Ch. 7 Networking, Internet, and the Web

While this course is primarily a course on how to work in the linux command line environment, knowledge of networking is necessary. Computers and computing machines are becoming ever more connected, and knowing how to work with the protocols that power the network is necessary. However, because this course is primarily geared toward the linux environment, we will be skipping most of this chapter (The vast majority of this material is covered in CS3010).

SSH

The `ssh` command allows for you to open a secure shell into a remote host:

```
ssh username@remoteHost
```

If you would like, you can also use the `-X` flag to enable X11 forwarding, a protocol which allows for the user to use applications via the tunnel. The downside, however, is that applications are data intensive, and the heavy load of information could slow down the interactions with the server.

SFTP

You'll inevitably need to transfer files one way or another via the command line. You can't simply just `cp` a file to a server. You instead need to use `SFTP`, or `Secure File Transfer Protocol` to move files from one location to another. `sftp` works luckily very similarly to `ssh` in that you need to log into your remote host first:

```
sftp username@remoteHost
```

Once you log in, you can navigate simply enough with the standard bash navigation: `cd`. To see where you are, you can also `ls` and `pwd`. To see a whole list of the commands, you can type `help`

Downloading a File:

To download a file from a remote server after logging in with sftp, you simply need to either navigate to the file itself, or provide the full path to the file and use the `get` command:

```
get someFile.txt
```

This will then download the file to the directory from which you logged into sftp.

Uploading a File

To upload a file via sftp, you need to provide the path from your home directory to the file as an argument to the `put` command:

```
put /Users/mjlane/Projects/test/Main.java
```

While it is possible to use a relative path with `sftp`'s put, it is significantly easier to provide the full path to the file you wish to upload. Depending on which directory you started from it can be confusing. To

Chapter 8: Basic System Adminstration

8.1 `sudo`

While we don't necessarily have super user powers on our current system, it is good to know how to use them. When you wish to execute a system level command, or override parameters, change owners of a file, etc, you can preface your commands with `sudo`.

`sudo` (accompanied with a password) lets a command run with super user privileges. This is a process that should ultimately be used sparingly since granting super user privileges to every process could result in some unintended consequences.

8.3 Managing Processes

Processes can kick off other processes. It's not entirely out of question that some of your processes will accidentally kick off more processes than they originally intended, processes could be left spinning infinitely, waiting indefinitely on input that will never come, etc.

In order to keep your processes well managed, you can view them with the `ps` command:

```
ps
```

> PID TTY TIME CMD
>
> 15688 pts/0 00:00:00 bash
>
> 20116 pts/0 00:00:00 ps

The `ps` command will display all of your current processes along with their PIDs, their command name, and the length of time they've been running.

Another method of viewing what is running on your environment is the `top` command. The downside, however, is that the `top` command doesn't display only your processes initially. It by default displays all processes running on the server in a separate window:

```
top
```

As you have likely noticed by now, every process has a `PID` or a *process_id* associated to it. Additionally, if you know the name of a specific command that is running, you can get its `PID` by using the `pidof` command:

```
pidof commandName
```

Process IDs are how the system labels each process uniquely. To specifically target a process for stopping, we can use the `kill` command to send a signal:

```
kill -9 PID
```

So, suppose I had a long running application, such as `takeforever.sh`:

```
./takeforever.sh &
ps
```

> starting to do work!
>
> ps
>
> PID TTY TIME CMD
>
> 15688 pts/0 00:00:00 bash
>
> 20830 pts/0 00:00:00 takeforever.sh
>
> 20831 pts/0 00:00:00 sleep
>
> 20832 pts/0 00:00:00 ps

```
 kill -9 20830
ps
```

> PID TTY TIME CMD
>
> 15688 pts/0 00:00:00 bash
>
> 20854 pts/0 00:00:00 ps