Lab 0: Introduction

Welcome to your first lab! CS15 labs run weekly and are an opportunity for you to get a hands-on approach to learning the lecture material. Labs are graded on completion and you may finish them any time within the next week - you can get checked off for a lab either during your lab section the week it’s released or during your lab section the following week.

Welcome to The Sunlab

You will likely be doing some of the coursework for CS15 here, so we hope this lab serves as a friendly introduction. The computers here run an operating system called Linux rather than Windows or Mac OS. It may feel a little unfamiliar at first, but Linux has many powerful features and you’ll quickly get used to the differences.

This first lab will cover:

- Setting up your CS department account
- Learning some basic information about using Linux
- Learn how to open and use commonly-used programs

Whew! That's a lot to do in one lab. It’s okay if you don’t understand everything covered here - you'll have all semester to perfect your Linux skills, so there’s no need to worry about memorizing every command.

Getting Started

- These boxes in each lab have instructions for you to follow.
- For labs, code will be represented with this font.
- Don't worry about clicking on the "wrong" thing: you don't have access privileges to break anything. :)

What We Assume

This lab assumes that you have basic familiarity with using computers — how to use a keyboard and mouse, a web browser, and navigation menus. If you have experience using Windows or Mac OS, this will all likely be familiar.

Please speak to a TA if you feel you do not have this level of background knowledge.
The Sunlab

First, a bit of background about the room you are in. It's called the "Sunlab" because it originally had computers made by a company called Sun Microsystems (incidentally, the same company that made Java!). The Sunlab is the primary on-campus workplace for CS undergraduates, and it's shared between the students in all of the department's courses.

The computers in the Sunlab are all custom-built machines that are constantly being upgraded by the CS Department. Treat the machines well and they will be your friends.

Getting Help

The TAs

Every lab section is proctored by several TAs who are there to help you with the lab material.

If you have any questions, please raise your hand and a TA will stop by to help as soon as possible.

If you've finished the lab within the 90 minutes, your TAs will check you off in the gradebook. If you're still working, you can finish the lab on your own time and get checked off during next week's lab.

The Course Website

The course website has an astonishing amount of information about the course's mechanics and material. Make sure to check the website regularly.

The Consultant

The computer in the corner by the entrance, numbered "cslab9a," is reserved for the Sunlab consultant. The consultant is there to help with any computer issues you may have (but not course material, even if the consultant is also a CS15 TA!). If you ever need help using the Linux environment or if you think that your computer isn't working the way it should, ask the consultant on duty.

Course Website

- Check out the course website to see what's there. You'll be referring to it a lot over the course of the semester. Some useful things to note:
  - When is the first CS15 assignment due?
  - Where are TA hours held?
Getting Started with Linux

User Interfaces

What’s displayed on the computer’s screen when you use it is a function of its "Graphical User Interface" (GUI), which defines the "look and feel" of the operating system. A GUI is the visual way you interact with a program. Typically, a GUI would include visual things like:

- Icons
- Mouse cursor
- Window borders and menus

and also includes basic behaviors of the interface, such as:

- What the mouse buttons do
- How you find and run programs
- How windows behave as you move them around and stack them

In Windows and Mac OS, you're more or less stuck with the standard GUI that Microsoft or Apple designed. In Linux, however, there are a many vastly different interfaces. Your account is probably set to use an environment called "Xfce" by default, so this tutorial assumes you’re sticking with that for now. You can play around with others using with the "Session" menu on the login screen later if you’d like.

When you first logged in, Xfce displayed the Desktop Environment. This is very similar to the desktop environments for Windows and Mac - a main page (the “desktop”) with some icons (also called “shortcuts” or “links”) to other programs or files. Along the top of the screen is a
menu bar to access basic commands like logging out. Along the bottom of the screen is the taskbar, which displayed the programs you’re running and allows you to toggle between them.

**The Terminal**

One of the most powerful features of Linux is that it gives you extensive access to go “behind” the graphics and work more closely with what the computer is actually doing. One way of doing this is through the use of a “terminal.” A terminal is a window that allow you to interact with your computer directly through written commands. The terminal itself is actually a GUI for a “shell,” the program that executes those commands.

### Opening Terminal

- To open a terminal, click the Applications menu in the top left corner of Xfce, click System, then click Xfce Terminal. (For future reference, you can also open a terminal by right clicking anywhere and clicking “Open Terminal Here”)

- **Note:** You’ll be opening new terminals often, so you can create a shortcut by dragging the terminal icon to the sidebar on the left within the Activities Menu.

Welcome to the Linux terminal! You’ll see a new window with something like:

```plaintext
  cslab5f ~ $
```

displayed on the first line. The first part is the name of the computer, the “~” is an abbreviated file path (more on this soon), and the “$” is called the “command prompt” - which prompts you to write commands.

**Shell Commands**

Let’s get started! A command typed into a shell may have up to three parts, depending on the command and how it’s used:

1. The command name
2. Modifications to the command (called "flags")
3. Things to perform the command on or with (called "arguments")

Not all commands require all three parts, but they all must contain a name.
Checking Your Groups
Before we get started using the terminal, let’s check to make sure that you are in the CS15 group.

Group Check
- At the command prompt, type:

```
$ groups
```

And press “enter.” You should be able to see `cs015student` listed as one of your groups.

If not, call over a TA, who will get a consultant with systems privileges (SPOC) to add you to the student group.

The man command
The first command that you may find to be incredibly useful is a command to learn more about other commands (how meta!)

The command `man <command_name>` display the “manual pages” for the command given as an `argument`. The manual pages include a description and other relevant information.

Man the Terminal
- Before continuing, type:

```
$ cd ~
```

And press “enter”. This makes sure the terminal starts off in your home directory - we’ll explain what this is shortly!

- At the command prompt, type:

```
$ man man
```

And press “enter.” You can navigate through the manual pages with the up and down arrow keys.

- When you are finished, press `q` to quit.
**pwd & A Quick Introduction to the Linux File System**

Let's take a look at a sample command: `pwd`, which stands for “print working directory.” Like the Windows Explorer and Mac Finder, Linux terminals have the concept of a "current location" when browsing the file system. Like the file system on Windows and Mac, files on Linux are organized hierarchically, with directories (also known as folders) located one within another.

One difference with Linux is that all files and folders are contained within a "root" folder, no matter what disk or drive they're on. This is great for a network setup like we have here in the Sunlab — you don't care what physical drive the files are physically stored on as long as you can access them. This "root" directory is just called "/".

The “path” to a file or directory (i.e. its location in the file system) can be specified as either absolute or relative. An absolute path leaves no ambiguity about its location, and is defined with the root directory "/" as the reference point. A relative path, however, is defined from the current directory as a reference point.

All subdirectories are also separated by a `/`, so a directory called "MyAwesomeDirectory" in root would be represented by the absolute path `/MyAwesomeDirectory/`.

Likewise, the absolute path `/stuff/things/my_thing.txt` would represent a text file called "my_thing.txt," which is contained in the directory (folder) "things," contained in a directory "stuff," contained within the root directory "/." If your shell was already located within “stuff,” the relative path `things/my_thing.txt` would have the same meaning.

Introduce yourself to a neighbor and discuss which one of these is an absolute path and which one of these is a relative path.

1. `/hop/pop/stop`
2. `lop/top/mop/bop`

Two directories on the department system that you will use often are:

- `/home/your_login/` (for example, `/home/dziring`): Everyone with a department account owns a "home" directory within `/home/`. This and its subdirectories are where you should store your own coursework and files. Only you should have access to your home directory.
- `/course/cs015/`: This is a directory where the CS15 support files are stored. It’s owned by the CS15 staff.
The location of the current shell (called the working directory) can be found by using the command `pwd`, which stands for "print working directory".

## Print Working Directory

- At the command prompt, type:

  ```
  pwd
  ```

  And press “enter.” (Remember, you can enter the command `man pwd` to see more information.)

By default, new shells will start in your home directory, which is what the results of the command show. (Note that you can also see the working directory to the left of the command prompt. The home directory is often abbreviated with the tilde character, ˜.)

## An Exploration of Shell Commands

Let’s try out some of the most commonly used terminal commands. Note: press enter after each command to enter it.
Navigating Directories

- At the command prompt, type:

  `ls`

The `ls` command lists the contents of a directory. If you specify a directory as an argument, it will list the contents of that one, but otherwise it will show the contents of the current (or working) directory.

The list includes both files and directories (which you can distinguish by the “/” at the end of the name, for example, `Desktop/`).

- Type:

  `cd Desktop`

The `cd` command stands for change directory. Your shell’s working directory is now `~/Desktop/` (which is the same as `/home/<login>/Desktop/`). You should see this change reflected to the left of the command prompt.

Syntactic Sugar for Directories: “.” and “..”

Linux also provides a shortcut for you to refer to the current directory and its parent directory when working with relative paths. The symbol “.” refers to whatever the current directory you’re working in is, and “..” refers to its parent.

Directory Hierarchies

- At the command prompt, type:

  `ls ..`

You should see the same list of contents from before, because “~” is the parent of `Desktop/`.

- Type:

  `mkdir NewDirectory`

The `mkdir` command is short for make directory. It takes the desired name of the directory as an argument.
Minimize any screens you have open to take a look at your Desktop. You should see an icon for your new directory there as well!

- Type:
  
  ```
  cd NewDirectory
  ```

  Then type:

  ```
  ls
  ```

  Soon this combination of entering a directory and listing its contents will become almost second nature! In this case, nothing should show up, since the directory you just made is empty.

- Type:
  
  ```
  touch MyNewFile
  ```

  Then type:

  ```
  ls
  ```

  touch creates a new file (among other things), which you should now see listed.

- Type:
  
  ```
  rm MyNewFile
  ```

  To delete your new file. Type `y` to confirm the removal of the file.

- Type:

  ```
  cd ..
  ```

  To change back to your home directory.

- Type:

  ```
  rmdir NewDirectory
  ```

  To remove the now-empty directory you’ve made.

- Type:

  ```
  ls
  ```

- Your files and shell are now in the same state that we started off in. Nice!
Running Programs
Let’s move on to running programs. From the Linux shell, you can launch familiar applications like web browsers and word processors. Many of these can be launched through the Applications menu, but it's often more convenient to use a terminal.

Due to copyright reasons, these Linux machines run a version of the Firefox web browser called “Iceweasel” (punny, right?).

The WorldWide Web
- At the command prompt, type:
  ```
  iceweasel &
  ```
  And press “enter.”

Foreground/Background Processes
By default, when you run a program from the command line — either a simple UNIX utility like ls or a full application like Firefox — the program runs in the foreground, meaning the Terminal will wait for the process to finish running before a new process can begin. For a very short process, waiting could be ok, but when you open an application like a web browser, you’re probably going to want to use it for a substantial amount of time.

In contrast, you can instruct the shell to open the program in the background by adding a & symbol to the end of the command, like you did above. Having the program run in the background means that the Terminal will free itself up to accept new commands.

Background Noise
- To see the difference, try opening `iceweasel` again, but without the &.
- Look at the terminal. You'll see that you don't get a command prompt.
- Close the browser, look back at your terminal, and you’ll see that you should have a new prompt.
**Hint:** In general, if you're opening an application, it might be advantageous to run it in the background with `&`.

**Important:** When you close a terminal window, all the applications you opened from that terminal window will close without prompting you to save your work. Closing the terminal **kills** the Terminal and everything running out of it.

## Writing Java Programs

Let's begin by noting that you can write a Java program with any text editor you want. You just need working Java code (which you will learn to write) and a way to **compile** your code. There is nothing "magical" about the text editors you will use in CS15, except that there are a lot of helpful programming tools built on top the text editors you will be using.

The text editor that we will use for now is Atom. Atom is just one of many editors that is available to you.

### Let’s try opening Atom:

- Type `atom &` to open the text editor.
- **Note:** If you want to open a specific folder or file in Atom, you can go to menu bar and click File > Open Folder or File > Open File and navigate to the folder or file that you want to access.

## File Extensions

In Linux, file extensions (the part of the filename after the ".", e.g. `.java`, `.txt`, `.docx`) are completely optional, and serve no purpose other than helpfulness for the user. For example, by convention text files that contain Java code end in `.java` (e.g. `MyJavaFile.java`)—which can make it easier to identify files, but it makes no difference to the computer.

One nice thing about using extensions, though, is that if you want to open all the Java files in a particular directory, you can refer to them all using the command `*.java`. The asterisk `*` is another Linux shorthand called a "wildcard" and stands for any number of characters.
So if you typed:

    atom *.txt &

Atom would open all the files in the current directory that ended in ".txt". Likewise,

    atom My* &

would open all files in the current directory that started with "My".

**Using Atom**

Here is what Atom looks like with two open Java files (don’t worry about understanding the code below):

1. The working directory. Looks like, `pwd`, right?
2. You can use these tabs to switch between files you are actively editing. The in the first tab shows that the document has unsaved changes, whereas the other tab has no unsaved changes.
3. The menus across the top of the window have basic commands like "open," "save," "new file," etc.
4. This side menu shows the folder you are in as well as the files in that folder.

The above capabilities are standard amongst most text editors (like TextEdit for example). Atom has a lot of additional functionality that makes it pleasant to code with.

- Atom has **syntax highlighting**, which means that different words are automatically displayed in different colors, depending on their meaning. This makes it easier to read code.

An example of syntax highlighting:

<table>
<thead>
<tr>
<th>In a normal text editor:</th>
<th>In Atom:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public static void main(String[] args){</code></td>
<td><code>public static void main(String[] args){</code></td>
</tr>
</tbody>
</table>

- **Code folding**, which is a way of showing and hiding parts of code temporarily, so that it isn't distracting. You can see the little arrow icons in the left margin of the above screen showing where you could click to *fold* the code (essentially, just minimizing it until the arrow is clicked again).

- Atom has a very smart autocompletion engine built into it. This means that as you are typing, it will attempt to predict what you're trying to type, and pop that up as a suggestion. If you hit tab, it will fill in the rest of the word highlighted in the suggestion box.

Unfortunately, you can't really play with some of these features until you have some code to write, which will have to wait another week or so. In the meantime, rest assured that Atom is tons o' fun.
Removing a Directory
In order to remove a project folder from atom, you have to right click the folder and click "Remove Project Folder", not "Delete." This is shown in the screenshot to the right.

Writing, Compiling and Running Java Programs

Whenever you write a Java application, you start by editing .java text files. You compile your code by running those files through the javac compiler. Then, your program is ready to run.
You'll learn more about the structure of a Java application as the semester progresses. For now, let's dive into an introductory example:

Hello World!

- Type cd to return to your home directory, if you're not already there.
- Run cs015_install lab0. This will create a directory for lab0 in your cs015 directory.
- Go to your lab0 directory using cd course/cs015/lab0.
- Type touch HelloWorld.java to create a new file.
- Open the newly created file in atom and copy in the following code:

```java
package lab0;

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

The System.out.println() command prints out text into your terminal. This is a great way of confirming your program's success.
Note: For now, don't worry if formatting is messed up when copying.

- Save the file, then compile your code by typing javac *.java in your terminal
  - If your code compiled successfully, nothing will print in your terminal! A new file called HelloWorld.class will appear in your directory (you can check with ls).
  - If error messages did print out, you may need to re-type the semicolons or other characters in your file (some PDF viewers display characters differently).
● After you compile your code successfully, run your code by typing `java lab0.HelloWorld` in your terminal

● Looking in your terminal you should now see the text “Hello World!” printed out.

Congratulations! You just compiled and ran your first Java application!

Please read the CS15 Collaboration Policy, which you can find here http://cs.brown.edu/courses/cs015/docs/collab.pdf, then complete the collaboration contract form linked to below. The form will prompt for your digital signature. When you get checked off, be prepared to discuss the collaboration policy with a TA and show the TA your form response confirmation.

https://docs.google.com/forms/d/e/1FAIpQLSd0YfYX3ZrzwzJcaNlspdG__X6vm_7GR8nxpisMQ9P5LTAdA/viewform

You will not be able to receive any points for any assignment until you have signed the collab contract.

You’re done! Remember to logout before you leave

To logout, click on the top right corner, then your name, then logout. You can also use the shortcut “Ctrl+Alt+Backspace” to log out quickly.