CpSc 1111 Lab 1
Introduction to Unix Systems, Editors, and C

Welcome!

Welcome to your CpSc 1111 lab! For each lab this semester, you will be provided a document like this to guide you. This material, as well as in-lab instruction, is intended to supplement lecture by focusing on the hands-on aspect of programming. What better way to learn than by doing this! Plus, you'll get individual help if you run into problems.

Typically, the labs serve to reinforce concepts learned in lecture. Sometimes, additional related concepts will be presented in lab. Most labs will involve some reading, some computer activity, and a deliverable to turn in for a grade. The best way to work is to read straight through, working on your computer when asked. You may not finish by the end of the first lab period, but you will have the Thursday lab period to finish up and get more one-on-one help from your lab TA, plus you should be able to work from anywhere with an Internet connection.

Overview

By the end of the lab, you will be able to:

- log on to and navigate the School of Computing's Unix workstations
- use one of the many text editors available in Unix
- create, compile, execute, modify, and submit a simple C program

Login and Navigate Unix

1. Log on to your Unix workstation
   - On the login screen is a place to enter your user name, and four controls:

     "Language", "Session", "Restart", and "Shutdown"

   - These controls are generally not useful, but you might find "session" useful in rare instances of severe "desktop malfunction."
   - Type your *CU username* in the space provided and then press the enter key.
   - When prompted, enter your password to complete the login procedure.

2. Starting up
   - Open a terminal window. If you don't see a terminal (command-prompt), it should be under Applications <Utilities/Accessories> Terminal.
   - Enter the `pwd` command. (It stands for "print working directory"). The working directory should be something like "`/users/your_username`" or "`/home/your_username`". This is also known as your *home directory*. Terminal windows open to this location by default. All your files and directories will be kept in a directory tree with this directory as *root*. Another name for your home directory is simply `~` (the tilde character), which you will see from time to time.

> **NOTE:** You may name and organize your directories however you want – whatever makes sense to you. Nobody else will be accessing your account – only you (DO NOT give out your password to anyone else!). If you do name your directories differently than in the instructions below, the commands you type will be different according to the names that you choose. For example, if you choose to name your first directory `cpscl111` instead of just `1111`, then when you change directories into that one, you would type `cd cpscl111` instead of `cd 1111`.

3. Create and navigate CpSc 1111 directories
   - Type `mkdir 1111` and hit the Enter key to create a directory for your CPSC 1111 files.
   - Type `ls` and hit the Enter key to list the contents of the current directory. You should see the new 1111 directory.
   - Enter `cd 1111` to change directories into the new 1111 directory.
   - Enter `mkdir lecture` to create a directory for the lecture portion of this course. You can use this directory and any others that you create inside of it for your lecture work – class examples, practice programs, assignments, etc.
   - Enter `mkdir lab` to create a directory for the lab portion of this course.
   - Enter `ls` to list the contents of the current directory. You should see both of those new directories.
   - Enter `cd lab` to change directories into the new lab directory
   - Enter `mkdir lab1` to create a directory for today's lab.
   - Enter `mkdir temp` to create a temporary directory.
   - Enter `ls` to list to verify both of those new directories are now there.
   - Enter the ls command again this way: `ls -l` which uses the `-l` flag to display the listing in long format. Notice all the extra information that is displayed!
   - Try another flag with ls: `ls -a` which shows all the files, including hidden files
   - You can also combine the flags this way:
     - `ls -a -l` or
     - `ls -l -a` or
     - `ls -al` or
     - `ls -la`

   - Use the `cd` command to change to the `lab1` directory.

   - To get all the way back to your root directory from anywhere that you may be in your directory tree, you can simply type `cd ~` or just `cd`
   - Typing `cd -` will take you back to the last directory you were in (which is different from `cd ..` which goes back to the parent directory).
   - You can also specify multiple directories at once:
     - `cd 1111/lab/lab1` to go directly to your lab1 directory from your root directory
     - `cd ../../lecture` to go directly to your lecture directory from your lab1 directory
   - You can also use absolute paths:
     - `cd /home/username/1111/lab/lab1` to go directly to your lab1 directory from anywhere

Useful Unix Commands To Get Started

Command	Purpose
ls	list all contents in your current directory
cd	takes you back to your home directory
cd ..	takes you back one directory
cd *dir_path*	takes you to the directory specified by the path provided
mv *src_file dest_file*	renames *src_file* to *dest_file*
mv *src_file dest_path*	moves *src_file* to the folder specified by the *dest_path*
mv *src_file dest_dir/dest_file*	moves *src_file* to the folder specified by *dest_path* and gives it the name *dest_file*
cp *src_file dest_file*	copies *src_file* to *dest_file*
cp *src_file dest_path*	copies *src_file* to the folder specified by *dest_path*
cp *src_file dest_dir/dest_file*	copies src_file to the folder specified by dest_path and gives it the name *dest_file*
rm *file_name*	deletes the file named *file_name*
mkdir *dir_name*	creates a directory name *dir_name* in your current directory
rmdir *dir_name*	deletes the directory named *dir_name* if it is empty
rm -rf *dir_name*	deletes the directory named *dir_name* (be careful when using -rf)
cat *src_file*	shows the contents of the file on the screen without opening it in an editor
control-c	send the terminate signal to a running process (kills the current process); good to use if your program is stuck in an infinite loop, for example
ps	shows a listing of processes that are running
kill -9 *[pid]*	kills the process you specify with the pid (process id #) which is shown when you type ps
man *unix_command*	displays the manual page (help page) for the specified Unix command
logout	logs you out

Also, here is an online **Unix Tutorial** http://www.ee.surrey.ac.uk/Teaching/Unix/. It looks like it might be a pretty helpful tutorial to look over, or perhaps a good reference to bookmark.

Text Editors

Which editor should you use????

- A typical Unix system provides several different text editors. On our system, we can use gedit, pico, nano, vi, vim, Atom, Emacs, and perhaps others. Any of these can be used to create and edit files of any type, including C programs. Some have more advanced features that make elements of programming easier. Which one you choose doesn't really matter for creating your programs, but keep in mind that the formatting of your code may be "off" depending on your choice of editor. Your lab TA (and instructor) can explain this further. ***(Points may be lost with incorrect or inconsistent formatting.)***

1. To use **gedit**, select the *Applications control* on the top panel and then left click *"Text Editor"*. This will start the *gedit* text editor. You may also want to create a *gedit* launcher on the top panel or on the desktop.

2. To use **vim**, simply type at the Unix command prompt `vim <filename>` which will open up a file for editing in the vim editor. (**vim** is the "new improved" **vi** editor – on our system, you can type either **vi** or **vim** and then the filename to start up the **vim** editor). This editor has more advanced features and is much more powerful than gedit – but – it has more of a learning curve than the others. The same is true of Emacs.

Vim resources:
Vim 101: A Beginner's Guide to Vim https://www.linux.com/learn/vim-101-beginners-guide-vim

A table with some of the more common vim commands to get you started
http://people.cs.clemson.edu/~chochri/Courses/Documents/vimCheatSheet.pdf
Another vi reference card https://people.cs.clemson.edu/~etkraem/1730/Labs/Lab1/viquickref.pdf
Another vi "cheat sheet" https://people.cs.clemson.edu/~etkraem/1730/Labs/Lab1/viref2.pdf

Vim Adventures http://vim-adventures.com/
An Introduction to Display Edition with vi http://docs.freebsd.org/44doc/usd/12.vi/paper.html
Why should I bother to learn vi? http://www.viemu.com/a-why-vi-vim.html

NOTE: When your files are viewed using vim, if you had used pico, nano, and maybe even gedit to create the files, this is where your formatting may look different in vim than it does when opened with those other editors. In other words, for those instructors who use vim to grade programs that were created using pico, nano, or gedit, the formatting may look different and be incorrect or inconsistent, resulting in a loss of points.

Creating Your First File

You will create a text file named `test.txt` and save it in your `1111/lab/temp/` directory.

1. In a terminal window, cd to your `1111/lab/temp/` directory.
2. Use an editor of your choice to create a file called `test.txt` . Edit the file, putting whatever you want in it, perhaps your thoughts on how much you are going to love this class.
3. When you are finished with adding text to the file, save the file and quit the editor.
4. At the Unix command prompt, type `ls` to see if the `test.txt` is there. If it is not, then you must have somehow not saved the file before quitting the editor – go back to the editor that you are using and create it again.
5. Using an editor, open up the file to see that whatever text you had added to the file is there.

Deleting and Copying Files, and Deleting Directories

1. In the terminal window in the `1111/lab/temp/` directory, delete `test.txt` by typing `rm test.txt`
2. Type `ls` and confirm the file is gone.
3. Type `touch file1` to create an empty file. You can use touch to create multiple new files with one command: `touch file1 file2 file3`
4. Typing `ls` will confirm the files created with `touch` are there.
5. Type `cp file1 test1` and then type `ls` to see what is there.
6. Type `mv test1 test2` and then type `ls` to see what is there. See the difference between `cp` and `mv`?
7. Delete (remove) the files you just created. You can do this by removing each one individually, or by typing the following, using a wildcard, the `*` which means "all files": `rm *`
   - o You will be prompted to enter a Y or N for each file to make sure you do intend to remove the file.
8. Leave the `temp` directory by typing `cd ..` to return one directory back to the `lab` directory.
9. Now delete the `temp` directory by typing `rmdir temp`. Notice that files are removed with `rm`, but directories are removed with `rmdir` and have to be empty before being removed.

Writing a C Program

Use the editor of your choice to create a simple C program in your `1111/lab/lab1` directory that will print the words "Hello, world!" to the screen. Name your file `simple.c` and type the following code:

```
/*  Name
    CpSc 1111 Lab, Spring 2018
    Lab #1
    My first "hello world" program.
*/

#include <stdio.h>

int main(void) {
   printf("Hello, world\n");

   return(0);
}
```

Reminder About Style, Formatting, and Commenting Requirements

- The top of your file should have a header comment, which should contain:
  - Your name
  - Date
  - Lab section
  - Lab number
  - Brief description about what the program does
  - Any other helpful information that you think would be good to have.
- Variables should be declared at the top of the main function, and should have meaningful names.
- Always indent your code in a readable way. Some formatting examples may be found here: https://people.cs.clemson.edu/~chochri/Assignments/Formatting_Examples.pdf
- Don't forget to use the `-Wall` flag when compiling, for example: `gcc -Wall lab3.c`

Compiling a C Program

1. In a terminal window, cd to `1111/lab/lab1`. Enter `ls` to confirm `simple.c` is there.
2. The compiler we use is gcc. To compile `simple.c` enter the command: `gcc simple.c`
3. If there are errors, go back and confirm you entered the program correctly, fixing the errors indicated by the compiler. Recompile after fixing the errors. You may have to edit/recompile several times until there are no errors left.
4. When there are no errors, use `ls` to confirm that the compiled machine language program (the executable) was created (it will have the name `a.out`)
5. To run the program, type the name of the executable, by typing `./a.out` The ./ is used to tell the system that `a.out` is in the *current directory*.
6. "Hello, world!" should print on its own line.

Turn In Work

1. Before turning in your program, edit `simple.c` to print out Hello World 20 times (DO NOT use a loop).
2. Rename the file as `lab1.c` and then compile it and run it to make sure it works. Always test, test, and retest that your program compiles and runs successfully on our Unix machines before submitting it.
3. Show your ta that you completed the assignment. Then upload your `lab1.c` file to the SoC handin page at http://handin.cs.clemson.edu. ***Don't forget to always check on the handin page that your submission worked. You can go to your bucket to see what is there.***