CS410C FALL 2019 DUE: 8/28/2019

LAB 0L

`Submit code:` **`0L`**

Description

This exercise is designed to give you practice using most of the tools you will need to edit, compile, and submit a program. These are basic skills you will need to complete all your programming assignments for the semester. You will copy a template file from the course public directory, make a few changes to it, and then compile, test, and turn in the file. Once you have done that you should practice with the system by trying some of the commands described below. In general, if you finish with the lab early, you are free to leave or start working on the next programming assignment. Feel free to get help from other students as well as from your TA/instructor on the lab assignment. Although seeking help from other students is ok for lab assignments, all work submitted must be your own and not copied from another student. Also, you cannot work together with another student to a point where your programs are pretty much the same.

Although you can help each other on the lab assignments, such help is strictly forbidden on the programming assignments. Your only source of outside help on the assignments will be your instructor, TA, and PAC consultants. Even a single case of collaboration on coding an assignment may result in an F for the course.

Setting Up Your Desktop(s)

Many of the computer clusters on campus have machines that run the Microsoft Windows operating system. Nearly all of the public computers in the CS department instead run the Linux operating system (including all the machines in the PAC—Programming Assistance Center). Furthermore, you will be required to compile and run all your programs on a Linux computer in the CS domain (if you're logging in remotely, your only choice is agate.cs.unh.edu). The following instructions will make it easier for you to use any of the cluster machines to access the CS domain. Note that you will need a different procedure to set up your own laptop, but some of the items would be similar. The following instructions should only have to be performed once, and will make working on your homework from a cluster machine much easier. Based on type of workstation you are logged into, follow instructions for either Linux or Windows.

1. **On Linux:** Most CS department machines run Linux, you will see a prompt for username and password
   On Windows: Make sure that the "WILDCAT" domain is selected before you try to log in.
2. Log in using the same username and password that you use for Blackboard. **If a dialog pops up giving you some options, press the middle button that says "Use Default Configuration".**
3. **On Linux:**
   a. Open a "Terminal" window by clicking on the icon in the bar at the bottom of the screen that looks like a dark computer monitor (or select "Terminal Emulator" from the little "f" symbol in either the top- or bottom-left corner of the screen).
   b. Type the following command to copy several files from one of my directories into your Desktop. Make sure everything appears *exactly* as you see it below, including the tilde (~), the slashes, the star (*), and the capital 'D' on Desktop. Note that there is a space just before the tilde (~) and after the star (*) but nowhere else.
      ```
      cp ~cs410c/Desktop/* Desktop
      ```
   c. To edit files on your Linux workstation you can use the SciTE editor. There should be a shortcut on your desktop at this point, if there is none you can access this editor using the Application/Development option under Menu.

   On Windows:
   a. Click on the "Start" menu button in the bottom-left corner of the screen.
   b. Navigate through the following items: Programs→Course specific→Computer Science

   c. Now use the *right* mouse button to click and drag the “Scite” application to the desktop. When you let go of the button, a context menu appears. Select the “Copy here” item (with either button) to copy the shortcut for SciTE to your desktop.
   d. Add the PuTTY icon similarly. Instead of navigating to the “Computer Science” menu, go through Programs→Internet Applications, then right-click-and-drag PuTTY to your desktop.
4. You’re done with setting up the desktop. You won’t have to do this for lab machines again.

Starting a Programming Work-Session

Each time you sit down to work on a lab or programming assignment, you’ll need to open a terminal window. A terminal window allows you to type commands to do things like move files, compile programs, and submit them. Again, the procedure is different on Windows and Linux, but they have some similarities:

On Linux:

1. ***Start a terminal window*** by clicking on the same “Terminal” icon that you used when you copied items from my directory (Step 3a on the previous page, if you’ve forgotten). This starts a session on whatever computer you are on.
2. (Later you may want to set up your own computer or laptop so that you can work from home. If you have a windows machine, you’ll want to install PuTTY and connect using that—see the Windows instructions above. If you have Mac OSX then you already have terminal and ssh (which is the same as using PuTTY to connect to the Linux server).
3. Continue onto step 5 toward the bottom of the page.

On Windows:

1. ***Start a terminal*** by double-clicking on the icon you placed on the desktop labeled PuTTY.
2. The first time you connect to a new server, you’ll need to follow these steps once:
   a. Type the server name (in this case “agate.cs.unh.edu”) into the box labeled “Host Name (or IP address)”
   b. On the left-hand side, click on the item that says “Data” under “Connection”
   c. Type your user name into the box labeled “Auto-login username”.
   d. On the left-hand side, click on the item at the top that says “Session”. This takes you back to the original screen.
   e. Type a short name that describes where you’re connecting to (in this case “agate”) into the box labeled “Saved Sessions”.
   f. Press the “Save” button.
3. From this point forward, all you need to do is double-click the short name you selected (in this case “agate”), and you’ll be connected to agate.
4. Type your password to log in. Note that the cursor doesn’t move, but every key that you type is still being understood. If you mess up, it may be best to hit the Backspace key a lot and start over.

Now, on both Linux and Windows

5. When you have opened a terminal window, you should see a shell prompt “[mylogin@agate ~]” (where “agate” is replaced by the name of whatever machine you’re on). Now you can begin working on your course work. The shell prompt is a place that you type in commands, rather than clicking on icons and menu items. Once you know some commands it will become pretty easy to use. From here on out, I will represent your shell prompt using the percent sign.
6. Try entering the following (don’t type the percent sign—it’s just a stand-in for the prompt):
   ```
   % ls $public
   ```
   If you get an error message, then you are probably not properly registered for the course yet, and you will have to enter the following command before you begin:
   ```
   % source ~cs410c/public/410c.login
   ```
7. Now it’s time to make a new directory for working on your CS410 assignments. Type this in.
   ```
   % mkdir cs410
   ```

Do this (steps 8 and 9) only if you want to create and edit your files always in Windows (not recommended since you have to transfer files back and forth between Windows and Linux all the time):

8. When you're working from a Windows machine, you'll have to copy file back and forth between agate and the machine you're working on—FileZilla will help with this. ***Start FileZilla***. In the appropriate blanks, enter "agate.cs.unh.edu" for the host, then your Blackboard username and password, and finally 22 for the port. Then hit the "Quickconnect" button. After the first time, there should be a shortcut from the down-arrow button just to the right of this button.
9. You can find files on agate in the right side of the FileZilla window, and files on your local machine on the left. You can move files back and forth by dragging them from one side to the other. Whenever you make a change on the local machine, remember to drag from left-to-right before you do any commands on agate. Conversely, whenever you create or change a file on agate, remember to drag from right-to-left to update your copy on the local machine.

How to Start Labs and Assignments

In many cases, you will have to copy files from the course public directory to your own working area. This lab is no exception. To start the graded portion of this lab, you'll need to create a new directory, copy the file "template.c" into it, and rename the file. Make sure to read all instructions and be sure you understand it. This is the only time you'll get step-by-step instructions on getting started!

Don't type the %, it represents whatever prompt you actually have.

10. Before doing any work, make sure to change directory to "cs410". The following command changes your *current working directory* to cs410:
```
% cd cs410
```
11. The labs often coincide with upcoming assignments, but it'll be good to keep separate directories for each so you don't overwrite your lab. Create a directory to work in. This lab starts with zero (as does most counting in C). Here's how to create a new directory called "0L" (for lab zero):
```
% mkdir 0L
```
12. Now change into the directory you just made.
```
% cd 0L
```
13. Copy "template.c" from the course directory to your own directory. Here's one way to do it:
```
% cp $public/0L/template.c ./
```
The dot-slash notation "./" indicates that we want to move the file to the directory we are presently working in. If you want to find out your *present working directory*, use the "pwd" command. Give it a try:
```
% pwd
```
If you make a mistake and go into the wrong directory, you can always go up a directory using…
```
% cd ..
```
Also, if you ever get lost, you can just go back to the top of your account by typing just 'cd':
```
% cd
```
…Just remember to use "`cd cs410`" afterwards to get to the right place! If you got lost during this step, make sure to get back to the correct directory:
```
% cd ~/cs410/0L
```
14. For this lab, rename "template.c" to "lab0.c". The command to do this is "mv" (it can also *move* a file from one place to another). "mv" needs you to tell it what you're moving, and where you'd like it to put it. In this case, we're moving "template.c", and just giving it the new name "lab0.c":
```
% mv template.c lab0.c
```
15. For every lab and assignment, you'll need to edit a "source" file to make changes to it; lab0.c is the source file for this lab. To do this on Linux (on a workstation in the lab) go to 15 (a). To do this on Windows on the cluster go to 15 (b).
    (a) To edit the file on your Linux workstation you can use SciTE editor. To use SciTe, your desktop is likely to contain a short-cut/icon which you can double-click. If you don't see it there, choose Applications/SciTE. Once the SciTe editor opens you can navigate to the folder that has your file to open it.
    After you save your changes to the file you are editing you will need to use the Terminal window that's connected to agate to continue.
    Alternately you can also edit directly on agate using a terminal window with text editors such as "nano" or "pico". You are likely to find this cumbersome since the editing commands are limited. The command to edit lab0.c in "nano" would be:
```
% nano lab0.c
```

(b) Only if want to use Windows to edit the file use “Scite” (but remember you will need to keep transferring files back & forth), you can use any text editor you like that saves to a plain text file (not .doc!), just remember to bring it to your machine first (using FileZilla).
Select “Open” from the “File” menu, and navigate through the “cs410” directory, through the “0L” directory, to the file “lab0.c” and open that file. ***Note: Never double-click on a source file like lab0.c on a Windows cluster machine—it will open the wrong editor (Visual Studio), and it may take several minutes to load on a Windows machine.***

16. **[25 points]** From inside the editor, edit the code so that it will print your name when it runs instead of “<Your name here>”
17. **[15 points]** Add your name in a *comment* toward the top of the file, between the top two lines full of equal signs. Each programming assignment you turn in should have your name at the top of the file and a brief description of what the program does—it doesn’t matter for any labs except this one. Add a comment by inserting text (with your name) on a new line that starts with
    ```
    /*
    ```
    and ends with
    ```
    */
    ```
    You can have any number of lines of text between these two parts, as long as the text “`*/`” or “`/*`” doesn’t appear in your comments.
18. Save your work and exit the editor

Compile, Execute, Test, Submit

Remember that compiling, executing, testing and submitting your program will always need to be done from your Terminal window for Linux or PuTTY session if you are using Windows. In other words, compiling, executing, testing and submitting your program is always done on the server “agate”.

Basically you will end up having both your terminal window and the SciTe editor window open most of the times. While you edit and save your files on SciTe editor, you will use the Terminal window to compile and run (execute) your program.

How to Compile and Run Your Source Files

Getting your programs working will almost always require several cycles of edit-compile-test, edit-compile-test. You’ve just learned how to edit a source file; here’s how to compile and test it. Use the compiler called “gcc” to ***compile*** your source file. Enter the following line into the terminal window on agate:

```
    % gcc lab0.c
```

If there are errors, gcc will give you some idea of what line they are on and what is wrong. Later you’ll learn how to read these error messages. For now, just ask a TA for help if you get an error message and if you don’t understand what it is trying to tell you.

[60 points for a running program] After gcc compiles your source file, it creates an executable file called “a.out”—this is your compiled program ready to run. To ***test*** your program, just type

```
    % a.out
```

If it doesn’t run, try “`./a.out`” instead. If it doesn’t work right, edit the source file (which is still be open in SciTE), save it, recompile it, and run it again. Ask a TA for help if you’re stuck!

Continue to work on the program

19. Once you run your program and see the changes you made to print your name works correctly let’s do a few more things:

Add several ***printf*** statements following the one you have. Each ***printf*** statement needs at least 2 things – a pair of open and close parenthesis and a pair of double quotes within the parenthesis. Finally a semi-colon to end the printf statement.
For example:

printf("I am a freshman student in Electrical Engineering");

will display the text within the double quotes.

If you want to print anything on a new line you need to add the sequence \n anywhere within the pair of double quotes.
For example:

printf("I am a freshman student in \nElectrical Engineering");

will print the text on 2 lines with the second line starting with Electrical. In fact you can have the a new line character completely on its own printf like this: ***printf("\n");*** to print a new line.

Add several printf statements and say a few things about yourself – tell me what your experience with computers and programming is, what interests you in your major and with computers, what you plan to get out of the course, etc. Instead of putting down everything in one printf statement split it up into several of them, so that your program is more readable.

What you say about yourself is not going to affect your grade in any manner, just a way for me to figure out what everybody's skills and interests are. Of course you can make it as interesting as you want for me ☺.

20. Save your work and exit the editor

21. Every time you make changes to the source file, you will need to compile your program using the **gcc** command on your terminal window.
    Compile and run your program and make sure that you are happy with what you see.

How to Submit your Source Files for Grading

Once you are satisfied that your program is working correctly, you will submit it so that you can get credit for it, using course commands called "submit" and "check". Here's how to submit your file for this lab:

1. Use the course command "submit" to submit your source file for grading. You need to provide the submit command with a submit code and the names of all your files. In this case, the submit code is "0L" and the name of your file is lab0.c:
   ```
   % submit 0L lab0.c
   ```
   Labs will have submit-codes that end with "L" (must be an upper-case "L") and programming assignments will have submit-codes that end with "P" (again, upper-case). The submit code will be near the upper-right corner of the first page describing your lab or assignment.
2. The very first time you submit, you may see a warning message about a log file. It is ok the first time, but may indicate a problem later. There may be other problems as well, so be sure to pay attention to what shows up on the screen. You should see something like this:
3.
```
starting submission for cs410c

check: no log file - if you have submitted files please tell instructor
creating a compilation area to test compilation
Compilation completed.
submitting files - lab0.c - for assignment 0L

-rw------- 1 fred42 ceps 330 Aug 27 11:58 lab0.c

starting to copy file(s)
      copying lab0.c
confirming - the following shows names and sizes of NEW COPIES of files
                  lab0.c --      330
end of cs410c submission
```

If you see any errors, you may need to visit or e-mail your instructor or TA to get your file submitted. ***You will be charged late penalties if we do not receive your file and you have not contacted us immediately after discovering a problem with your submission***.

4. You can check to make sure that your file(s) has been submitted properly using the course command "check" followed by the assignment number:
   ```
   % check 0L
   ```
   This will tell you what files were submitted, and what time the "submit" command was invoked. If you don't see a time that looks recent, try submitting again or e-mail your instructor or TA.
5. After you submit a program, don't edit it again. An unedited file has a date/time stamp on it that shows when you completed it. This is your proif that the assignment was completed on time.
6. This completes the graded portion of the lab. The next page has additional resources and information you may find useful in the coming weeks. ***Be sure to log out before you leave!***

Finishing a Programming Work-Session

It is very important that you log out properly before you leave the computer you're working on. You don't want to leave your files exposed to others who may copy your work or inadvertently destroy some of it.

On Linux: Click the little icon of the green person running in the bottom-right corner (this is the "logout" button). It's usually best to make sure that the checkbox that says something like "Save session" is unchecked, and then click the "Logout" button.

On Windows: It's ok just to go to the Start Menu and select "Log Off". You may want to type "exit" at the command prompt in the terminal window to be sure your PuTTY session closes cleanly first, though.

Summary of Selected UNIX Commands

Command	Meaning	Command	Meaning
`gcc` *`myfile.c`*	Compile *myfile.c* into the executable named a.out	`pwd`	Find out what the present working directory is

`a.out`
or
`./a.out`
	Run the executable file that was created when you last compiled a code file successfully	`cd` *`someplace`*
		
		`cd ..`
			Change the present working directory to *someplace*. Note that ".." means go up one directory, and "../.." means go up two

`submit` *`7P file.c`*	Submit *file.c* for grading for programming assignment 7	`ls`
		
		`ls -l`
			List the files in the present working directory. The minus-L (-l) argument provides a "long" list with additional file details

`check` *`7P`*	Check to see if your last submission for assignment 7 worked properly	`cp` *`file target`*	Copy the existing file named *file* to the *target*. If *target* is a directory, then *file* is copied into that directory. Otherwise the copy is given the name *target*
`date`	Display the date and time	`mv` *`file target`*	Move or rename the existing file named *file* to *target*. If *target* is a directory, then *file* is moved into that directory. Otherwise the file is renamed to the name *target*
`nano` *`myfile.c`*	Start the nano editor on the filed named *myfile.c*	`mkdir` *`newdir`*	Make a new directory named *newdir* under the present working directory
`quota`	Display how much disk space you're using, versus how much you're allowed	`rm` *`extrafile`*	Remove the file named *extrafile*. This deletes the file permanently—***there is no recycle bin or trash can***
`logout` (or `exit`)	Logout of your ssh or terminal session	`less` *`file`*	Display the contents of *file* a page at a time. Use the space bar to go to the next page and the Q key to quit.