Exercises: Post FreeBSD Install: IP Services Workshop SANOG 9

January 14, 2007

Please do not change the root password on your machine for any reason!

PLEASE: Read everything on this sheet to properly complete these exercises

Your instructor may indicate some updates or changes to these exercises. The initial exercises are to finalize our machine configurations for use later today, and throughout the week, as well as to highlight some areas that are different from Linux - particularly methods by which you can install software.

Exercises

To Be Done by Everyone: Final Configuration

1. Edit /etc/make.conf and /etc/hosts
2. Use sysinstall and pkg_add to install lynx, bash, sudo and joe
3. Installing two more packages (gmake, unzip)
4. Installation of software using ports
5. Use pw to create a new userids that you will use instead of root
6. Setup sudo for privileged commands
7. Start a Caching Nameserver
8. Installing/configuring X11 (Xorg), Gnome and gdm

Some Differences from Linux

9. FreeBSD package (pkg) and ports Systems Revisited (**Recommended**)
10. Process startup and the RC system
11. Disk partitions: how to view your disk
12. Getting help using manpages, docs, and the FreeBSD Handbook
13. Shutdown and reboot your system
14. Initial login, virtual terminals, command line manipulation

For Those who Want More Practice

15. Additional exercises (Ask your instructor for these if you are using paper handouts)

Notes (CRITICAL)

1. The "#" and "$" characters before commands represents your system prompt and is not part of the command itself. "#" indicates a command issued as root while "$" indicates a command issued as a normal user.

2. **rehash:** If you install software, update your environment as root and the change is not immediately available try typing `rehash` at the root shell prompt. This is only necessary when running a C shell (e.g., like /bin/csh).
3. ***italics***: Items that are in *italics* are to be replaced with something of your choice. For instance, *username* means choose your own username, don't literally choose the word "username".
4. **noc:** References to "noc" refer to the classroom server that is used during the workshop to store software, act as a local web server, provide DNS services, etc.

1.) Edit /etc/make.conf and /etc/hosts [Top]

During the workshop you will be using the command to install software via the FreeBSD ports system. We must override a default setting on your system so that FreeBSD will attempt to fetch the source for the port from a local server and not from the default remote site of ftp.freebsd.org.

We can specifiy this by editing the file /etc/make.conf and adding a single line (don't break the line) that will force `make` to look on a local server first. As we have not yet installed any alternate editors you will need to use the VI editor for this exercise. Refer to your VI Reference Sheet if necessary.

First, edit /etc/make.conf (you must be root to do this):

```
# vi /etc/make.conf
```

Scroll down to the last line of the file, press the "o" (lower-case oh) key to insert a line below the last line, and then type in the following:

MASTER_SITE_OVERRIDE=ftp://noc/pub/FreeBSD/ports/distfiles/

To save and exit you must press:

:wq <ENTER>

Now you may note that we only typed in "noc". No IP address and no full domain. So, how will our machine know to find "noc" - Well, either we have added this to a local DNS server that you may be using, or you will give it a specific entry in your /etc/hosts file. This is what we are going to do now.

First, edit /etc/hosts (you must be root to do this):

```
# vi /etc/hosts
```

This file has a lot of text in it if you are used to the Linux format. Scroll down to the line that reads:

```
127.0.0.1            localhost  localhost.my.domain
```

and press the "o" key. Now you are going to enter in an entry for our noc box.

PLEASE NOTE: Your instructor will give you the correct name and IP address during class. What is shown below is only an example.

Now, on the new line enter something like:

```
192.168.0.100         noc   noc.ips.ws.sanog.lk
```

Again, to save and exit you must press:

:wq <ENTER>

Now go on to the next exercise.

2.) Use sysinstall and pkg_add to install lynx, bash, and joe [Top]

For this exercise you need to be logged in as root.

FreeBSD has a large collection of preconfigured binary packages for each release that can be installed in a number of ways, including:

1. Using the post-installation configuration utility `sysinstall` (don't use this now).
2. Directly from the FreeBSD installation CD-ROMs using pkg_add.
3. From a local network collection of FreeBSD packages specifying the network path and protocol using pkg_add.
4. Blindly across the network using the "-r" option of the pkg_add command

The core package utilities available in FreeBSD are:

```
pkg_add
pkg_delete
pkg_info
```

You can read about each of these package commands using the "man" command, i.e.:

```
# man pkg_delete

# man pkg_add
```

Remember, the pkg utilities under FreeBSD are very powerful as they can resolve dependencies, let you test installation before actually installing software, and keep track of what is installed with utilities to query the package database and assocated files.

For our first package we are going to install a text-based web browser called *lynx*. We'll use the sysinstall program to install lynx from the first disc of the FreeBSD CD-ROM installation set.

First,you can see if lynx is already installed on your system. Use the pkg_info command to do this. To get an idea of what packages are installed and what the default pkg_info output looks like type:

```
# pkg_info | more
```

You'll notice a pause while your machine prepares to show all the packages installed, in alphabetical order. Press space to scroll down the list, or ctrl-c to stop the list.

Now to check for just the lynx package type:

```
# pkg_info | grep lynx
or
# pkg_info lynx\*
```

If you don't understand what "grep" is doing type "man grep". In optional exercises we go in to detail about the use of "*grep*", and the pipe facility represented above by the "|" character.

If lynx is already installed, for purposes of this exercise, first remove the package from our system (it's good practice!). You can do this by typing:

```
# pkg_delete lynx-2.8.5_2
```

Now type again:

```
# pkg_info | grep lynx
```

And, assuming that lynx is now not installed you should just get your prompt back.

Now to install lynx using the FreeBSD install CD-ROM do the following:

Insert the FreeBSD Disc 1 Installation CD in your machine's CD-ROM drive.
Be sure that you are logged in as the root user:

Now start the "sysinstall" utility by typing:

```
# sysinstall
```

At this point you take the following steps in the sysinstall menu system to choose the lynx package and install it:

- Choose "Configure" (you can press "C" and ENTER)
- Arrow down to "Packages" and press ENTER
- For installation media choose "CD/DVD" and press ENTER (note you can use network options here)
- If prompted choose "acd0" for your CD-ROM drive
- Scroll down the list of presented categories (using the down arrow key) until you see "www", and press ENTER
- Scroll down to "lynx-2.8.5_2" to highlight this then press the spacebar to choose the package, then press ENTER
- Now choose "Install" and press ENTER to install Lynx version 2.8.5_2 on your system

  At this point you will be prompted for disc 2 of the FreeBSD installation set. Most of the FreeBSD packages have been moved to this disc. Go ahead and follow the on-screen instructions to swap cd-roms.

- Once installation is complete you will be presented with the initial FreeBSD Configuration Menu - Scroll to the top of the screen and choose "X" for "Exit" and press ENTER
- Finally choose "Exit Install" to leave the "sysinstall" utility

After exiting this utility your FreeBSD install CD-ROM may still be mounted. You can type "df" and look for an entry like this:

```
Filesystem  1K-blocks     Used    Avail Capacity  Mounted on
/dev/acd0      660384   660384        0   100%    /cdrom
```

If you don't see the "/cdrom" entry, then type:

```
# mount /cdrom
```

again to make sure your CD-ROM (FreeBSD disc 2) is correctly mounted.

Installation of bash

Now let's install the bash package using a single command. As your FreeBSD CD-ROM is mounted under /cdrom, and as your instructor knows where the bash package resides on your CD, you could install bash by typing:

```
# pkg_add /cdrom/packages/shells/bash-3.1.10_1.tbz
```

But, rather than doing this, let's do this a bit differently. First use the *cd* (Change Directory) command to change to the initial cd-rom directory. Then we'll change to the directory /cdrom/packages/All. The "All" directory contains a list of all packages available to be installed on the cd-rom. It has the advantage that if you use pkg_add installing from this directory that if there are dependent packages that are needed, then they will be found and installed as well. So, first do:

```
# cd /cdrom

# cd packages/All
```

Now let's look for the bash package. We can use the *ls* command to do this with a wildcard. You'll be practicing and learning much more about these basic system commands in a few more exercises. OK, so do:

```
# ls bash*
```

You should see *bash-3.1.10_1.tbz* on your screen. This is how you know the exact name of the package you want to install. In addition, once you have installed and are using the bash package it will be easy to use *command completion* to figure out complicated filenames like these. You'll see more of this in exercise 7. So, now that you know you want to install the package called *bash-3.1.10_1.tbz* (we're going to use the newer version vs. the 2.05b version that's listed), and you are already in the directory where this file resides, then you can simply type:

```
# pkg_add bash-3.1.10_1.tbz
```

and the *bash* package will be installed. By the way, *bash* is an alternate shell environment that you can use which has more features than the default shell *sh*. Notice the file extension of "tbz" on the file bash-3.1.10_1.tbz, this means that the file is actually a group of files that have been compressed and saved as a single file.

Installation of joe

Some of you may practice the use of the "vi" editor in later, optional exercises. vi is not as friendly as the optional "joe" editor. There are reasons for this, which we'll explain. But, for later exercises let's install "joe" right now, but this time we'll install from our local workshop server, "noc".

Note: If the server name is different than what is shown below your workshop instructor will let you know.

To install the "joe" editor from the local workshop server you can simply type the following:

```
# pkg_add ftp://noc/pub/FreeBSD/6.1-RELEASE/i386/packages/All/joe-3.3,1.tbz    (Note
the comma)
```

This will download and install the joe 3.3,1 package to your machine from the local workshop server. Note that this editor has been added by us. It no longer ships with the default FreeBSD installation.

Do a little bit of exploring about what you have just installed by entering the following commands:

```
# rehash
# man lynx (we'll discuss man a bit later)

# man bash

# man joe

# pkg_info lynx-2.8.5_2

# lynx www.freebsd.org

'q' (to 'q'uit lynx)

# joe

ctrl-c (to abort, you can say "no" to saving data)
```

3.) Installing two more packages (gmake, unzip) [Top]

For this exercise you need to be logged in as root.

During the week we will need two additional packages to do our work. The two packages we need are:

- **gmake**: The GNU Make utility.
- **unzip**: List, test and extract compressed files in a ZIP archive.

Both of these items are packages available on the first FreeBSD CD-ROM. We have placed these files on our classroom server in the /pub/FreeBSD/packages directory. So, to install these three packages simply issue the following three commands

```
# pkg_add ftp://noc/pub/FreeBSD/6.1-RELEASE/i386/packages/All/gmake-3.80_2.tbz

# pkg_add ftp://noc/pub/FreeBSD/6.1-RELEASE/i386/packages/All/unzip-5.52_2.tbz
```

If you want to learn more about these three packages you can type:

```
# rehash

# man gmake

# man unzip
```

4.) Installation of software using ports [Top]

For this exercise you need to logged in as root.

The ports collection, which we installed during the FreeBSD install, is larger than the packages collection available on the first FreeBSD install cd-rom. As of June 2005 there were over 13,000 ports available! The FreeBSD Handbook has an excellent discussion of using Ports in section 4.5. In a nutshell these are the key points to remember about ports:

- The ports collection you installed contains descriptive files about how to make a software package and where to get the actual source files. The actual source files (referred to as "distfiles") are not installed.
- You can tell your machine to find distfiles on a local server, or you can get them from CD-ROM, otherwise when you install a port file the software source will be downloaded from the Internet.
- You can use the cvsup tool to keep your ports collection up-to-date. That is, all the descriptive files of where to find source, version numbers, etc.
- Once you have installed a software package using ports it is possible to "deinstall" the package.
- There is a tool called portupgrade which will find all installed ports and keep them up-to-date.
- If you have a ports distfiles cd-rom it needs to be mounted as /cdrom for the ports utilities to find the distfiles.

We are now going to install a utility that can be useful to look at network processes.

- **lsof**: LiSt Open Files. Shows information about files opened by processes.

First do the following:

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

This will output something like:

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

From this you can see that the lsof utility resides in /usr/ports/sysutils/lsof. So, to install lsof on your system you do the following:

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

Make will download the lsof source (hopefully from our local server!), and then compile the source. "Make install" will place the compiled binary files in the appropriate directories on your system and update the appropriate configuration files if necessary. You can issue the single command "`make install`" instead.

You can now type:

```
# rehash       (to re-read items in your directory path)
# man lsof
```

For more information. Finally, if you want to deinstall a port once it is installed, for instance lsof, you would do the following (please don't do this):

```
# cd /usr/ports/sysutils/lsof
# make deinstall
```

And, if you decided that was a mistake, you can now do the following to reinstall a port (after it's been installed once):

```
# cd /usr/ports/sysutils/lsof
# make reinstall
```

Also, note that the lsof port has been built and installed and will now appear as a package. You can type:

```
# pkg_info lsof\*
```

You will see that the lsof port now appears as an installed package. Thus, you could manipulate this software with package commands (`pkg_info`, `pkg_delete`).

If you do a "make clean" then "make reinstall" will no longer work for that port. If you want to see `lsof` in action you can type:

```
# rehash
# lsof -i
```

In addition, FreeBSD installs the utilities `fstat`, `sockstat`, and `netstat` that can display similar information. You can review this topic in more detail by doing exercise 9 of this worksheet.

5.) Use pw to create a new userid that you will use instead of root [Top]

First login to your computer if you have not already done so. Login as userid "root" using the password given at the start of class and that you used while installing FreeBSD.

Now that you are root you can create a new user account on your machine. If more than one person is using your machine, then be sure that you create an account for each and every person.

To create or remove a user account you should use the "pw" command. To get a feel for the power and complexity of this command take a look at its man pages:

```
# man pw
```

So, first pick a username that you want to use. For example, use your first name, last name, a combination of both, or whatever you prefer. For purposes of this example we use *username*. So, to add a new user to your

system type:

```
# pw useradd username -m -s /usr/local/bin/bash
```

This creates the user *username*. The "-m" says add a new home directory of /usr/home/*username* and copies files from /usr/share/skel to the new user's home directory. The default shell "bash" will be used. Note that at this point the bash shell has not been installed on your machine. We'll be doing this in the upcoming exercises.

Next you need to set the password for the new userid, otherwise you won't be able to use the account properly. To do this use the `passwd` command like this:

```
# passwd username
```

You will be prompted to enter in a new password, then to enter it again to verify. You need to pick a secure password. Note, as root (OK, and by default on FreeBSD!) you are allowed to pick any password you want, but you need to pick a secure password for your new account. Here are some quick guidelines to picking a secure password:

- Password should be at least 6 characters in length.
- Don't use any words in a dictionary in any language - forwards or backwards.
- The password should contain a mix of letters, numbers, and upper and lower-case letters.

For example, something like "1ps4vIce5!" is OK ("IP Services!") isn't bad. Please don't pick this for this exercise.

So, finally, let's change your new user's password using the command:

```
# passwd username
```

Please complete these steps:

Create the userid *admin*. This account may be used later for ssh exercises, and to allow your instructor access to your machine during the week if necessary. The password for the *admin* should be written at the front of the room. If not, please ask your instructor for the admin password now. The `pw useradd` command format will be a bit different. This will be explained in exercise 5.

```
# pw useradd admin -m -s /usr/local/bin/bash -G wheel
# passwd admin        (password for admin account given in class)
```

6.) Using sudo for privileged commands [Top]

When you created your own user account *username* you did not place it in the wheel group. This means that you cannot become the root user from your *username* using the command *su*. This is a useful thing to be able to do if you don't want to have to logout and log back in each time you need to be root. In addition, the change we are going to make to the "sudoers" file, which controls access to the *sudo* command is based on your *username* being in the wheel group. So, the first thing you need to do is be sure you are logged in as

root:

```
$ exit

Login: root
```

Now we are going to use the *pw* command to make a change to your *username* account make it a member of the wheel group. The wheel group is a special group in Unix that allows members to become root. To do this type:

```
# pw usermod username -G wheel
```

The "-G" indicates "group list". If your user already belongs to other groups you *must* include all these groups in list format (e.g. "group1,group2,group3,etc.), otherwise your user will be *removed* from the other groups when you issue this command. To see what groups a user belongs to you can type:

```
# groups username
```

Once your user is a member of the wheel group you can use the su (Substitute User identity) command to become another user without having to login and logout of your session.

To practice this you need to login as your *username* account now:

```
# exit

$ Login: username
```

Now you can practice using *su*.

```
$ su - root
```

If you skip the "-" option, then you won't execute the root login scripts. In general you can type "su user", or "su - user", and switch to any user's environment.

Now let's drop back to your standard user account and try to do something that requires privileges:

```
# exit

$ less /etc/master.passwd
```

You should get the message, `"/etc/master.passwd: Permission denied"` - Now try running the same command like this:

```
$ sudo less /etc/master.passwd
```

You noticed that this did not work, right? First we need to add a package called "sudo" to your machine, then we need to configure the file /etc/sudoers to allow you to use this facility. Let's get the package installed first. We'll use pkg_add and we'll get it from our local server:

```
$ su - (you must be root to install this package)
```

pkg_add ftp://noc/pub/FreeBSD/6.1-RELEASE/all/packages/All/sudo-1.6.8.12_1.tbz

Now we need to configure the file /usr/local/etc/sudoers so that you can use the `sudo` command:

```
# rehash
# visudo
```

This is a special utility to edit the /etc/sudoers file using vi, but to lock the file so that no one else can edit it while you are using `visudo`. So, remember your vi commands and scroll down the screen until you find the following lines in the file (if you need help with vi ask your instructor or one of the classroom helpers):

```
# Uncomment to allow people in the group wheel to run all commands
#%wheel         ALL = (ALL)       ALL
```

(Note/hint: there are other ways of finding this line in vi. You could search for something using "/".).

Now, once you are here, just make one, small change. Remove the "#" symbol (it's a comment symbol) from in front of "%wheel". Once you do this save and quit from the file. So, your file should now look like:

```
# Uncomment to allow people in the group wheel to run all commands
%wheel          ALL = (ALL)       ALL
```

And, to save and quite it's:

```
:wq
```

That's it. You are now ready to run "sudo" as a general user (or, at least a general user in the wheel group). Make sure you exit from the root shell you are in:

```
# exit
```

And, now as a general user, try issuing the command:

```
$ sudo less /etc/master.passwd
```

You will be prompted for a password, and the first time you use sudo you'll receive a warning. The password you should enter will be your user account's password. This is different from Linux. Type this in, and you should be able to view the /etc/master.passwd file even though you are running as a standard user. Try doing this again ("less /etc/master.passwd") and you'll notice that you are no longer prompted for the root password. You can issue any privileged command (pretty much) without needing a password at this point. Once you login and logout again, then you'll have to enter a password the next time you use sudo.

sudo is very useful to allow you to do system administration tasks on your machine without needing to be logged in as root. This can help to protect you from making mistakes running as root, which can be costly.

7.) Start a Caching Nameserver [Top]

While this is not strictly necessary, it is likely to make your life much easier. In addition, if you are running a server box, but not planning to run it as a nameserver, then you may not want to rely on a third party to provide you with reliable DNS. If you use an ISP that has slow response time on DNS queries, then using a caching nameserver can help to speed up your network queries. Note, this adds another services, thus another

security vector to consider. This is a tradeoff you will need to decide upon. It, also, means you should be prepared to update your DNS software (BIND) in case of security issue with the version you are using.

To start a caching nameserver is really quite simple. First you must edit the file /etc/rc.conf to tell FreeBSD that you plan on using this service. If you do not do this, then you cannot start the server.

Edit the file /etc/rc.conf (you must be root to do this):

```
# vi /etc/rc.conf
```

Go to the last line in the file and press the "o" key. On the new line enter in the following text:

named_enable="YES"

To save and exit you must press:

:wq <ENTER>

Now at the command line type:

```
/etc/rc.d/named start
```

You'll see some error messages on the screen. Don't worry, these indicate we have not completely setup DNS as expected, but you will be doing that later this week.

And, that's it! Now each time you boot your machine DNS resolution will take place locally. Let's see if things are working. Try the following:

```
# ping noc
# ping sanog.org
```

Did these both work? If not let one of your instructors know to help you fix any problems you may have.

8.) Installing/configuring X11 (Xorg), Gnome and gdm [Top]

For this exercise you will need to be logged in as root.

Before we begin there are a few items to understand to put this in perspective:

- A much more complete discussion of using X and Desktops under FreeBSD can be found in Chapter 5, The X Windows System, of the FreeBSD Handbook. I *highly* recommend reading this entire chapter if you wish to fully configure your X Window environment for FreeBSD.
- FreeBSD is a server-based operating system. Configuration of X, display managers, and desktops like Gnome and KDE are not done during installation as this is not the objective of the OS.
- You can configure FreeBSD to autostart in to a GUI environment, but I would not recommend this, except, possibly, for desktop and laptop use.
- To automate the use of X Window you can edit /etc/ttys (see the Handbook), xinitrc, .xinitrc, .xsession, Xsession, and KDE config files to your liking. For our purposes we are going to install Gnome and

manually call the Gnome Dislpay Manager (gdm) when we wish to use an X Window environment.

- X11 is the implementation of the X Windows environment. Xorg is the X11 implementation that FreeBSD uses by default. You can still use XFree86 if you wish, but configuring Xorg is easier. Gnome and KDE are Desktop environments that include Window managers and entire sets of desktop applications.
- You can clean up and make your font environment considerably nicer under X, particularly fonts used by OpenOffice, Netscape/Mozilla/Firefox, and The Gimp. This is easier to do using Xorg. Read Chapter 5 of the FreeBSD Handbook for more details.
- The Gnome and KDE installations that come with the FreeBSD 6.1 CD-ROM installer are quite up-to-date (as of June 2006). The Gnome version is 2.12.3 and KDE is version 3.5.1. You can read about Gnome on FreeBSD here:

  http://www.freebsd.org/gnome/

  You can read about KDE on FreeBSD here:

  http://freebsd.kde.org/

- If you plan on installing both Gnome and KDE and/or possibly offering up Desktop environment via your network from a server, then you will most likely use the KDE Display Manager (kdm) or, possibly the X Display Manager (xdm). kdm can offer access to both Gnome and KDE Desktop environments at log in.

For this workshop we are going to install Gnome from your FreeBSD CD-ROM install set and we are going to use gdm manually to start our X Window environment. We have already configured an Xorg configuration file for your particular machines setting the correct monitor refresh rates and setting your desktops to use 24-bit color at a depth of 1024x768. You will be copying this file from our classroom server, noc, to the appropriate location.

During our initial installation of FreeBSD we installed Xorg, but we have not yet configured it to run on our hardware. First, let's install Gnome using the `sysinstall` utility.

Start the `sysinstall` utility (as root) by typing:

```
# sysinstall
```

At this point you take the following steps in the sysinstall menu system to choose the Gnome packages and install them:

- Choose "Configure" (you can press "C" and ENTER)
- Arrow down to "Packages" and press ENTER
- For installation media choose "CD/DVD" and press ENTER (note you can use network options here)
- If prompted choose "acd0" for your CD-ROM drive
- Scroll down the list of presented categories (using the down arrow key) until you see "gnome", and press ENTER
- Scroll down the list to "gnome2-2.12.3" to highlight this then press the spacebar to choose the package, then tab to OK, then press ENTER

You should have noticed that multiple Gnome packages are automatically chosen as dependencies (a "D" appears in the selection boxes vs. an "X") when you choose just gnome2-2.10.0.

- Now choose "Install" and press ENTER to install Gnome version 2.12.3 on your system. If prompted, choose OK to the dialogue asking you if you are sure.

  At this point you will be prompted for disc 2 (if you started this exercise using disc1) of the FreeBSD installation set. Most of the FreeBSD packages have been moved to this disc. Go ahead and follow the on-screen instructions to swap cd-roms.

  If there are problems with this let your instructor know. There are many possible bits and pieces to this that can go wrong. An easy work-around is to download the entire gnome folder from noc/pub/FreeBSD/packages/All to your local drive and then use pkg_add in the Gnome directory to install everything at once.

- Once installation is complete you will be presented with the initial FreeBSD Configuration Menu - Scroll to the top of the screen and choose "X" for "Exit" and press ENTER
- Finally choose "Exit Install" to leave the "sysinstall" utility

At this point both Xorg (the X11 XWindow system) and Gnome are installed on your system.

To get them to work there are quite a few variations of what might happen and how you can do this. If Xorg is able to correctly identify your video and monitor hardware you can do the following to configure your XWindow server and Gnome:

```
# Xorg -configure
# cp /root/xorg.conf.new /etc/X11/xorg.conf
# /usr/X11R6/sbin/gdm
```

You may notice that we skip a step (`X -config /root/xorg.conf.new`). This is on purpose. If additional configuration is needed your instructor will have done this and will help you in class as necessary.

Once you log in using Gnome (log in as the user "admin" or as your *username*, but not as root) you may need to manually set the screen resolution to be something more reasonable. You can usually do this by going to the Screen Resolution utility under the Desktop menu, then the Preferences menu.

Some Differences from Linux

9.) FreeBSD package (pkg) and ports Systems Revisited [Top]

If you are used to systems like rpm, apt-get, yum, portage, etc., then you'll probably enjoy using the FreeBSD software packaging system. First the relationships:

- Packages are ports that have been precompiled and configured and are wrapped up in a single, compressed file.
- The pkg system (pkg_info, pkg_add, pkg_delete, etc.) installed packages. It can resolve dependencies,

show you what will be installed, remove a package, or all dependent packages, etc.
- If you install the ports collection in /usr/ports/ this is a series of directories divided by category that each contain a descriptive set of files of the software packate to be installed.
- Ports are compiled. When you decide to compile a port the source is retrieved and the software is compiled. If you chose to do a `make install` of a port, then a package is created and installed.
- Since ports are installed like packages you can remove them using the pkg system, or you can use the `make deinstall` make directive to remove a port.
- There are many more ports than packages.
- You can configure ports as needed using command line directives with the `make` command.

If you know the generic name of a piece of software you can install it quickly by typing:

```
# pkg_add -r name
```

For instance to install rsync on a machine you'd just type:

```
# pkg_add -r rsync
```

and this would get the latest version of rsync in package form from the default package server. This is not always an efficient method to install if your internet connection is slow.

If you want to know if a package has been installed you can type commands like this:

```
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\*
$ pkg -I bash\*
```

To remove a package use `pkg_delete`. Here are some examples:

```
# 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
```

And this simulates for you all files that will be removed.

Here is an example of how you can use the ports system:

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). A Linux utility not installed by default on FreeBSD:

```
# 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
WWW:    http://people.freebsd.org/~abe/
```

So, to make and install lsof on your system you would do:

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

You should have already done this earlier, so both commands should complete very quickly.

You can join the two `make` commands and just type "`make install`" if you wish.

To prove that the lsof port is now installed as a package type:

```
$ pkg_info lsof\*
```

For more information you can read /usr/share/docs/handbook/ports.html and the man pages for the various `pkg` commands. There are quite a few:

- pkg_add
- pkg_check
- pkg_create
- pkg_deinstall
- pkg_delete
- pkg_fetch
- pkg_glob
- pkg_info
- pkg_sign
- pkg_sort
- pkg_version

10.) Process startup and the RC system [Top]

This is a rather complex topic. But, in a nutshell, under FreeBSD 6.1, when your machine boots processes (that is daemons or services) are configured and/or started like this:

- Items that are configured or referenced to start in the file /etc/defaults/rc.conf contain corresponding script entries in /etc/rc.d/.
- You can override settings in /etc/defaults/rc.conf by placing settings in /etc/rc.conf.
- Every script in /etc/rc.d/ will attempt to run, but if it does not see:
  script_enable="YES"
  in /etc/defaults/rc.conf or /etc/rc.conf then the script will not run to start the corresponding service.
- Some third party software will place entries in /etc/rc.conf and corresponding third party startup script packages in /usr/local/etc/rc.d/.
- Some third party packages will place startup/config scripts in /usr/local/etc/rc.d/ that do not look in any file, but simply execute upon system boot.
- Some older software may still place startup items in /etc/rc.local, but this has been deprecated, even though it is still supported.
- Generally third party software with scripts in /usr/local/etc/rc.d/ with the execute bit set will automatically run:

  ```
  /usr/local/etc/rc.d/script-name.sh start
  ```

  at system boot time. They may, or may not look in /etc/rc.conf for configuration settings.
- *Always* read the contents of any new third party script placed in /usr/local/etc/rc.d/ to understand how it works and if you need to edit /etc/rc.conf for the script to run properly.

To get a better feel for this you should read:

```
$ man rc
```

Then, you should probably read this again...

Now, if you want to start a process each time your machine boot you generally add an item to /etc/rc.conf to indicate this. For instance, if you wanted to enable the ssh daemon (server) each time your machine started then you would add the line:

```
sshd_enable="YES"
```

to /etc/rc.conf. If you look at /etc/defaults/rc.conf you'll see that sshd is not enabled in this file by default. By enabling sshd in /etc/rc.conf this overrides the setting in /etc/defaults/rc.conf. In addition, if you look in /etc/rc.d/ you'll find an sshd script file that starts this service.

To see how sshd is enabled initially do this:

```
$ grep sshd /etc/defaults/rc.conf
```

You should see something like:

```
sshd_enable="NO"                # Enable sshd
sshd_program="/usr/sbin/sshd"   # path to sshd, if you want a different one.
sshd_flags=""                   # Additional flags for sshd.
```

These are the lines in /etc/defaults/rc.conf that deal with the ssh daemon. When you specified to enable the ssh daemon during installation then in the file /etc/rc.conf the lines that read:

```
sshd_program="/usr/sbin/sshd"   # path to sshd, if you want a different one.
sshd_flags=""                   # Additional flags for sshd.
```

became active. So, if your ssh program was not "/usr/sbin/sshd" for some reason, then in /etc/rc.conf you could add:

```
sshd_program="/new/directory/sshd"   # new path to sshd
```

to override what's in /etc/defaults/rc.conf.

You could just put everything in /etc/defaults/rc.conf, but you don't want to do this. If you upgrade your system it's almost certain that /etc/defaults/rc.conf could be overwritten. In addition, this file is large and it would be hard to see the changes you had made if you were to do them in /etc/defaults/rc.conf.

To start a service manually you can use it's startup script by hand. For instance, try typing:

```
$ su - (you must be root to run these commands)

# /etc/rc.d/sshd
```

What is returned on the screen? It should be something like:

```
Usage: /etc/rc.d/sshd [fast|force|one](start stop restart rcvar keygen reload status poll)
```

So, you could type

```
# /etc/rc.d/sshd status
```

to see if ssh is running. If it is, then try:

```
# /etc/rc.d/sshd stop
```

to start the service. Now type:

```
# /etc/rc.d/sshd start
```

to restart the service. Note the startup script option "reload" - This would let you make changes to the ssh configuration file(s) and then reload the service without actually stopping it so that it reads the new configuration. Note that already connected clients would not see this new configuration change until the logoff and log back in again.

Finally, and this is not obvious, if a startup script in /etc/rc.d has not been enabled in /etc/defaults/rc.conf or /etc/rc.conf, then even if you manually invoke the script it will not run. You will not get any indication of this other than the service not starting (i.e. use "*ps auxw| grep servicename*" and you won't see it started).

At this point take a closer look at /etc/rc.d/

```
# ls /etc/rc.d
```

and use "man" to read about some of the services. If you want to try and start and stop some of these services

feel free to do so now, but remember you'll need to add 'servicename_enable="YES"' in /etc/rc.conf for the service to start.

11.) Disk partitions: how to view your disk [Top]

First, make sure you are logged in as your user and not as root.

Now in a terminal lets look at the partitions. Type:

```
$ df
$ df -h
```

What difference did you see between "df" and "df -h". How can you see what your swap contains (note it was not listed using "df")? Use this:

```
$ swapinfo
```

If you want to see more detailed information about your disk slices you can use the "fdisk" command. As a general user you are not allowed to run this program, so you must use sudo. Try this by typing:

```
$ sudo fdisk
```

Be careful with fdisk as you can remove slices, partitions, etc.

If you are interested in how much space files are taking up in a directory or a group of directories you can use the "du" command. Try it out by typing:

```
$ du

$ du -h
```

As usual you can get more information by typing "man du".

Now that you've seen these methods for viewing disk information, you should be able to understand the output of the *disklabel* command a little better. The *disklabel* command is really the proper way to view the status of partitions in your FreeBSD slice at it will show you all your FreeBSD slice information no matter if a partition is mounted or not. In some cases this is critical if you are trying to troubleshoot problems and not all partitions have mounted. So, if FreeBSD resides on the slice /dev/ad0s1 on your disk, then to see disk information you can type (as a normal user): `$ sudo disklabel /dev/ad0s1` (Note, we are assuming we are using IDE drives) If we are This shows you output along these lines (will be different for this workshop):

```
# /dev/ad0s2:
8 partitions:
#        size   offset    fstype   [fsize bsize bps/cpg]
  a:   409600        0    4.2BSD     2048 16384 25608
  b:  1433600   409600      swap
  c:  8385930        0    unused        0     0         # "raw" part, don't edit

  d:  1843200  1843200    4.2BSD     2048 16384 28552
  e:  3072000  3686400    4.2BSD     2048 16384 28552
  f:  1627530  6758400    4.2BSD     2048 16384 28552
```

As you can see it includes the "c" partition and "b" or the "swap" partition as well as filetypes.

12.) Getting help using manpages, docs, and the FreeBSD Handbook [Top]

Now that you have FreeBSD up and running you probably want to have a way to figure out how to use it when there are no instructors around, or other FreeBSD-knowledgeable people. First and foremost, make it a habit to read the man pages (MANual pages) for the commands that you use. You might be surprised at some of the things these commands can do! In any case, as you have seen, when in doubt about what a command does, or how it works simply type:

```
$ man command
```

Optionally, you might find some additional information for some commands typing:

```
$ info command
```

And, to get information about both these commands type:

```
$ man man
```

```
$ info info
```

After this there is a large amount of documenation available to you in serveral ways. For instance, if you look in:

/usr/share/doc

you will find multiple FreeBSD articles in various languages available to you. In addition, the FreeBSD Handbook is available here under /usr/share/doc/handblook. If you wanted to start reading the FreeBSD Handbook from your local har drive you could either point a web browser to the file /usr/share/doc/handbook/index.html, or you could type the command:

```
$ lynx /usr/share/doc/handbook/index.html
```

Go ahead and type this command now to get a feel for what information is available to you in the FreeBSD Handbook. Now let's look at the FreeBSD FAQ by typing:

```
$ lynx /usr/share/doc/faq/index.html
```

After this, have a look at some of the available articles by doing:

```
$ cd /usr/share/doc/en/articles
```

```
$ ls
```

Finally, there are several papers available as well. Try:

```
$ ls /usr/share/doc/papers
```

If you have a network connection, then you can go to `http://www.freebsd.org/docs.html` for even more

information. Become accustomed to the idea of using man to get specific information about commands, and then using these additional resources to get an overview of entire sub-systems of the FreeBSD operating system. If you want to read the FreeBSD Handbook online it is available here `http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/`. If you want to understand FreeBSD concepts in more depth, then the FreeBSD Handbook is *really* where you should begin.

13.) Shutdown and reboot your system [Top]

For this exercise you need to be root. It is better to close open files and programs (for example Firefox, vi, etc.), but it is not necessary. Before continuing read the man pages for shutdown, init, halt, and reboot (you'll see they are all connected):

```
# man shutdown
# man init
# man reboot
# man halt
```

Now, in a terminal do the following (save data, etc. as this will immediately reboot your machine!):

```
# shutdown -r now
```

Now your machine is rebooting. The "-r" stand for "reboot" and the "now" meant to take this action "now". This takes a moment. To stop your machine entirely you can use the command:

```
# halt
```

Or, you can also change your run level to run level 0, which is the same as "halt". So, you would write:

```
# init 0
```

And, to reboot this is the same as init 6, or:

```
# init 6
```

If you are running something like gdm for a graphical login prompt on your machine you can usually use provided menu choices to reboot or shutdown. The thinking is that once you have this level of access, then you can simply turn off the machine's power if you wish. At the very least it is much more friendly to use a software interface to shutdown or reboot than pulling the power as processes have a chance to clean up, save data, etc.

Note, sometimes it is useful to bring your machine down to runlevel 1, or "single user mode". For instance, if you are running X Windows and want to shut it down quickly (you'd really only do this on a desktop machine, by the way!), then you can open a terminal window as root (or use "su"), and then type"

```
# init 1
```

This will shut down X Windows, networking, and quite a bit more. Now you are in "single user mode". To get back to "multi-user mode" you simply type:

```
# exit
```

This exits your single-user modem shell and tells the system to go back to multi-user mode. Notice that the "runlevel" concept under FreeBSD is different than under Linux. The major difference are:

- Linux uses runlevels 1, 3, and 5 (single user, network with no GUI, network with GUI [X]).
- FreeBSD uses runlevel 1, single user mode. To get back to network mode with or without GUI you `exit` runlevel 1 (or you can reboot).
- You can use `init c` if you with to block further logins under FreeBSD.
- Both FreeBSD and Linux use runlevels 0 and 6 in the same manner (halt and reboot).

14.) Initial login, virtual terminals, command line manipulation [Top]

The first time you login on your system after installation you will be presented with a prompt that looks something like this:

```
FreeBSD/i386 (name.domain) (ttyv0)

login:
```

At this point you can enter in "root" and, when prompted, the password we gave you in class for the root account. Once you are logged in you can work with an additional 7 virtual terminals if you wish. Actually, you can login from any virtual terminal you want at any time. To do this simply press:

ALT-FN

With "FN" being anything from the F1 to the F8 key. By default you are in ttyv0 which corresponds to the ALT-F1 keyboard sequence. Go ahead and login and then press:

ALT-F2

and login again. Feel free to do this on F3, F4, F8, etc. as you wish. You can cycle through each terminal session easily. This is an extremely useful technique when you wish to do more than one thing at the same time, but you are not using a graphical interface. Since we are loading the mouse daemon (mouse support) you can, also, copy and paste text between your virtual terminals. To do this go back to your initial login terminal by pressing:

ALT-F1

Try typing in the command:

```
$ clear
```

and pressing ENTER. Note that your screen clears and goes to the top. Now, you can easily recover your last command by pressing the UP-ARROW key on your keyboard once. Get used to doing this as it can be very useful if you have entered in a long command and made a mistake. You can press UP-ARROW to get the command back, then you can use the LEFT-ARROW to move your cursor to where the mistake is and correct it, then simply press ENTER to reissue the command. Note, you *do not* need to go back to the end of the command line before pressing ENTER.

In any case, you've pressed UP-ARROW once and should have the command "clear" visible. Now take your mouse and highlight just the command using the left mouse button. Without doing anything else with your mouse now press:

ALT-F2

to get to one of your other terminal sessions. Now just press the *middle* mouse button once. What happened? The text "clear" should have pasted on to your command line. Now you can press ENTER to execute the command. You can use this same trick to copy and paste in to editor windows, long and complex commands, between applications in a graphical environment, etc.

One final useful tip when working in your terminal sessions. As you type each command it is being saved in to a file called your "history file". This has a very useful purpose. Go back to your original login terminal pressing:

ALT-F1

and type the command:

```
$ history
```

You should see a list of commands you have entered in earlier. This list is probably short and it will even include incorrect commands you may have typed in. To quickly and immediately recover and execute a prior command make note of the number in the left-hand column next to the command and then just type:

```
$ !N
```

Where N is the number. So, if "clear" had been the second command, and you typed in:

```
$ !2
```

Then clear would appear on the command line very quickly and immediately execute. During the week you are going to be typing in some long and complex commands. The use of history can save you considerable time. If you press the UP-ARROW key repeatedly you can scroll through the previous commands you have entered beginning with the last command. Give it a try.

One *very useful* trick when using XWindow. If you need to escape your graphical environment and go to a console for some reason you can use the same virtual terminals, except that you press the keyboard combination:

ALT-CTRL-F1 through F8

At this point you will exit your graphical environemnt (KDE, Gnome, etc.) and be presented with a text console login prompt. Your graphical environment is still running as you left it. To get back to your graphical environment press:

ALT-CTRL-F9

Why might you need to do this? Here's one scenario:

You've just installed your favorite MTA (email server/Mail Transport Agent) and you typed something like (don't do this now):

```
pw usermod username -G mail
```

to place your user in the "mail" group. What just happened if your user was also a member of the "wheel" group and this is how you are using `su` and `sudo`? You can no longer become root in your session. To fix this you could go to a virtual console window, log in as root, and reissue the command above, but this time correctly:

```
pw usermod username -G wheel,mail
```

To fix this problem.

[Return to Top]

Hervey Allen

Last modified: Sat Jan 13 17:09:14 IST 2007