Exercises: FreeBSD Introduction: AfNOG 2004

May 16, 2004

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

Exercises

1.) 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.

Now that you are root you can create a new user account on your machine. You might not use the same machine tomorrow, so do not worry about keeping this account.

First we will have you create an account using the interactive mode of the "adduser" command.

At the prompt type:

```
adduser
```

Now you will be prompted for information. Below is a sample of how you should respond. Feel free to choose your own username and password, but keep all other options *exactly* the same:

```
bash-2.05b# adduser
Username: test
Full name: Testing Account
```

```
Uid (Leave empty for default):
Login group [test]:
Login group is test. Invite test1 into other groups? []: wheel
Login class [default]:
Shell (sh csh tcsh bash nologin) [sh]: bash
Home directory [/home/test]:
Use password-based authentication? [yes]:
Use an empty password? (yes/no) [no]:
Use a random password? (yes/no) [no]:
Enter password: yourPassword
Enter password again: yourPassword
Lock out the account after creation? [no]:
Username   : test
Password   : *****
Full Name  : Testing Account
Uid        : 1002
Class      :
Groups     : test wheel
Home       : /home/test
Shell      : /usr/local/bin/bash
Locked     : no
OK? (yes/no): yes
adduser: INFO: Successfully added (test) to the user database.
Add another user? (yes/no): no
Goodbye!
```

Now you have a user called test (in this example) that belongs to the "wheel" group. This means that you will be able to execute root-privileged commands from your account using the command "sudo" and you will be able to "become" the root user using the command "su", but you must konw the root password, in both cases, to be able to do this.

The faster way to create a user account is to simply use the "pw useradd" facility. For example, to do what we did above you could do by simply typing:

```
pw useradd username -m -G wheel -s
/usr/local/bin/bash
passwd username
```

Following the prompts to set the user’s password when you use the "passwd" command. By the way, pick a secure password...

Now that you have a new user account log out of the root account (type "logout" or "exit") and login to your computer using your new account.

Sample of this:

```
# exit
Login: test
Password:
bash-2.05b$
```

Remember you can remove a user by using the "rmuser" command and you can update a user’s account (or the user themselves can do this) using utilities like chpass, chsh, and chgrp. As an administrator you will probably be most interested in the "chpass" and "rmuser" commands. Be sure to do:

```
    man chpass
    man rmuser
```

To understand how these commands work.

2.) Practice with basic commands [Top]

Be careful in this exercise. Running as root means that you can easily damage your system. If you are not sure of a command ask the instructor or helpers before continuing. You should be sure you have logged out as root and logged in using your new userid.

The first command that we are going to use is "man", this is short for "man"ual. Read about each command to see the range of options that exist. Do the following:

```
    man cp
    man cd
    man ls
    man mv
    man rm
```

After do the same, but now use "info" like this:

```
    info cp
    info cd
    info ls
    info mv
    info rm
```

If you have problems exiting from "man" press the "q" key. Also, you can use the keyboard arrows to move around in the descriptions.

Now we are ready to practice a bit with the commands:

```
    cd /
    ls
    ls -la
    cd /tmp
    cd ..
    cd tmp
```

What’s going on here? If you don’t understand, ask.

```
touch text.txt
cp text.txt new.txt
mv text.txt new.txt
```

What’s happening now? If prompted to overwrite, respond "y". Note that "userid" is the name of the user account you created in the first exercise.

```
cp text.txt /home/userid/.
cd ../home/userid
```

Do you understand that you can do "cd /home/userid", or "cd .." and after "cd home/userid" or "cd /home, then cd userid", and in the end you’ll arrive to the same directory?

Now play with the use of the tab key. For example, in /home/userid start to type the first part of the command "cp text.txt text.txt.bak" - then, type:

```
cp te
cp text.txt te
cp text.txt text.txt.bak
```

The tab key makes life much easier. Now type:

```
mkdir tmp
mv text.* tmp/.
ls
```

Finally, we are going to remove the directory that contains the two archives.

```
cd tmp
rm *
cd ..
rmdir tmp
```

You can force this using a command like this:

```
rm -rf tmp
```

The use of "rm -rf" is **very dangerous!**, and, naturally, very useful. For example, if you are "root" and you type "rm -rf /*" this would be the end of your server. This commands says "remove, forcibly and recursively, everything" - Or, if you start in the root directory (/), remove all files and directories *without asking* on the entire server. If you want to use "rm -rf *" always take a deep breath and check where you are first (really, do this!):

```
pwd
```

First this says in what directory you are. If you are mistaken, then you have the opportunity to not remove files that you might really need.

3.) Practice with more commands [Top]

As you can see from the presentation there are many, many commands that can be used under FreeBSD. If you go to the directories /bin, /usr/bin, /sbin, /usr/sbin, /usr/local/bin, etc. you can see hundreds of files that are programs - many of these are user and system commands. The /bin directory has the critical commands for the operating system. The directory /sbin has, in general, commands that only root runs, or that root can run to make changes (like /sbin/ifconfig).

Please run these commands as logged in as your new user and not root.

To start, do the following:

```
cd /sbin
ls
```

And, now read about some of these commands. For example:

```
man dmesg
```

With care play with the commands listed in the presentation, and/or with commands you find in the directories mentioned. I *strongly* recommend that you read about any command before trying to run it (i.e. "man command").

4.) Create a file and use vi to edit the file [Top]

Now we are going to open an exmpty file and write something in it. The vi editor works with two modes; one to input data, and the other to give commands. This is **fundamental** to using vi - You must go in to input mode using a command first, such as i (input) or o (for a newline after the cursor ready for input). To get out of input mode and be able to issue command you must press the ESCape key. Now let’s do the following:

```
cd /root
touch temp.txt
vi temp.txt
```

Now you are in vi. Press the "i" key to switch to input mode.

Type something like, "VI is great! I think I'll be using vi from now on instead of Microsoft Word."

Press to add lines. Type some more stuff, whatever you like.

Now, remembering the following:

```
Open: vi fn, vi -r fn, vi + fn, vi +n fn, vi +/pat fn
Close: :w, :w!, :wq, :wq!, :q, :q!
Movement: h,j,k,l, w, W, b, B, :n
Editing: i, o, x, D, dd, yy, p, u
Searching: /pattern, ?pattern, n, N
```

vi Cheat Sheet

Open:

vi filename	(fn=filename)
vi -r filename	Recover a file from a crashed session
vi + filename	Place the cursor on last line of file.
vi +n filename	Place the cursor on line "n" of file.
vi +/pat filename	Place cursor on first occurrence of "pat"tern

Close:

:w	Write the file to disk. Don't exit.
:w!	Write the file to disk even if read/only.
:wq	Write the file to disk and exit.
:wq!	Write the file to disk even if read/only and quit.
:q	Quit the file (only if no changes).
:q!	Quite the file even if changes.

Movement:

h	Move 1 space backwards (back/left arrow).
j	Move down 1 line (down arrow).
k	Move up 1 line (up arrow).
l	Move 1 space forwards (forward/right arrow)
w	Move cursor to start of next word.
W	Same as "w".
b	Move cursor to start of previous word.
B	Same as "b".
:n	Go to line number "n" in the file.

Editing:

i	Enter in to input mode.
o	Add a line below cursor and enter in to input mode.
x	Delete character (del key in some cases).
D	Delete line from right of cursor to end of line.
dd	Delete entire line.
u	Undo last edit or restore current line.
p	Put yanked text before the cursor.
yy	Yank current line.

Searching:

/pattern	Search for "pattern" in the file going forwards.
?pattern	Search for "pattern" in the file going backwards.
n	Find the next occurrence of pattern found forwards.

```
N                      Find next occurrence of patter found backwards.
```

Play with moving around. Move your cursor to a line with text y see what happens when you go in to command mode (ESCape) and use "w" or "W" or "b" or "B" - remember, to get in to command mode press the ESCape key.

Now press "/" and type a word that is in your document, then press . What happens?

Do the same, but press the "?" key at first. Use ESCape to start in command again again if necessary.

To save your file press the ":" key and next type "w" and enter . .

To exit and save do:

    :wq

To exit and not save anything (lose all changes you have made since the last save) do:

    :q!

But, try to save your file for later use. Practice saving, exiting, opening a file in vi again, etc.

5.) Searching for more information about your system [Top]

If you want to see the contents of a file there are three typical ways to do this:

    cat
    less
    more

The "less" command has more functionality, but does not work with all files. The command "cat" will show the contents of a file in all cases. The "more" command is like cat, but pauses after each page. .

Test this using the three commands using them with an informational file like:

```
cd /etc
cat motd
more services
less services (you can exit with "q")
```

Try looking at some more files, for instance, fstab, rc.conf, termcap, etc. If you don’t understand what you are looking at, then use the "man" command. For example, type:

```
    man fstab
    man rc.conf
    man termcap
```

If you have any questions ask the instructor or one of the class helpers.

6.) Using sudo [Top]

When you created your test user you placed it in the wheel group. This means you can use the su (Substitute User identity) command to become another user without having to login and logout of your session, and you can use the sudo command to execute privileged commands from your standard user account. Let’s practice this a bit. First from your standard user account use su to become root:

```
    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 will be prompted for a password - this is the root password. 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.) Install the lynx web browser using pkg_add [Top]

Now we are going to add the lynx text-based web browser using the pkg_add command and our local network server.

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
```

If you don't understand what "grep" is doing type "man grep". You'll see that lynx-ssl is already installed. For purposes of this exercise we will first remove the package from our system (it's good practice!). You can do this by typing:

```
pkg_delete lynx-ssl-2.8.5
```

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 directly from our noc box in T1 you can simply type:

```
pkg_add
ftp://84.201.31.1/home/ftp/pub/pkgs/lynx-ssl-2.8.5.tbz
```

Now see if it appears in your package database:

```
pkg_info | grep lynx
```

You could, of course, download the package file "lynx-2.8.5.tbz" using ftp first, and then just install the package directly from your own machine.

Test lynx out by typing:

```
lynx noc
q (to exit lynx)
```

8.) Permissions and files* [Top]

*Reference: Shah, Steve, "Linux Administration: A Beginner's Guide", 2nd. ed., Osborne press, New York, NY.

If you look at files in a directory using "ls -al" you will see the permissions for each file and directories. Here is an example:

```
drwxrwxr-x    3 hervey   hervey       4096 Feb 25 09:49 directory
-rwxr--r--   12 hervey   hervey       4096 Feb 16 05:02 file
```

The left column is important. You can view it like this:

```
Type User    Group World Links  owner  group  size   date   hour  name
d    rwx     rwx   r-x   3      hervey hervey 4096   Feb 25 09:49 directory
-    rwx     r     r     12     hervey hervey 4096   Feb 16 05:02 file
```

So, the directory has r (read), w (write), x (execute) access for the user and group. For world it has r (read) and x (execute) access. The file has read/write/execute access for the world and read only access for everyone else (group and world).

To change permissions you use the "chmod" command. chmod uses a base eight (octal) system to configure permsissions. Or, you can use an alternate form to specify permissions by column (user/group/world) at a time.

Permissions have values like this:

```
Letter  Permission   Value

R       read         4
W       write        2
X       execute      1
```

Thus you can give permissions to a file using the sum of the values for each permssion you wish to give for each column. Here is an example:

```
Letter  Permission                    Value

---     none                          0
r--     read only                     4
rw-     read and write                6
rwx     read, write, and execute      7
r-x     read and execute              5
--x     Execute                       1
```

This is just one column. Thus, to give all the combinations you have a table like this:

```
Permissions  Numeric      Description
             equivalent

-rw-------   600          Owner has read & execute permission.
-rw-r--r--   644          Owner has read & execute.
                          Group and world has read permission.
```

-rw-rw-rw-	666	Everyone (owner, group, world) has read & write permission (dangerous?)
-rwx------	700	Onwer has read, write, & execute permission.
-rwxr-xr-x	755	Owner has read, write, & execute permission. Rest of the world has read & execute permission (typical for web pages or 644).
-rwxrwxrwx	777	Everyone has full access (read, write, execute).
-rwx--x--x	711	Owner has read, write, execute permission. Group and world have execute permission.
drwx------	700	Owner only has access to this directory. Directories require execute permission to access.
drwxr-xr-x	755	Owner has full access to directory. Everyone else can see the directory.
drwx--x--x	711	Everyone can list files in the directory, but group and world need to know a filename to do this.

Now lets practice changing permissions to see how this really works. As a normal user (i.e. don’t login as root) do the following:

```
cd (what does the "cd" command do when you do
this?)
echo "test file" > read.txt
chmod 444 read.txt
```

In spite of the fact that the file does not have write permission for the owner, the owner can still change the file’s permissions so that they can make it possible to write to it:

```
chmod 744 read.txt
```

Or, you can do this by using this form of chmod:

```
chmod u+w read.txt
```

The forms of chmod, to add permissions, if you don’t use octal numbers are:

chmod u+r, chmod u+w, chmod u+x
chmod g+r, chmod g+w, chmod g+x
chmod a+r, chmod a+w, chmod a+x

Note that "a+r" is for world access. The "a" is for "all", "u" is for "user", and "g" is for "group".

Now, change the file so that the owner cannot read it, but they can write to the file...

```
chmod u-r read.txt
```

Or, you can do something like:

```
chmod 344 read.txt
```

You probably noticed that you can use the "**-**" (minus) sign to remove permissions from a file.

Finally, there is a concept that when you execute a file you normally execute it using the permissions of the user who does this. Por example, if the user "carla" types "netstat", the netstat programs runs with their privileges. But, if you want netstat to always run with the permissions of the owner or of the group of the netstat program, then you can configure the "SetUID" or "SetGID" bit. You can do this using chmod. However, remember that this can be a bad idea from the viewpoint of security...

To do this add a "4" to the chmod octal to set the SetUID, or a "2" to set the SetGID bit.

As an example, you could do:

```
    chmod 4755 /usr/bin/netstat
```

Naturally you would need to be root to do this, or you would have to use the "sudo" command.

And, to set the SetGID bit it would be:

```
    chmod 2755 /usr/bin/netstat
```

After you do "sudo chmod 4755 /usr/bin/netstat" the permissions on the file would look like this:

```
-rwsr-xr-x    1 root     kmem       106344 Feb 23  16:42 /usr/bin/netstat
```

Note the "s" en the owner column.

And, after you issue the command ""sudo chmod 2755 /usr/bin/nestat" the permissions look like this:

```
-rwxr-sr-x    1 root     kmem       106344 Feb 23  16:42 /usr/bin/netstat
```

Not that we would necessarily recommend that you do this... But, it is good to understand this concept. So, if you did this, then to unset the SetGID and/or the SetUID bit you can issue chmod like this:

```
    chmod 0755 /usr/bin/netstat
```

and now you would see:

```
-rwxr-xr-x    1 root     kmem       106344 Feb 23  16:42 /usr/bin/netstat
```

9.) Commands - programs - shell - path [Top]

For this exercise we want you to run as a user other than root. So, if you are root do this:

```
    su - user
```

What do you think the "-" does? (hint: "man su", and we talked about this earlier)

When you type a command or the name of a program the system looks for something with that name using in the directories specified in your PATH environmental variable. Or, if the command is a built-in shell program (such as "cd", see "man builtin"), then it will execute the command without needing to use the PATH variable. To see your what your PATH is set to, do this:

```
    printenv PATH
```

The PATH variable is configured when you login to your account in the file ".profile" in your home directory.

To see how this works let’s create a shell script that will run a simple command, but which resides in a directory outside your PATH statement ("user" here is the name of your own userid).

```
    cd /home/user
    mkdir scripts
    cd scripts
    vi hello.sh
```

Now in the file add these lines:

```
#!/bin/sh
#

echo hello
```

Remember to save and exit the file (:wq). And, to ensure that you can execute the file use the command:

```
    chmod u+x hello.sh
```

Remember that this is using chmod to set the eXecutable bit for the user only.

Now we are going to add the /home/user/scripts directory to our login profile PATH statement.

```
    cd /home/user
    vi .profile
```

Look for the line that reads (more or less - could be different on your machine):

PATH=/sbin:/bin:/usr/sbin:....:$HOME/bin; export PATH

and change it so that at the end of the PATH statement you add:

PATH=/sbin:/bin:/usr/sbin:....$HOME/bin:$HOME/scripts; export PATH

Save the file and now do the following:

```
hello.sh
. .profile
hello.sh
```

What just happened? You changed the PATH statement to include /home/user/scripts, but when you tried to run the script in /home/user/scripts it didn’t work. This was because you had not actually updated the PATH variable in your shell. When you did ". .profile" you executed your user profile again, which updated the PATH variable with your new PATH variable. You can verify this by typing "printenv PATH" again.

You may have noticed the "$HOME/bin" item in the PATH. As you can see FreeBSD has the concept that you may wish to have your own personal bin directory for executables, so normally the "hello.sh" script might reside in /home/user/bin, but for purposes of this exercise we used /home/user/scripts.

Finally, if you want to change something like the PATH for everyone you can do this in two ways. One, you could update /etc/profile with a new PATH statement. This means that everyone on your system will see this change as soon as they login the next time. Or, you can change /usr/share/skel/dot.profile so that all new accounts have the new PATH, but previous users will not see this change. In both cases changes like this should not be done lightly. When setting up a server with many users you will probably want to think about what directories your users need to have in the PATH from the beginning and update /usr/share/skel/dot.profile before creating initial user accounts.

In addition, you can run the "hello.sh" script by typing "/home/user/scripts/hello.sh" at any time.

To finish up we are going to change how the "rm" command runs to make it "safer" (in my opinion). Here are the steps:

```
vi /home/user/.profile
```

Go to the end of the file and type an "o" (add a line after the cursor and put you in to input mode). Then type:

```
alias rm=’rm -i’;
```

Now exit and save the file (:wq). After that type:

```
touch temp.txt
rm temp.txt
. .profile
touch temp.txt
rm temp.txt
```

And, what happened? Now the "rm" command asks you before you erase a file if you are sure you want to do this. If you don't like this you can remove the alias in the .profile, re-run .profile, and leave things as they were. Note, you can always just use "rm -f" to force remove files and skip the prompt. My advice is to leave the "rm" command in interactive mode - you are likely to be very thankful for this at some point in the future.

10.) Processes and removing them [Top]

If you would like to see what is running on your system, then you use the "ps" command (ProceSs). For example, to see everything running on your system for all users, and even items running that are detached type:

```
ps -aux
```

If you find something that you wish to stop (maybe your web browser session has hung), then you can look for the process ID number, and you can issue a "kill" command to stop the process. This is a *very* powerful feature of UNIX. If you need to kill a process for another user then you must have root privileges to do this. So, if for example you had a Netscape browser (Mozilla) running that had stopped responding, then you could open a terminal window, and type:

```
ps -aux | grep mozilla
```

or, if you know that it's running under your userid you can just type "ps" or "ps | grep mozilla". Now, once you have the process ID number you type:

```
kill nnnn
```

Where "nnnn" is the number of the process you wish to stop. If you try this and the process will not close after, say, 15 or 30 seconds, then you can issue the stronger command:

```
kill -9 nnnn
```

But, *be careful* this shuts down the process without giving it any chance to save data, remove lock files, or generally clean up. I.E. you could lose or corrupt data.

If you see something running that you do not want to have run each time you boot, then you can, generally, edit /etc/rc.conf and override the setting that starts this service (see /etc/defaults/rc.conf). You can stop the service immediately by using the "kill" command. If you are testing a configuration file for a known running service (say the Apache web server) you can, often, tell the service to restart reloading the configuration file, but to reload using the same parameters it originally used to start. This can be very useful if the service requires a complex set of parameters to restart. The command to do this is:

```
kill HUP nnnn
```

Try using the ps command. See if there is anything you can stop by using the kill command. Note, you could cause all sorts of interesting behavior if you do this with running services that you need. I suggest doing this exercise as your standard user, not root, and then starting some program, finding it using "ps -aux | grep progname" and then using kill to stop it.

Finally, some programs spawn many processes when running. Web browsers are an example of this. It can be time-consuming to kill each process one-by-one until you have completely shut down the program. An alternate command is "killall" which will kill processes by name. Naturally you have to be a bit careful with this as you could shutdown something unexpectedly, but if you have 20 processes all called "Mozilla" that you want to stop, then you can simply type:

```
killall mozilla
```

I suggest reading the man page ("man killall") about this command before using it regularly.

11.) Using /etc/hosts [Top]

If you look at /etc/hosts (cat /etc/hosts) you’ll see that the name "localhost" and "localhost.localdomain" are associated with the address 127.0.0.1. This is setup by default. In some cases it can be useful to manually give a name to an IP address, particularly when you are on a network without a DNS server (say, for a workshop...). As a truly artificial example of this let’s just give a different name for our NOC. For instance, right now you can type "ping noc" or "ping noc.ws.afnog.org" and this will work. The noc box is IP address 84.201.255.1. Let’s give noc a couple of different names in /etc/hosts to see how this works. Open the file /etc/hosts (using vi), and after the line with 127.0.0.1 add the following:

```
84.201.255.1     alt-noc           alt-noc
84.201.255.1     another-noc       another-noc.ws.afnog.org
```

Note that you do not even need to type a domain name for an entry. Naturally, none of this scales to even a moderate number of machines. Can you think why?

Now try the following:

```
lynx 84.201.255.1
```

and you should see the main web page on the noc. Now let’s access this several other ways by doing:

lynx alt-noc

Then "q" to exit from lynx

lynx another-noc

lynx another-noc.ws.afnog.org

As we said, this was an artificial example, but you can see how you could connect a few PCs on a private network, give them an IP addressing scheme, and then you could give them names as well in /etc/hosts to make your life easier when working with them. Things like this happen in the real world - for instance, a rack with machines that compute on data, but are not on a public network. Or, a group of machines at a trade show (say in a large kiosk) that are not on a public network and there is no DNS running.

12.) Shutdown and reboot [Top]

For this exercise you need to be root. It is better to close open files and programs (for example Mozilla, 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:

```
shutdown -r now
```

Now your machine is rebooting. 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.

13.) Disk partitions [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 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 partitions, etc.

If you are interested in how much space files are taking up in a directory or a group of directory 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".

14.) Modules [Top]

This is short, but just so that you know you can manipulate dynamically loaded modules in FreeBSD. To see what is loaded type:

```
kldstat
```

You could remove a module by using "kldunload". You can load a module (typically they are in /boot/kernel, /boot/modules, /modules) using "kldload". You can update the module search path using "kldconfig"

In our case there is probably nothing you can unload, but this is good thing to understand. You can type "kld" and tab to see the various kld commands. Then type "man kldxxxx" to read about the details of using each command. That is, look at these commands in more detail like this:

```
man kldconfig

man kldload

man kldstat

man kldunload

man kldxref
```

Hervey Allen
May 2004