[Spring 2019] Lab 0

Deadline: End-of-day Tuesday, January 29th

Hello! Welcome to CS61C! We're so happy to have you along for the ride :)

Every week you'll have a lab to complete. The labs *are* meant to be applications of the things you're covering in lecture and discussion, but they *aren't* meant to be completed within the span of a single lab section. We recommend you start the labs when they are released (before coming to your lab section!) and spend time in class working with TA's, tutors, and CSM mentors to clear up any misconceptions you have and get credit.

- This semester's lab checkoff policy differs from past semesters. First, you are required to complete all exercises before asking for a lab checkoff. To get your lab checked off, add your name and your partner's name to the checkoff queue. You must get your lab checked off in person within one week of release for full credit. You do not have to go to a particular assigned lab, any lab will be able to check you off. If you checkoff after the 1-week window, you will earn half-credit. Any lab submitted later than 2-weeks from release is worth no points, though we still recommend you do the lab for practise.
- Once you've been checked off you'll be given a checkoff token. Tokens are unique and cannot be used more than once. They are also only valid for the assignment they are generated for. It is your responsibility to submit the token to gradescope to mark your assignment as completed. Your assignment is considered complete when you submit your token, meaning if you checkoff within the first week but don't submit your token until the second week, you'll be given half credit.
- To submit, create a file named tokens.txt and add your token inside. Upload this file to the respective gradescope assignment. We consider sharing tokens to be cheating and will respond accordingly.

Each lab will begin with a few objectives; use them to reflect on your learning and guide your thinking! Here are the objectives for today's lab. (TSWBAT = "The student will be able to")

- TSWBAT describe and adhere to all course policies including the lab checkoff policy.
- TSW gain familiarity with unix and git commands
- TSW correctly sign up for github classroom, gradescope, piazza, and other course-related forms.

Sign-ups

If you added the course before January 11th, you should already be added to gradescope and piazza. If you are not, you can add gradescope with the entry code “MRKEYX”, and you can join piazza by clicking here.
For this course, we’ll be using github classroom to distribute and collect your programming assignments. If you do not have a github account, you should make one now. You can do so here.
After creating or logging into your github account, follow this link to accept our invitation to github classroom. Select your SID from the list (Do not skip this step…). If your SID is not on the list, let a TA know. Accept the lab zero assignment; you may need to check your email and confirm the invitation.

Course Policies

You can view the course policies on the class website. You are responsible for understanding and adhering to all policies throughout the course of the semester. In particular, please be sure you’re familiar with the grade breakdown, lab policies, and academic dishonesty policies.

Once you’ve finished reviewing the policies, take this quiz.

Unix Commands

If you took CS61A and CS61B, you likely have some experience with the terminal and terminal commands, but we’d like to list some less common commands here that may come in handy during the course. For a review of the basic unix commands, look over this guide. Be sure to read and understand section B along with the commands below, as you’ll use them all in the checkoff exercise for this lab.

As a reminder, when typing commands or file paths, you can use your Tab key to autocomplete the entry. You can also use CTRL + A to navigate to the beginning of your line to fix any mistakes (CTRL + E will go back to the end). Your up and down arrows will allow you to refill commands you’ve used previously without typing them over while CTRL+R will let you search through your recently used commands

touch, cat, and touch - Working with Files

```
> touch filename
```

This will create a blank file with the file name you provided. If you’d like to create a file with contents already inside, you can use

```
> echo “Your contents here, inside quotes”
```
This will print “Your contents here, inside quotes” to terminal. It will not create a file in the process.

```
> echo “Your contents here, inside quotes” > filename
```
This will create a file with the name filename in your current directory. If the file already exists, it will be overridden. The file will contain “Your contents here, inside quotes” but without quotes. The ‘>’ symbol takes one argument which modifies where values printed to stdout are piped. Here, we are redirecting them to a file named filename.

```
> cat filename
```
This will print out the file contents of filename to your terminal. The file must be in your current directory, but you can provide a relative or absolute path to print out non-local files.

ssh - “Secure Shell”

For this class, we’ll expect you to test most of your projects, homeworks, and labs on the hive machines (also called ‘instructional machines’). In order to access them, you’ll need to make an instructional account **for this class**. You can do so here. If you are a concurrent enrollment student, or have not yet been enrolled in the course, please fill out this form to obtain an account (note, this will not happen immediately but staff will let you know when your account has been generated). Once you have an account, you can ssh into the instructional machines with the following command:

```
> ssh cs61c-<your 3-letter login>@hive<1..30>.cs.berkeley.edu
```

Having trouble? Check your WiFi. You’ll only be able to ssh on AirBears2, not CalVisitor. Since the hive machines are used by many different classes, some of the machines may be overloaded or offline. To check the status of the hive machines, please visit https://www.ocf.berkeley.edu/~hkn/hivemind/. We suggest you pick one which is underutilised.

You’ll need to enter the password given to you after you created an account. If you’d like to change this password, you can run the following command:

```
> ssh cs61c-<your 3-letter login>@update.cs.berkeley.edu
```

scp - “Secure Copy”

This command is used for copying files from a ssh environment onto your local computer, or vice versa.

If you’re working on your own computer, you’ll need a way to get starter files for some of the assignments. We’ll use git for the most part, but you may occasionally want to get individual files or entire folders from the hive machines (or move files from your computer the other direction!)

You can do this with the following command:

```
> scp source destination
> scp cs61c-XXX@hive2.cs.berkeley.edu:~/my_folder/my_file.txt
~/Desktop/CS61C/lab0/
```

Note that in the source path, we specify <account><colon><path> to indicate our source is remote. The destination path is assumed to be local, so we don't need to specify an account. There is a space between source and destination.

If I wanted to copy the other direction (from my local machine to my hive account) I would use:

```
> scp ~/Desktop/CS61C/lab0/
cs61c-XXX@hive2.cs.berkeley.edu:~/my_folder/my_file.txt
```

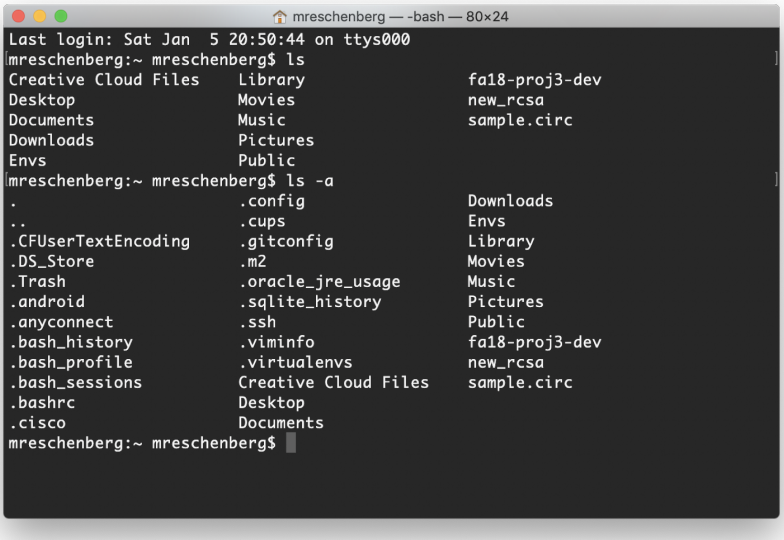
- Note: You shouldn't run this while you're logged into a ssh instance of your hive account! Run it locally in both cases.

ls -a - "List All"

```
> ls -a
```

You're probably familiar with the normal 'ls' command which can be used to list files. Adding the -a flag shows all files in a directory, including ones which are hidden and start with a period. You should run this before you add files to your git repo to avoid adding hidden system files. It is also helpful when debugging git as we'll see later on!

You can see the difference between running a regular 'ls' command and an 'ls -a' command in the same directory below:



Try it yourself. See anything cool? Morgan recommends running it in a git repo and poking around :)

man - Manual Pages

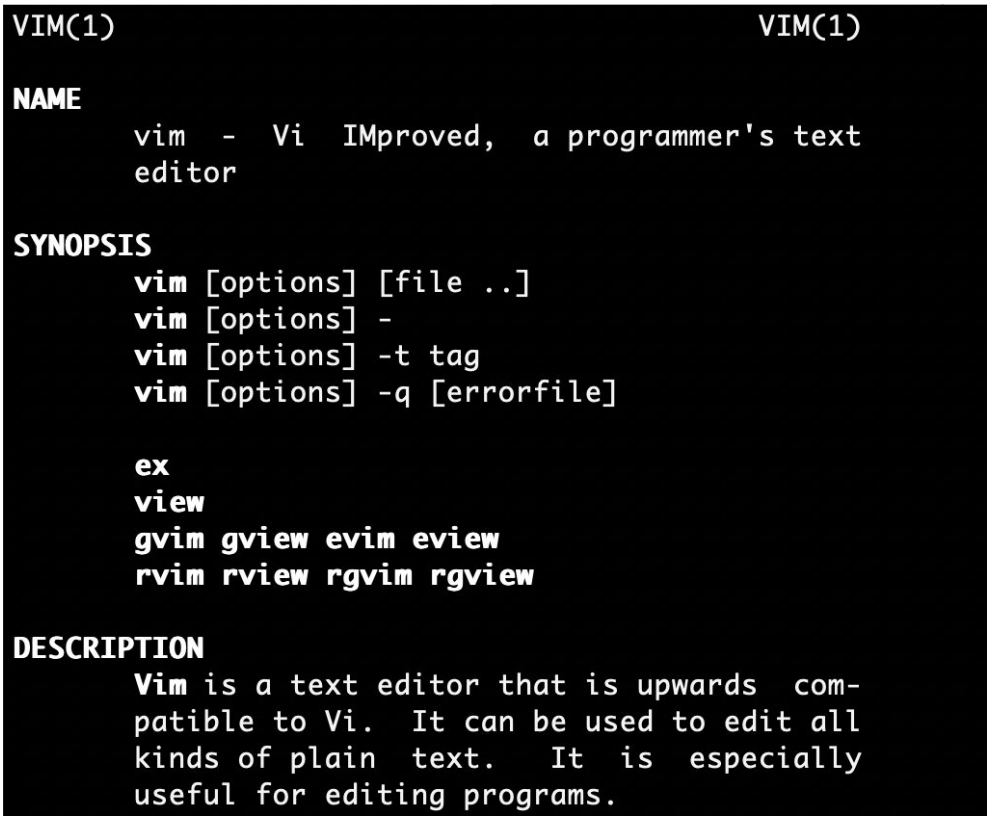
The manual pages are a great unix reference that is often underused; they hold the official documentation on everything from definitions of programming concepts to language standards and conventions. In this course we'd like you to get comfortable with them, especially for c- and unix-related questions.

`> man -k single_keyword | less`
This command will search the manual pages for a command that pertains to the keyword you mention. Forget how to open files in vim? You can search for editor and get a list of all related editor commands on your machine. When Morgan runs this on her machine, she gets the following:

```
mreschenberg:cunit mreschenberg$ man -k editor
ed(1), red(1)            - text editor
nano(1)                  - Nano's ANOther editor, an enhanced free Pico clone
pdisk(8)                 - Apple partition table editor
psed(1)                  - a stream editor
sed(1)                   - stream editor
vim(1)                   - Vi IMproved, a programmer's text editor
zshzle(1)                - zsh command line editor
Mach-O(5)                - Mach-O assembler and link editor output
mreschenberg:cunit mreschenberg$
```

`> man command_name | less`

If you know a command you'd like to look up, you can omit the -k flag and get the entry by itself. Say we used the above search to find vim, and now we want to read more about it. When we run the above command, we might get something like the following. This page is scrollable (use your arrow keys, or the space bar). Hit 'q' to quit the manual page and get back to your terminal prompt. It contains information about what the program is used for, what certain flags do when you invoke the program with them, and where to go for more information.



Vim Basics

Vim is a text editor included on the hive machines. It comes with most unix-based computers and you probably have it on your machine too! You can type vim to test if the app is available. Below are a series of commands we expect you to be able to use:
Note: like we saw above, there are many editors available for use; you're welcome to pick whatever you're comfortable with, but we'll expect you to know how to use vim in addition to whatever you choose to work in.

Command	Explanation

`vim file.txt`
 	
`vim ../tests/test.c`
	Opens a file in vim. The file must be in your current directory. If it isn't, you can also provide a

	relative or absolute path
<escape>:q	Closes (quits) vim without saving
<escape>:wq	Closes vim with saving
<escape>:w	Saves your file
<escape>:q!	Force quit vim (for when you've made changes but do not wish to save them)
<escape>i	Insert mode, allows you to type into the file
<escape>/dogs	Searches your file for the nearest occurance of the word 'dogs'. Press 'n' to go to the next occurance
<escape>:set number	Shows line numbers within your file
Note: these commands are preceded by <escape> because you'll need to hit the escape key to switch you out of your current mode. For example, if I'm inserting (typing) into a file and then would like to save, I'd have to hit <escape> to get out of insert mode, then type :w to save my file. If you aren't in a mode (ie. you've just opened your file) you don't need to hit escape first :)	

Git Exercise

This exercise will show you how to set up your repositories locally and on the remote, use vim, and work with a variety of git commands. By the end of it, you should feel comfortable ssh'ing, resolving merge conflicts, editing files, modifying your .git/config file, and pushing/pulling/committing. If you'd like to review your git commands before beginning, you can check out this guide Sruthi was kind enough to put together :)

First, ssh into your inst account. If you are unable to do so because you do not have an instructional account, you may complete the exercise below on your local computer. First, you'll need to use the man pages, as described above, to search for a command that will help you make directories. Once you've found it, create a new folder called 'lab0'. CD into this new directory.

View all hidden files. What do you see?

Init a new git repository (note: do not clone, you should init an empty repository.

View all hidden files. What has changed? How do you know if a directory contains a git repository?

What we have created is a local repository that has no remotes attached. If you try to push and pull, git won't understand where to put your information (also, the repo is empty and there is nothing to push).

Before we modify our repo, let's tell git who we are. This information will be used to sign and log your commits. You may not need to do this if you've set up git before, but if you're on the hive machines it's likely a step you'll need to take. Run the following inside your git repo; make sure to change the name and email to match your information.

```
> git config --global user.name "John Doe"
> git config --global user.email johndoe@example.com
```

Okay, now we'll start by adding remotes; these are 'trackers' that let git know where to push/pull when the respective commands are used. Each remote has a name and one or more branches associated with it. To add a remote, run the following inside your git directory (you may want to copy/paste this). Do this now:

```
> git remote add origin https://github.com/61c-teach/sp19-lab0.git
```

Once you've done that–oh wait hm. This isn't quite right. We gave you the link to our starter code, but we don't want you to push there. You'll need to set up your remote to point to your github classroom repository. Let's do that now.

In order to modify our remotes, we can remove and then re-add them, or we can modify our git configuration file. Let's do the second. CD into your hidden git folder, and vim the config file. Delete the existing URL and replace it with the correct one linking to your github classroom repo. You should only need to modify the URL. CD back into your lab0 directory once you're finished.

Great! Now that we've done that, we can pull our files from the remote. Go ahead and do that. List your files when you've finished. What changed?

Open 'first_set.txt'. This file contains descriptions of unix commands. Use the man pages to figure out which command they refer to; **be sure to include any necessary flags**. Once you've found them, write the commands on one line at a time in a new file called 'answers.txt'.

Add and commit this file with the commit message "Initial commit.". Push this to your github classroom repository.

Now, we'll need to switch over to your local computer. Exit your ssh session by typing 'exit'. You should be back at your own terminal prompt. If you're working on the (physical) hive machines, do the following in a different directory.

SCP your entire folder to your local machine or new directory. You may find it helpful to use the 'pwd' command to find your current directory/path. This will give you an absolute path to the files you wish to copy, meaning that the files should be locateable from anywhere. Relative paths on the other hand, locate files with respect to your current location within a directory.

Once you've copied the entire folder, CD into this folder and open your answers file.

*Note to linux users: you can run CTRL + SHIFT + C to copy from terminal (and the same args w/ V instead to paste).

Above where you wrote your answers, add commands (and relevant flags, if needed) that fit the following descriptions:

- Move files from one directory to the next
- Change a file's permissions
- Show the directory you're currently in
- Check if a host is online

Save this file but do not commit or push it.

Log back into your instructional account. In the same file, find this last command (and relevant flags, if needed) and list it at the top.

- Show file permissions for all files in a directory

Save, add, commit, and push this file.

Once you've done this, go back to your local environment or other directory, and add/push the file we were previously working. You should get a message from git. What does this mean? How do you resolve it?

Fix the issue and push the file.

Once you've done so, you should have all your answers in one file together on your local machine. The final ordering should be as follows:

- Show file permissions for all files in a directory
- Move files from one directory to the next
- Change a file's permissions
- Show the directory you're currently in
- Check if a host is online

- Which command walks a file hierarchy in search of a keyword?
- Which command displays information about processes running on your machine?
- Which command terminates a process?
- Which command can help you find the difference between two files?

Checkoff

- Run git log to show you correctly alternated between local/hive and that you solved the git issue above.
- Display your final list of commands for staff to check
- Describe the git command you'd use to roll back your directory to the initial commit.

Other tools you may find helpful!

These are some tools you may find helpful, but are by no means required for this course :)

Text Editor vs. IDE

CS61C doesn't endorse any particular text editor or IDE. Many people get by in this course using a text editor with no frills (think: vim/emacs/vi). We'll expect you to know how to use vim as it is the default editor on the instructional machines.

For your own work, you may find it nice to have CLion from JetBrains if you're used to working in IntelliJ for CS61B. Note though that we won't be providing any course-official support, so setting it up and maintaining it are up to you.

The majority of students do their work in a local editor (Sublime, Atom, VSCode) and use git to copy their files from their local machine to the hive. Some students also set up Cyberduck to make copying files over easier. Again, we won't provide any course-official support, but you're welcome to do what works best for you.

Byobu/Tmux

If you're working on the hive machines, you may run into 'Connection lost: broken pipe' errors which cause you to lose your session. If you'd like to be able to regain your session and support multiple terminal tabs across one ssh interface, you may want to install Byobu or Tmux. They're virtual terminal clients which add additional functionality to your normal shell.

Quick SSH

Tired of typing your entire ssh to get online? You can edit your config file in ~/.ssh to have the following contents (edit for your information):

Host 61c
HostName hive<1..30>.cs.berkeley.edu
User cs61c-XXX

Then, type ssh 61c and enter your password to login. You can add as many accounts as you'd like to this file.

You can also set up your account to not require a password by using your RSA key. This is the same key you can set up to not have to login to github everytime you make a commit/push! You can find the instructions here. Once you've generated a key, you can use it over SSH by following these instructions.