Exercises: Unix Bootcamp: AfNOG 2006 Workshop, Nairobi

May 7, 2006

Exercises

Note: 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.

Note 2: 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).

1.) Practice with basic filesystem and user commands [Top]

Be careful in this exercise. Running as root means that you can easily damage your system, so we ask that you log out of your root account and log in as your own user account instead.

If you are not sure of a command ask the instructor or helpers before continuing.

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. We've already been using *man*, but now we'll practice some more:

Many of the basic commands we'll be practicing are built in as part of your shell environment (that is you won't find a binary file for *cd*). To read about commands like *cp, cd, ls, mv, rm* in more detail you can just type:

```
$ man builtin
```

And, for a command like *ls* you can type:

```
$ man ls
```

And, even for a built-in command you can just type "man commandName", or something like:

```
$ man cd
```

and this will open the "builtin" man page for you.

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

As we move around directories an extremely useful command is *pwd*, which return the working directory name you are in. So, if you get lost just type:

```
$ pwd
```

We'll do this from time to time as we use directory commands.

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

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

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

```
$ cd (take you back to your home directory)
$ pwd
$ 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.

```
$ touch text.txt (the previous "mv" command deleted "text.txt")
$ cp text.txt /home/username/.
$ cd ../home/username
```

Now play with the use of the tab key. For example, in /home/*username* start to type the first part of the command "cp text.txt text.txt.bak" - then, type: (Note: means press the TAB key)

```
$ cd (to return to our home directory)
$ cp te >tab<
$ cp text.txt te >tab<
$ cp text.txt text.txt.bak
```

The tab key makes life much easier. Now type:

```
$ cd
$ 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
```

To find out how much disk space is in use you can use the command:

```
$ df
```

If you type this command as:

```
$ df -h
```

"-h" for "human readable" format, then you'll see sizes in in units such as K (Kilobytes), M (Megabytes), G (Gigaytes, etc. Read up on this command, as usual, by typing:

```
$ man df
```

To see information about a directory or a directory and all of its subdirectories you can use the Disk Usage command, `du`. Let's see how much space is used up in the /boot directory and all of its subdirectories:

```
$ cd /boot
$ du -h
```

Try running `du` without the "-h" option as well to see the difference. Type:

```
$ man du
```

for more information.

If you are interested in seeing who's logged in to your system and what they are doing you can use the `w` command. Give it a try:

```
$ w
```

Ask your instructor or class helpers if you don't understand the output of any of these commands.

2.) General job control (ctrl-c, ctrl-z, bg) [Top]

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

When you wish to stop an active process in your shell you can use the keyboard combination "ctrl-c". If this does not work, then you may want to try "ctrl-z". The ctrl-z sequence will suspend the process allowing you to place it in the background using the `bg` (BackGround) command. Generally this command is most useful if you are running in a graphical and have started a service that has locked you out of a shell window. Here are some examples to try to demonstrate these concepts:

```
# cd /
# ls -R (starts to recursively show contents of all directories on your system)
press CTRL-C (aborts the output)
```

In addition, when you type "man command" you can press the "q" key to exit from viewing the man pages. The same goes for "less" as well.

Now let's start a process in your shell to requires you to use ctrl-c to end the process:

```
# tail -f /var/log/messages
```

In this example you are viewing the end of the main log message for your system with data appended to the output as the file grows. That is, if a message is generated and placed at the *end* of the log file you will see this message interactively appear on the screen. This is a very useful tool when you are trying to debug problems on your system. You can open one terminal window (as root, or with an account that can run privileged commands), type "tail -f /var/log/messages", start a process in another terminal window and then view the end results in the window where the *tail* command is running.

Now to end the "tail -f" process press ctrl-c to abort the output. In this case "q" or "ESC" will not work. You need to know about ctrl-c to do this.

3.) 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/index.html`. If you want to understand FreeBSD concepts in more depth, then the FreeBSD Handbook is *really* where you should begin.

4.) 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
```

Each of these commands has it's own features:

- **cat:** or conCATentate a file. By default to the screen. The cat command will display the contents of almost any file, including files that are "open". In some cases if you use more or less, file contents will not be properly dispaly. In addition, cat displays a file without first clearing your screen - something that can be annoying about less and more if you just want to look in a small file and keep what's on the screen visible.
- **less:** Lets you scroll back and forth while displaying a file. Let's you search (like in vi) while seeing the ocntents of a file. Pauses after each screen of information.
- **more:** is similar to less, but with less functionality. Basically more simply pauses after each screen of information and you can scroll by pressing the space-bar for each new screenful of information.

the typical saying is to remember that "less is more" when it comes to Unix.

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

```
$ cd /etc
$ cat motd
$ more services (youcan exit with "q")
$ 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
```

Finally, there is one other useful command for looking inside files. First make sure that you are root:

```
su -
```

Now, as root, type:

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

This will show you the last few lines of your main logfile on your system. This can be *really* useful if you just want to see what the *last* thing written to a file was. Your file /var/log/messages will get very large over time, so if you used cat, less, or more to view this file for new messages this could become very time-consuming. Now, even more fun is to do the following:

```
# tail -f /var/log/messages
```

Now press ALT-FN (say "F3") to go to one of your virtual terminals. Login as root on this terminal. Now go back to your original terminal where you typed the command `tail -f /var/log/messages`. You should see a message on the screen saying that root just logged in. The "-f" option means, "output appended data as the file grows" - or, you can watch each new item as it's written to the end of a file. This is incredibly useful when you are trying to debug problems and you need to see what happens in your logfiles in real time.

Don't forget to log out of your other terminal window.

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

5.) Processes and stopping them [Top]

For this exercise please be sure you are logged in as root.

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 auxw
```

The options to the `ps` command mean:

a: Display information about all user processes, including your own.
u: Display the following process information: user, pid, cpu, mem, vsz, rss, tt, state, start, time, and command used.
x: Modifies the "a" option to include information for detached processes as well.
w: Use 132 columns to display information vs. just your window size. Critical to include actual command that was used to start the process, particularly when using `ps` with the `grep` command. Use "w" twice to display entire width regardless of size.

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 privileges to do this, usually root. The example we are going to use is to kill an account login in another terminal. We'll be going back and forth from one terminal to another and we'll use tail, /var/log/messages, and ps for this exercise. First, let's start an interactive *tail* command of our /var/log/messages file:

```
# tail -f /var/log/messages
```

Now switch to a second terminal and log in again as root:

```
ALT-F2

Login: root
```

Switch back to your original login terminal where the interactive *tail* command is running by pressing ALT-F1. What do you see. You should see something indicating that root logged in on terminal ttyv1. More specifically the entry will look basically like this:

```
Jan  10 17:21:05 localhost login: ROOT LOGIN (root) ON ttyv1
```

OK, now break out of the *tail -f* process by pressing "CTRL-C". Now let's find the process ID of this new login session. To do this type:

```
# ps auxw| grep login
```

If you had just typed "ps aux" you would have seen all processes and then you may have figured out that the "login" process of your other root login can be found by looking for the login keyword.

Now we are going to kill the login shell of your other root login on ttyv1. But, how do you know which process to kill. You probably saw something like this when you did "ps auxw| grep login":

```
root   1060  0.0  0.3  1612 1308  v0  Is    17:22PM   0:00.03 login [pam] (login)
root   1061  0.0  0.3  1612 1308  v1  Is    17:22PM   0:00.03 login [pam] (login)
```

These look almost identical, but note the process IDs. One is 1060 and the other is 1061. This should be your clue. Your *first* root login will have a lower process ID number than your next root login. In addition, if you had waited and started other processes before logging in your second root login, then the process ID numbers might be separated by more than 1. So, in this case you will want to stop the process ID 1061. **Note:** Naturally on your system the actual process ID numbers will almost certainly be different. Now, to stop process UD 1061 do this:

```
# kill 1061
```

Now go back to the other terminal and see if the process has been ended. That is press ALT-F2 and see if the "`Login:`" prompt appears. If it does, then you just forced the user off the system. If the "`Login:`" prompt does not appear, then you may need to use a more forceful *kill* command in the form:

```
# kill -9 1061
```

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.

In general, if you do:

```
# ps auxw | more
```

And you see items 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 auxwl 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.

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

We are going to open an empty file and write something in it.

The vi editor uses "modes"

This is a critical point. The vi editor has two modes. These are:

- Command mode
- Input mode

To go back and forth between these modes when you are in vi you can press:

- ESCape key (command mode)
- Letter "i" for Input mode, letter "o" for input mode with newline below cursor

Remember this as it is confusing. The easiest thing to do when you get confused in vi is to press the ESCape key a couple of times and start over.

Now let's do the following:

```
    $ cd /home/username
    $ 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 ENTER to add lines. Type some more stuff, whatever you like.

Here is a short list of vi commands:

```
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
```

OK, let's save the file that you are in. To do this do:

```
Press the ESCape key to get in to command mode
```

Press ":" to get ready to issue a file command

Type "w" and press ENTER to save your file.

Press ":" to get back to the prompt to issue a file command

Press "q" to quite the file

Instead of the multiple steps you could have type ":wq" to write and quite at the same time. If you need to quit a file without saving it *after* you've made changes, then you press **:q!**. For many people this is the most important command to remember in vi :-).

Below is a more complete vi cheat sheet. In addition you will be receive a vi summary book as part of the book package for this workshop.

```
                        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:

A                       Move to end of line, change to insert mode.
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.

Copy/Cut and Paste
nyyp                    Copy n lines to buffer, paste below cursor
nyyP                    Copy n lines to buffer, paste above cursor
nddp                    Cut n lines and copy to buffer, paste below cursor
nddP                    Cut n lines and copy to buffer, paste above cursor
```

Now let's copy a large file to your home directory so that you can play around with some more vi commands. We'll copy over your /etc/defaults/rc.conf file for this exercise. To do this we do:

```
$ cd
$ cp /etc/defaults/rc.conf .
```

Now let's edit this file. To do this type:

```
$ vi rc.conf
```

Play with moving around. Move your cursor to a line with text and 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 ENTER. 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.

Feel free to open this file again and practice using the vi commands listed above.

7.) Basic network configuration [Top]

To view the status of your network interfaces you use the command `ifconfig`. If you have an ethernet card, then, in general, you will see two network devices when you type the command:

```
$ ifconfig
```

Here is a sample output from this command on a laptop that uses a Lucent wireless card:

```
lo0: flags=8049 mtu 16384
        inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
```

```
            inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
    wi0: flags=8843 mtu 1500
            inet6 fe80::202:2dff:fe5e:86fb%wi0 prefixlen 64 scopeid 0x2
            inet 196.200.222.247 netmask 0xfffffe00 broadcast 196.200.223.255
            ether 00:02:2d:5e:86:fb
            media: IEEE 802.11 Wireless Ethernet autoselect (DS/2Mbps)
            status: associated
            ssid AfNOG 1:AfNOG
            stationname "FreeBSD WaveLAN/IEEE node"
            channel 1 authmode OPEN powersavemode OFF powersavesleep 100
            rtsthreshold 2312 protmode CTS
            wepmode OFF weptxkey 1
```

There are two devices listed here, "lo0" (loopback 0), and "wi0" (wireless 0). FreeBSD references network interfaces by the driver that is loaded to communicate with the device. Thus the "wi0" device uses "wi" driver, which in this case supports a Lucent Technologies wireless card. The "lo" or "loopback" interface is a special interface that is always present. It is a virtual network interface that is only on your computer.

You can use the ifconfig command to bring "down" a network device, to make it active, to change the IP address associated with the device, the netmaks, the SSID for wireless cards, and so forth. There are many, many options and capabilities for the `ifconfig` command. You should read:

```
$ man ifconfig
```

to understand what is possible.

Substitute "rl0" with your network card name as appropriate. And, before making changes note down what your current IP address and netmask are so that you can reset your machine's network configuration. Do not worry if you can't do this at the end of the exercises. Simply ask an instructor for help. It is an easy thing to fix and a good way to learn.

If you wanted to assign an IP address to the "rl0" (RealTek driver) network card and give it a netmask of 255.255.255.0 (or a /24 network), then you could do this (you need to be root to change settings on a network interface):

```
$ su -
# ifconfig rl0 inet 192.168.1.100 netmask 255.255.255.0
```

Optionally you could do:

```
# ifconfig rl0 inet 192.168.1.100/24
```

If you wish to make this permanent each time you boot your machine, then you should edit the file /etc/rc.conf (as root) and add an entry like this:

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

```
# bring up the rl0 network interface
ifconfig_rl0="inet 192.168.1.100 netmask 255.255.255.0"
```

Once you've done this you can type a command like:

```
# /etc/rc.d/netif restart
```

to reinitialize the network with the new parameters.

Finally, if you are on a network that uses DHCP to obtain network addresses you can use the following tool to obtain an address for a network interface quickly:

```
# dhclient rl0
```

And, if you want your machine to boot and automatically get a network address via DHCP for an interface, then you edit the file /etc/rc.conf and add the following entry:

```
ifconfig_rl0="DHCP"
```

If you wish to completely stop network and/or restart networking you can use the `netif` script in /etc/rc.d/ (as root). For instance, if you type:

```
/etc/rc.d/netif stop
```

your network interfaces ("lo0" and "rl0" for instance) will stop. You can type:

```
/etc/rc.d/netif start
```

to restart them. You may have noticed the use of "0" (zero) in the interface names. This is how you can distinguish between two identical network cards if you have a machine with two (maybe your machine is acting as a firewall or gateway). In this case you might have two RealTek-based cards in your machine. You would then have network interfaces "rl0" and "rl1".

8.) 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.

Here's one scenario where virtual terminals might be useful...

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.

9.) 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 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).

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

This is a rather complex topic. But, in a nutshell, under FreeBSD, 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.

[Return to Top]

Hervey Allen

Last modified: Sun May 7 01:59:43 EAT 2006