Introduction

This class utilizes the course locker system provided by the Unix cluster in the SOE. This document assumes you have no prior knowledge of terminals, SSH, and the submission commands. If you do have knowledge of terminals and SSH, skip directly to the last section for reading on how to submit.

Terminals

The virtual terminal is a command line interface to the operating system that you are on through an interface called a shell. This interface allows the user to type commands to run or interact with most software on the computer, which POSIX systems such as Mac OS X and GNU/Linux offering more capabilities by default.

This documentation will only discuss the Bash shell as that is the default on the Unix cluster that you will be using. The ‘$’ symbol at the start of the examples refers to the prompt given to you by the Bash shell and everything after that is what you will enter.

The shell is all about file handling and processing input and output between different commands. The shell has a concept of a “current working directory”, which is where all commands execute relative to by default. This is like how your file browser works, where it generally displays the contents of a single directory at a time. Spaces are used to separate the various commands, arguments, and parameters entered.

Changing directories

To do this you can use the change directory command, `cd`. If no argument providing an argument in the form of a path. The ‘~’ symbol has a special meaning when used as a path and refers to your user directory (usually located at /home/USERNAME). Additionally the ‘.’ symbol refers to the current working directory and the ‘..' symbol refers to the parent directory.
To enter a sibling directory of the current directory.
$ cd ../cmpe13

To go back to your home directory:
$ cd ~

List files

To list the files in a directory, the list files command `ls` is used. It takes a list of paths to list the files at, with the current directory being the default. Specifying the additional argument of `-hal` turns on human-readable file sizes, shows hidden files, and uses a one-file-per-line list with file details. Can be useful.

To list all files in your home directory:
$ ls ~

To list files in a subdirectory:
$ ls -hal Spring14/cmpe13/lab1

Copying files

The `cp` command does this.

Copy the provided header file as the basis for a new source file (make sure all header-specific code is removed!):
$ cp Leds.h Leds.c

Making directories

Use the mkdir command. It takes a single argument: a path to make. Note that it can only make one level of directories at a time, so if you want a cmpe13 directory inside a courses folder and the courses folder doesn't already exist, you'll need to make the courses folder first:

$ mkdir courses/cmpe13
mkdir: cannot create directory `courses/cmpe13': No such file or directory
$ mkdir courses
$ mkdir courses/cmpe13

Deleting files/folders
To delete files the `rm` command is used. Arguments are a list of file names. Note that it only works for files. For deleting directories, use the `rmdir` command on an empty folder or use `rm -rf`.

**Running programs**

Programs on the shell are run similar to the `cd` command introduced previously. They always look like, where arguments are optional):

```
$ PROGRAM [ARGUMENT0 ARGUMENT1 ...]
```

PROGRAM is going to be a path, or if the program is in a special location, merely the name of the program. Arguments are space-delimited and come after the program name as the following examples show.

Most programs provide their own documentation. Some programs output this help when run with no arguments. Almost all programs will output their documentation when given a --help command. Finally some programs provide help via the `man` program which takes a single argument, the name of the function (if you use `man`, the arrow keys scroll and typing 'q' and then ENTER will quit).

```
$ tar --help
$ tar -h
$ man tar
```

To pull down a file from the website:

```
$ wget http://classes.soe.ucsc.edu/cmpe013/Spring14/Labs/Lab1/Lab1.zip
```

To unzip a zip-file (where the file is specified as the first argument to the unzip program):

```
$ unzip Lab1.zip
```

To zip up files into a zip-file for emailing or saving:

```
$ zip lab0_done.zip part3.c
```

**SSH**

SSH stands for Secure Shell and is a way to securely access a terminal on a remote machine. To login you need to know your username and password to the remote machine and have an IP address or URL to the machine.

For accessing the Unix cluster at UCSC, use your CruzID Blue login credentials and the URL 'unix.ucsc.edu'.

On Windows, the **PuTTY** program provides an SSH client and terminal. On most other operating systems, a standard terminal program will work where you can run the `ssh` command (search through provided applications for 'terminal' or look in the System Tools section of the main menu).

From the Bash shell on GNU/Linux or Mac:

```
$ ssh USERNAME@unix.ucsc.edu
```

To use PuTTY on Windows (which is already available on the lab computers), just launch it from the start menu. Enter the following:

For address: unix.ucsc.edu
For port: 22

Once it starts it will prompt you for your username and password.

**Copying files**

Files in your home directory (the X: drive on Windows lab computers) is shared with the Unix cluster computers, so no file transfer is required if you only use the lab computers.

If you use a separate computer, you will want to copy files back and forth between computers. You can do this most easily by using SFTP with the easiest option for all platforms to be **FileZilla**.

To use FileZilla, you just enter the same information as required to do SSH:

Address: unix.ucsc.edu
Port: 22
Username: SOE_USERNAME
Password: SOE_PASSWORD

After you log in you will see your directory on the Unix cluster on the right and your home directory on the local computer on the left. Simply drag files back and forth to copy them. You can also drag files from a file browser into the right-hand pane to copy files, so you don't always have to browse to the files on the left.

**Using the course lockers**
The course lockers are just directories for the class that you have write privileges to and read privileges for your own files. The easiest way to write to them is to use the `submit` command with at least 3 arguments (FILELIST can be a space-delimited list of filenames): COURSENAME-INSTRUCTORINITIALS.QUARTER, ASSIGNMENTNAME, FILELIST. You can submit multiple times, so you should submit your assignment as you complete each section, this way even if something horrible happens to your work, you have some working code submit. Good points for this would be when finishing a library or a specific feature in the main file.

To submit to the Winter 2015 CMPE13 class for lab 1:

```
$ submit cmpe13-md.s15 lab1 part1.c part2.c Oled.h OledDriver.h Serial.h part3.c README.txt
```

To submit all C source files in the current directory you can use the special '*' character to indicate any string:

```
$ submit cmpe13-md.s15 lab2 *.c
```

To view the files you've submitted for an assignment, like lab 1, just use the `ls` command to list the files in the appropriate directory:

```
$ ls /afs/cats.ucsc.edu/class/cmpe13-md.s15/lab1
```