the kernel boots, which is located in “/” (in Linux it's usually /boot) it hands over control to the program /sbin/init.
- If filesystems look good then init begins reading the resource configuration of the system. These files are read in this order:
  - /etc/defaults/rc.conf
  - /etc/rc.conf (overrides previous)
- Mounts file systems in /etc/fstab

How FreeBSD Boots cont.

The init process cont.:

- Once file systems are mounted then the following starts:
  - Networking services
  - System daemons
  - Locally installed package daemons (/usr/local/etc/rc.d scripts)

Init process and shutdown:

- When `shutdown` is called then init runs the scripts /etc/rc.shutdown.

FreeBSD Password Files

There are four files:

- /etc/passwd

  ASCII password file, with passwords removed

- /etc/master.passwd

  ASCII password file, with passwords intact

- /etc/pwd.db

  db(3)-format password database, with passwords removed

- /etc/spwd.db

  db(3)-form encrypted password database, with passwords intact

FreeBSD Password Files cont.

- /etc/master.passwd has the same functionality as the *shadow* password file under Linux
- Accounts without shells are specified using a shell of `/nonexistent` and the

  `/sbin/nologin` utility for polite login refusal.
- /etc/pwd.db and /etc/spwd.db are *hashed* index files. This means that username lookup is *not linear* as in Linux.

Configuring Network Interfaces

During boot if a NIC is recognized then the appropriate code is loaded to support the NIC (a module).

After boot, using "`ifconfig`" you can see if the NIC exists. Look for MAC address.

Initial NIC configuration can be done with `ifconfig`, or try "`dhclient dev`"

If NIC works, edit /etc/rc.conf and put in device specific entries for each boot.

Configuring Network Interfaces cont.

Example lines in /etc/rc.conf for network device:

```
hostname="localhost.my.domain"
ifconfig_wi0="DHCP"
```

Set the hostname and indicate that NIC "`wi0`" will use DHCP to get network information. FreeBSD uses specific names for each network device. "`wi0`" indicates the first "Wireless" card.

Configuring Network Interfaces cont.

- FreeBSD 6.1 allows you to rename network interfaces as you like, e.g.:
  - `ifconfig em0 name etho`
- Linux users who prefer “eth0” instead of “wi0” could configure this in /etc/rc.conf with:
  - `ifconfig_wi0 = “DHCP name eth0”`
- Some programs, however, expect specific-named network interfaces (gnome wireless applet for one...).

Shutdown and Restart a Server

How do you shutdown a FreeBSD box?

- shutdown 1 message
- halt
- init 0

And, to restart?

- reboot
- shutdown -r now
- init 6

Run Levels

FreeBSD has the concept of run levels:

Run-level	Signal	Action
0	SIGUSR2	Halt and turn the power off
1	SIGTERM	Go to single-user mode
6	SIGINT	Reboot the machine

So, in reality, you either run in single-user mode with “everything off” and just root access (run-level 1), or your system is up and fully running in multi-user mode.

To go from single-user to multi=user mode type “`exit`” at the command line.

Starting/Stopping Services: Review

How does a service start/stop?

- `kill, /etc/rc.d/service stop`
- /etc/rc.d/service start ==> system
- /usr/local/etc/rc.d/script.sh ==> 3rd party
- /etc/rc.conf ==> system & some 3rd party
- /etc/defaults/rc.conf ==> leave alone
- Old school: /etc/rc.local
- Read "`man rc`" *several times*! :-)

Software Install Methods

There are three methods to install software on your FreeBSD system. These are:

1.) FreeBSD packages and the `pkg` utility.

2.) The ports collection /usr/ports.

3.) Installing from source (gcc make).

You are most likely to install from packages, then ports, then from source.

There are advantages and disadvantages to each.

The “pkg” Commands

In general the pkg_add and pkg_delete facilities allow you to install and remove software on your system in an efficient and consistent manner.

The pkg_info command allows you to see what's installed, quickly, and to get detailed information about each software package that is installed.

Package Installation Using pkg_add

- You can get “packages” from local source (a CD), off FreeBSD sites, or your local network.
- To install a package from a CD-ROM:

```
pkg_add /cdrom/dir/package_name
```

- To install from an ftp server you can do:

```
pkg_add ftp://address/dir/package_name
```

Using pkg_info

Find out if something is already installed:

```
pkg_info      (list all installed packages)
pkg_info | grep moz    (find all packages
                        containing “moz”)
```

Get more information about an already installed package:

```
pkg_info name\*

pkg_info -I name\*
```

For example “`pkg_info -I bash\*`” returns:

```
bash-3.1.10_1    The GNU Project's Bourne Again Shell
```

Using pkg_delete

If you have a package you wish to remove you can simply type:

```
pkg_delete package_name
```

But, if you want to remove the package and all its dependent packages you would do:

```
pkg_delete -r package_name
```

But, *be careful* about doing this. You might want to check what will happen first by doing:

```
pkg_delete -n package_name
```

Installing from Ports

First you must have installed the /usr/ports collection during system installation. Otherwise, use `sysinstall` after installation and then choose Configure, Distributions, then Ports.

Once the “ports collection” is installed you can see the entire tree under /usr/ports. There are several thousand (15,000'ish) software packages available.

This collection contains minimal information so that you can “make” a software package quickly, and easily from separate CD-ROMs or a network site containing the port source.

Installing from Ports cont.

To see if a software package exists as a port:

```
cd /usr/ports
make search name=package
make search key=keyword
```

Let's do this for “lsof” (LiSt Open Files):

```
cd /usr/ports
make search name=lsof
```

(or “`whereis lsof`”)

And the output from this is:

```
Port:   lsof-4.76.2
Path:   /usr/ports/sysutils/lsof
Info:   Lists information about open files (similar to
  fstat(1))
Maint:  obrien@FreeBSD.org
Index:  sysutils
B-deps:
R-deps
```

Installing from Ports cont.

From the previous page you'll note that the port is in /usr/ports/sysutils/lsof.

If you have a network connection…

You can simply type "`make install`"

But, you might want to do:

- `make`
- `make install`

To automatically get ports from a local server you can do this by changing a system variable:

- `export MASTER_SITE_OVERRIDE="ftp://local.site/distfiles/ fetch"`

Installing from Ports cont.

You can install from cdrom. If you have a cdrom with the full ports distfiles, then simply mount it. Then you would do:

- `cd /usr/ports/sysutils/lsof`
- `make`
- `make install`

And the port will find the distfile on /cdrom instead of from the internet.

Once a port is installed use "`pkg_info port\*`" to verify its installation, and standard "`pkg`" commands to manipulate the installed port.

CVS and CVSUP

One issue that arises, “How to keep your ports collection up-to-date?”

CVS, or **C**oncurrent **V**ersions **S**ystem, can do this:

First you must install the cvsup-withou-gui package, then you can tell this tool to look on a server that has the latest ports collection and update your local collection with a single command like:

```
cvsup -g -L 2 -h cvsup.freebsd.org \
  /usr/share/examples/cvsup/ports-supfile
```

Portsnap vs. CVS

The `portsnap` utility is a new method for maintaining your ports collection and is in the default system as of FreeBSD 6.0.

`portsnap` downloads a compressed snapshot of the ports tree (approximately 40-45Mb).

Manually you'd run (first time only, then use cron):

- `portsnap fetch`
- `portsnap extract` (first time only)
- `portsnap update`

Read "`man portsnap`" to set this up automatically using cron.

Summary

- Aimed at stability not user desktops.
- Very, very good track record for stability and security.
- Scales to very large sizes for services.
- Massive collection of software (15,000 ports as of June 2006), including the ability to run Linux packages.
- Software can be installed in several ways.
  - FreeBSD `pkg` facility is arguably superior to rpm as it can resolve dependencies.
  - Fedora, Red Hat, others have have largely solved this using `yum`.

More resources

This presentation is located here:

http://ws.edu.isoc.org/workshops/2006/ccTLD-Samoa/day1/ha/freebsd/intro-freebsd.pdf

- http://www.freebsd.org/
- http://www.freebsd.org/support.html
- O'Reilly books (http://www.oreilly.com/)
- http://www.freshports.org/
- http://www.freebsddiary.org/

Additional topics...

...if there's time

The FreeBSD Kernel

- You might rebuild a kernel to add hardware support, additional filesystem support, etc.
- Or, to remove extraneous drivers.
- Kernel source, if installed, is in /usr/src/sys
  - "If there is not a /usr/src/sys directory on your system, then the kernel source has not been installed. The easiest way to do this is by running `sysinstall` as root, choosing Configure, then Distributions, then src, then sys." (FreeBSD Handbook 8.3)
- To rebuild your kernel you use the default configuration file, update settings as needed, then recompile the kernel, installing it in /boot.

Recompiling the FreeBSD Kernel

See FreeBSD Handbook section 8.3

- Config file in `/usr/src/sys/`*`arch`*`/conf`
- Example:
  - `cp GENERIC /root/kernel/MYNEWKERNEL`
  - `ln -s /root/kernel/MYNEWKERNEL`
  - `Edit MYNEWKERNEL file to set options`
    `see /usr/src/sys/arch/conf/NOTES`

After you've edited MYKERNEL for options

- `cd /usr/src`
- `make buildkernel kernconf=MYNEWKERNEL`
- `make installkernel kernconf=MYNEWKERNEL`

Recompiling the FreeBSD Kernel cont.

- Kernel installed as /boot/kernel/*kernel*
- Old kernel is in /boot/kernel.old/*kernel*
- If new kernel does not boot, go to boot loader prompt and type:
  - `unload`
  - `boot /boot/kernel.old/kernel`

Recompiling the FreeBSD Kernel cont.

The kernel config file has many options. For a more complete explanation of the various options see (e.g. on a PC with Intel CPU):

- /usr/src/sys/i386/conf/NOTES
  And, for non-architecture specific notes see:
- /usr/src/sys/conf/NOTES

Or look at the FreeBSD Handbook section 8.4 for some more examples.

Kernel and Hardware Support

FreeBSD is moving towards “modularizing” hardware support. That is “drivers” (kernel loadable modules) are loaded at boot time to support your systems' hardware.

Some hardware is still supported by statically loaded software directly in the kernel.

Some hardware use is optimized by setting kernel state using the sysctl facility.

Kernel Loadable & Static Modules

- Static (in conf) – built-in during recompile vs.
- Kernel loadable (kld) /boot/kernel modules.
- Autoloading using /etc/rc.conf directives and/or using /boot/loader.conf, which overrides /boot/defaults/loader.conf
- Commands `kldload, kldstat, kldunload`