Linux System Administration and IP Services

Initial Ubuntu System Administration

Notes

* Commands preceded with "$" imply that you should execute the command as
  a general user - not as root.
* Commands preceded with "#" imply that you should be working as root using
  "sudo"
* Commands with more specific command lines (e.g. "RTR-GW>" or "mysql>")
  imply that you are executing commands on remote equipment, or within
  another program.

1. Find out what's installed

Log on to your machine using SSH as the user specified in class.

Once you are logged in, take a look at all the packages installed on your
system:

```
$ dpkg --get-selections
```

All installed packages fly by on the screen. Let's slow that down:

```
$ dpkg --get-selections | less
```

The "less" command lets you quickly search text. Is the "openssh-server" server
installed on your machines? (It should be if you are logged in :)

Type "/openssh" and press <ENTER>

You should see something like:

```
openssh-client                                  install
openssh-server                                  install
```

with the "openssh" text highlighted. Press "q" to exit the less screen.

Another way to see packages is:

```
$ dpkg --list | less
```

Try it!

OK, what version of "openssh-server" is installed?

```
$ apt-cache policy openssh-server
```

Or, you could also say:

```
$ dpkg --list openssh-server
```

2. Find out if a package is available to be installed

You have a local cache of all packages available to be installed from the Ubuntu
package repositories. You can search this cache using the "apt-cache" command. Before
you can use apt-cache the first time you need to update your local cache. Let's do this
now (we did this for you when setting up your machine):

```
$ sudo apt-get update
```

Once this completes we can search for available packages. Let's see if the "ipcalc"
package is available in our Ubuntu repositories:

```
$ apt-cache search ipcalc
```

It looks like there are three packages matching the name "ipcalc". Try typing:

```
$ sudo apt-get install ipcalc
[sudo] password for sysadm: .... <- your password

$ ipcalc 41.93.45.101/24
```

This is very useful! We'll talk more about what all this means later today or
tomorrow.

3. Stopping and starting services

The scripts to run services on your machine are located in /etc/init.d/. By default,
when Ubuntu installs a package the startup scripts for the package are run and the
package is configured to automatically run at system startup.

Try viewing the status of the ssh server, stopping and starting the server and

reloading the server's configuration file (/etc/ssh/sshd_config):

The control script for ssh is here:

```
/etc/init.d/ssh
```

... but it is more common in modern Linux to use the "service" command to control
services:

```
$ service ssh help
```

You are shown the commands you can perform on the ssh service.

Try to view the status of the ssh server:

```
$ sudo service ssh status
```

Since we are connected using ssh we cannot stop this service. If we did, then you would
lose your connection and need to go to your machine's console to manually restart the
service.

is a web server running:

```
$ sudo service apache2 status
```

Yes? Let's look at the default page of your machine's web server:

```
$ lynx localhost
```

Type "q" to exit this text-based web browser (a very powerful tool).

Let's stop the Apache web server:

```
$ sudo service apache2 stop
```

Can you see the web server's default page any more?:

```
$ lynx localhost
```

Let's start the server again:

```
$ sudo service apache2 start
```

And, verify that you can see the page:

```
$ lynx localhost
```

Later today, or tomorrow we'll look at other ways to check for running services.

4. Turning a service off

If, for some reason, you decide that a currently running service should be turned off
permanently, but that the software should not be removed, then you need to use the
update-rc.d utility.

To stop the Apache web server permanently you would do:

```
$ sudo update-rc.d apache2 disable
```

Did you see something like this?

```
 Disabling system startup links for /etc/init.d/apache2 ...
 Removing any system startup links for /etc/init.d/apache2 ...
   /etc/rc0.d/K09apache2
   /etc/rc1.d/K09apache2
   /etc/rc2.d/S91apache2
   /etc/rc3.d/S91apache2
   /etc/rc4.d/S91apache2
   /etc/rc5.d/S91apache2
   /etc/rc6.d/K09apache2
 Adding system startup for /etc/init.d/apache2 ...
   /etc/rc0.d/K09apache2 -> ../init.d/apache2
   /etc/rc1.d/K09apache2 -> ../init.d/apache2
   /etc/rc6.d/K09apache2 -> ../init.d/apache2
   /etc/rc2.d/K09apache2 -> ../init.d/apache2
   /etc/rc3.d/K09apache2 -> ../init.d/apache2
   /etc/rc4.d/K09apache2 -> ../init.d/apache2
   /etc/rc5.d/K09apache2 -> ../init.d/apache2
```

these are logical links in the file system telling it to not run the Apache
web server at any runlevel the next time the machine starts. If you really did
not want the Apache web server to be running any more right now, then you would,
also, need to manually stop the service.

Oops! But, we'll need the web server. Let's re-enable the server:

```
$ sudo update-rc.d apache2 enable
```

Type man update-rc.d for more details on how this works.

5. Reboot your system

To restart your system, you could use:

```
$ sudo shutdown -r TIME
```

... where time can be a day, hour, minute...

Or you could try and reboot your machine *NOW*:

```
$ sudo shutdown -r now
```

The "-r" means reboot. Another command for doing this is "reboot". Go ahead and reboot your machine. You will lose your ssh connection, have to wait a few moments and then be able to reconnect to your machine.

To stop a machine you could do (don't do this now!):

```
# halt -p
```

or

```
# shutdown -h -P now
```

Be careful when using halt on remote systems! Don't do this in class. If you
do let the instructor know and they'll restart your machine.

6. Figure out how your machine has been partitioned

You want to display free disk space, or "df":

```
$ df -h
```

Use:

```
$ man df
```

to understand what the "-h" option does.

Look in /etc/fstab. This is where file systems are mounted in Linux. Read the
man page on this file:

```
$ cat /etc/fstab
$ man fstab
```

Notice that defined file systems are pointing to /dev/vda*. Have a look
at these files:

```
$ ls -lah /dev/vda*
$ file /dev/vda*
```

What type of files are these?

7. Use the top command

The top command let's us see the status of our system at a quick glance. To use top simply do:

```
$ top
```

The item at the top of list of running processes is the process using
the most CPU resources.

Open a new SSH connection to your PC. In that window type:

```
$ ls -lahR /
```

Now in the other window where top is running you should start to see the "ls"
process listed using some amount of your total CPU.

At the top of the top window you'll see something like:

```
top - 03:17:03 up  1:47,  2 users,  load average: 0.51, 0.19, 0.09
Tasks:  79 total,   2 running,  77 sleeping,   0 stopped,  0 zombie
Cpu(s):  4.9%us, 10.9%sy,  0.0%ni,  3.6%id, 79.6%wa,  1.0%hi,  0.0%si,  0.0%st
Mem:    508924k total,   491968k used,    16956k free,    59052k buffers
Swap:   905208k total,     4584k used,   900624k free,   128712k cached
```

This is a good, quick way to see how much RAM, Virtual memory, CPU, total running processes, etc. that your machine has, and is using.

You can adjust the output of top as it is running. Exit from top by typing "q" and then do:

```
$ man top
```

Now run top again and change what it is displaying interactively.

All the information in top is part of a dynamic file system located in
/proc. As an example do the following:

```
$ cd /proc
$ ls
```

The numbered directories correspond to actual Process IDs of processes
that are running. Look at the file meminfo:

```
        $ less meminfo
```

Remember: space bar to go to the next screen of output.

Note that it includes your total RAM. Top uses this file to get this information. Same for cpuinfo, loadavg, uptime, etc.

If you want to know what command was executed to start a number process
you can type (for instance):

```
        $ less /proc/1/cmdline
```

You'll see that the first process started on the system is init.

8. Viewing your log files in real time

Now that you have two ssh windows open to your machine do the following:

In one window type:

```
        $ tail -f /var/log/apache2/access.log
```

In the other window do

```
        $ lynx localhost
```

The "q" to quit, then do:

```
        $ lynx localhost/junk
```

Do you see the log messages indicating your access to the main page, and your
attempt to access localhost/junk, which does not exist. Note the "404" on the
output line of the message. The number 404 means "Not Found".

Now do:

```
        $ cd /var/log/apache2
        $ ls
```

Note there are several log files.

Look for "404" in the access.log log file:

```
        $ grep 404 access.log
```

Now let's make your web busy and watch this in the log file. Be sure you still
have one ssh window with the tail command running:

```
$ sudo tail -f /var/log/apache2/access.log
```

In the other window do:

```
$  while :; do lynx -dump localhost; sleep 1; done
```

Note the timestamp column in the access.log update each second. When you are done
go to the window where you are running the "while" loop and press ctrl-c to
terminate the process.