Some Helpful Linux (Unix) Information/Commands that might help you in ENSC251

First, like Windows, Linux and Unix (for our purposes, they are the basically the same) use a hierarchical directory structure. What would be called a "folder" in Windows is called a "directory" in Linux. Each directory can contain a number of files and other subdirectories, just like Windows.

Unlike Windows, Linux users usually use a command line interface (rather than a GUI). From what I have seen in industry, most engineers spend much of their time using a command-line interface, so if you want to be employable (not just successful in ENSC251), it is worthwhile learning just a little bit of the command line interface (you don't have to know very much to be effective). With experience, you will find that the command line interface is much faster and more flexible (how many times have you searched window after window for a particular setting… what if everything could be done by just entering a command?).

Each user has a "home directory", and when you first log in, this is the directory you are in. You can make subdirectories and files within that home directory.

When you start Linux, open up a terminal window (if your configuration doesn't automatically open one for you. You will face a command line prompt (often it contains something like a ">") followed by a flashing cursor. This is where you will type all your commands. Some commands (like "gedit") will open windows, but most will just produce text output.

Some of the commands you can type at the Linux prompt include:

ls

The "ls" command lists all the files in the current directory. When you log in the first time, you may not have many files (but there might be a few there).

mkdir directoryname

This command will make a subdirectory within your current directory. The new directory will have the name *directoryname*. You can create as many directories in your current directory as you like. Try making a few, and then doing a *ls* to see the new directories.

cd directoryname

This will allow you to "go into" a subdirectory called *directoryname* (which must have been previously created using *mkdir*). When you go into a directory the first time, of course it will be empty. You can create a new subdirectory within a subdirectory, etc.

If you want to go back to the parent directory, use ".." (two dots). So for example,

```
cd ..
```

would move you up to the parent directory.

Some comments about files:

Like Windows, files can be of many types. Usually (again like Windows), an extension is used in the filename to indicate what type of file it is (.html, .txt, for example). This is just convention, you don't *have* to do this. A special type of file is a "text file". To create a text file, you need a text editor (same idea as Winpad or variants in Windows). Religious wars have been fought about the best text editor to use in Linux. Most text editors (but not all) do not open up a new window, but instead use the current window to display and edit the file. The two most common text editors are called "emacs" and "vi/vim," but they may take some time to learn (I have a separate "vim cheatsheet" handout for those of you who want to try it). However, you may find that gedit is perhaps the simplest to learn; it opens a new window to edit the file, meaning Xming (or other X-server software) has to be running.

To create a file:

```
gedit filename
```

This will open the editor to create the file filename. When you save, your file is written to the current directory, and is called *filename*. Create a number of files in your current directory, and do an *ls* to see that they are all there.

Note that some people choose to edit their files on Windows, and then transfer them using a file transfer facility. If you find that easier, go ahead- it's fine for this course. However, be aware that if you do this when you go to industry, people will laugh at you. (Also be aware that a "*dos2unix*" command may be necessary as carriage returns in Windows and Linux environments are different.)

To remove a file:

```
rm filename
```

deletes the file named *filename* in the current directory. Note that there is no "trash can" to save you if you make a mistake, so delete with care! One way to interpret this is, "real engineers don't make mistakes." However, the better way to think of it is: "***<u>Real engineers always make backups.</u>***"

To copy a file:

```
cp filename1 filename2
```

The above command will make a copy (duplicate) of *filename1* and call it *filename2* in the current directory. After executing this, the two files will be identical.

Sometimes, you might want to make a copy of an entire directory. You can do this, by using the *–r* flag in the cp command as follows:

cp –r dirname1 dirname2

The above will make a duplicate of the directory named *dirname1* and call it *dirname2*. All files and subdirectories are copied (and subdirectories of subdirectories, etc…). In short it does a "recursive" copy. You might consider doing this to make backup copies. So, if you are in your home directory, and you have a subdirectory called ensc251, you could do something like this:

cp –r ensc251 backup

It will create a backup of everything in that directory just in case disaster strikes later. I would strongly suggest doing this regularly, particularly if you use no other form of revision control! There is revision control software (e.g. cvs, svn, and github), that provides a more powerful set of features; but we'll talk about that later (google for more info if you want).

You can also copy a file into your current directory using the following command:

cp /bin/important_stuff.txt .

Note that the destination name in this case is a dot. The dot means "put it in my current working directory and call it the same thing as the source directory". So, in the above, you will get a copy of the *important_stuff.txt* file from the /bin directory in your working directory.

You can also remove an entire directory using:

rm –r dirname1

It will remove the directory named *dirname1* and everything below it (recursively). Warning: this is very dangerous! You can get in trouble very quickly. One command could mean you delete your entire ensc251 directory! The department does have backups, but it would take some time for them to dig the tapes out, so be very careful with this. My suggestion is *never* to use the *–r* option with *rm* during this course as Linux is too new to you.

Another miscellaneous command you might use is "ssh," which allows you to log into another machine (e.g. like the server where you will submit your assignments & project).

On Filetypes:
Just as in Windows, Linux files can be of different types. However, most tools create files in a human-readable text format. Even large CAD tools (like those from Altera and Xilinx) rely on scripts and text files to describe a designer's circuit. This means you can open the circuit descriptions with a text editor! This is different than Windows (you can't open a .doc file with a text editor). Engineers like this, because it allows them to go in and understand the files and possibly edit the contents directly, even if the original tool is not available or difficult to use. This is something you will really appreciate later, if you choose to take ensc452 or ensc450.

On Directories:
Your directory is actually a subdirectory of some higher level directory that contains directories for all the team logins of your classmates on your workstation. That directory is a subdirectory of an even higher-level directory, and so on. You can quickly refer to your home directory using the tilde symbol. If your ENSC userid is *lesley*, then *~lesley* refers to your home directory. So, you can go

cd ~lesley

to change to your home directory.

You can also specify longer pathnames using the / symbol. This is the same as in Windows (except Windows uses the \ symbol). So, if your user id is *lesley*, and you have a subdirectory called *ensc251* and that directory contains a subdirectory called *assign2*, you could quickly change to that subdirectory using:

cd ~lesley/ensc351/assign2

Technically, you could also do something like this: if your friend's ENSC id is *bob*, you could do this:

cp –r ~bob/ensc251 ~lesley/ensc251

This will copy his *ensc251* directory into yours! This seems like a security hole, and it is. By default, everyone else can see all your files! You can set the "permissions" for your directory so that no one can see your files using:

chmod og-rx ~lesley

where *lesley* is your userid. This will prevent anyone from seeing any of your files (including me). There's a lot more you can do with the chmod command, but that is beyond the scope of this document.

System Directories:
There are a number of directories in the system that do not belong to anyone in particular. These directories contain files that everyone might find useful, as well as executable files. Script files can be executed using the "source" command (commonly "shell" script files), which are generally denoted with ".csh" and ".sh" file extensions.