CSCI 2132: Software Development

Lab 1: Getting Started in the Unix Environment

Synopsis

In this lab, you will:

- Start using Unix
- Learn to use basic Unix commands
- Write a simple Java program on Unix
- Learn how to navigate and move around files and directories
- Submit your work electronically using SVN
- Learn a little about the `head` and `tail` commands

Contents

Overview

In this lab, you will learn how to use UNIX and some basic UNIX commands. Strictly speaking, you will be working on a Linux system, which is one of several UNIX-style systems. The word UNIX, with all capital letters, is a registered trademark and refers to the original UNIX implementation. Since we talk about UNIX-style systems in genenal, such as Linux, and their commands, we will use the word Unix, without full capitalization, to refer to this general environment.

You will also write a simple Java program on Unix. The purpose is not to improve your Java programming skills, which you have already gained in previous courses, but to help you get ready to program in C on Unix. You will learn the basic use of `emacs`, a common text editor on Unix systems. You will take your first basic steps in navigating the file directory structure, creating new directories, and copying files and directories. The main commands you will use are:

`pwd:`	print your current working directory
`cd:`	change your current working directory
`mkdir:`	create a new directory
`mv:`	move or rename a file or directory
`ls:`	list the contents of a directory
`chmod:`	change the permissions of a file or directory

`chmod` is a complex command that you will learn more about later in class. You will use it only for one basic but important step here.

You will also learn to use a few Subversion (SVN) commands to submit your work:

`svn co or svn checkout:`	get a working copy of your SVN course repository
`svn add:`	add files or directories to the set of files and directories to be submitted to SVN
`svn commit:`	submit your work to SVN

Finally, you will learn to use the `man` command to learn about the `head` and `tail` commands.

Be sure to get help from teaching assistants whenever you have any questions.

An unapologetic note about the command line and text editors

You will spend much time using the Unix command line in this course and you will use `emacs` to complete your text editing tasks. Both will feel alien if you have mainly used Windows `Explorer` or macOS `Finder` to navigate the directory hierarchy of your computer and you have used IDEs, `Notepad`, `Atom` or `VisualCode` for your editing work. However, the command line and seemingly ancient editors such as `emacs` and `vim` are the power tools in your arsenal and are worth the effort to learn. There is a reason why they have survived for so many years and continue to be used by professionals. Both `emacs` and `vim` are also available for Windows and do have GUI versions.

A carpenter uses a nail gun to hammer nails into the wood whereas you or I would use a manual hammer for our little home improvement projects because a nail gun is too great an investment for our limited needs whereas a carpenter spends much of his day hammering nails into wood and a nail gun is a worthwhile investment to make his work more efficient. Similarly, `Notepad`, `Atom` and `VisualCode` (even though the latter also has some neat "power user" features) are great for casual text editing and are easy to learn. For complex text transformations, however, they are inadequate and `emacs` or `vim` can often perform them with ease. Since you will spend much of your working day editing code or text in general, it is worthwhile to learn these power tools.

A similar comment applies to the Unix command line. You need to learn basic Unix shell commands to use it effectively, but for complex work flows, it is almost always the more efficient tool to work with compared to graphical file managers such as `Explorer` or `Finder`, which do not allow you to automate anything. It is another power tool in your arsenal as a programmer.

With this sermon out of the way, let's get started!

Step 1: Log into `bluenose`

If your computer is a Linux or macOS computer (or a Windows computer running `cygwin`), you can often perform the tasks required to complete the labs in this course on your own computer, without connecting to `bluenose`. Note, however, that TAs or the help desk are not responsible for helping you set up the right working environment on your own computer and you should also learn how to connect to other computers using `ssh`. You are therefore encouraged to log into `bluenose` even if you have a Unix system at your disposal, and perform your work directly on `bluenose`.

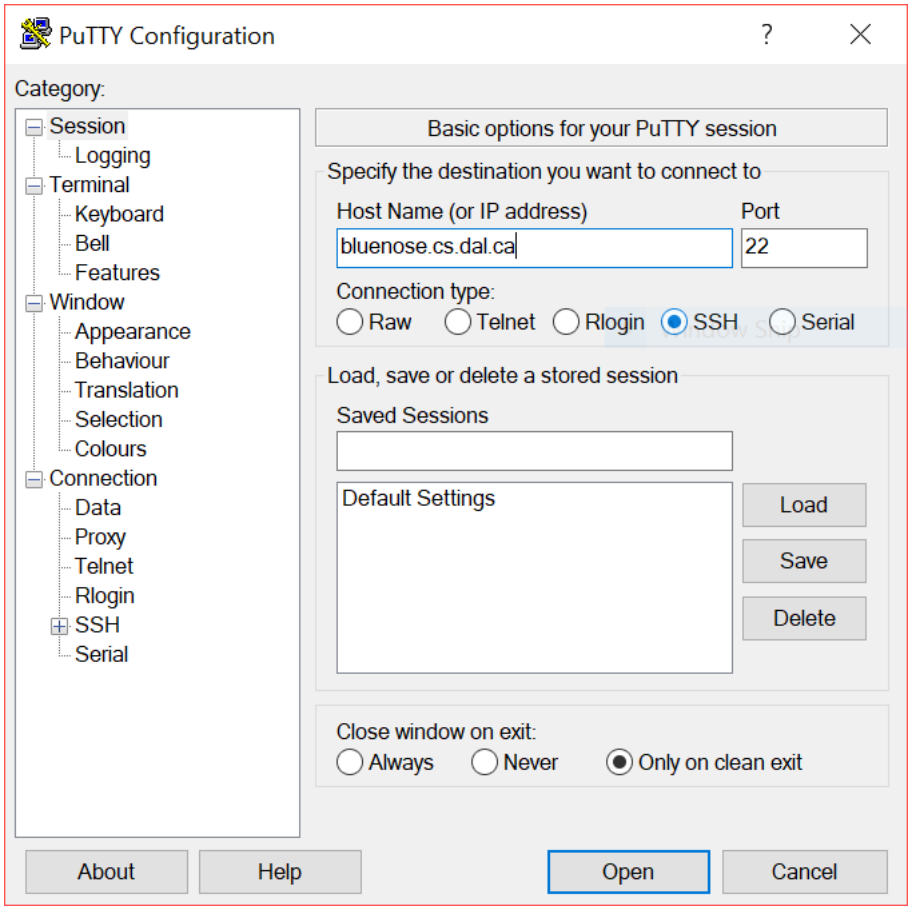
Depending on the type of computer you use for the lab, follow one of the following sets of instructions to connect to `bluenose`:

Windows:	Continue to Step 1w: Log in using `PuTTY`.
Unix (Linux, macOS):	Continue to Step 1u: Log in using `ssh`.

Step 1w: Log in using PuTTY

You will need a terminal emulator to connect to a Unix server from Windows. Two options that are available are `MobaXTerm` and `PuTTY`. Here, we discuss how to use `PuTTY` because this is the terminal emulator installed on the Windows machines in the lab.

Double-click the `PuTTY` icon. The following window should appear:



You need to fill in the host name `bluenose.cs.dal.ca` yourself. Also make sure that `SSH` is selected as the connection type and that the port is `22`. Then click `Open`. A terminal window should open with the prompt

```
login as:
```

Enter your CSID and press `Enter`. Next you are asked for your `bluenose` password. Enter it and press `Enter`.

Step 1u: Log in using ssh

If you use Linux (your own computer), I expect that you know how to start a terminal on your computer. On macOS, you should start the `Terminal` application. One way to do this is to click on the search icon in the top right corner of the screen. This opens Spotlight, macOS's search tool. Type `Terminal` and press `Enter` to start a terminal. Another way to find the `Terminal` application is to look in the Mac applications folder.

With `Terminal` open, log into `bluenose` by typing:

```
$ ssh CSID@bluenose.cs.dal.ca
```

The `$` sign is the prompt (do not type it). The prompt may look different, such as `user@hostname:~$`. `CSID` should be replaced with your CSID. I would type:

```
$ ssh nzeh@bluenose.cs.dal.ca
```

Step 2: pwd

By default, when you log in, your current working directory is your home directory. All pathnames without a leading slash are interpreted relative to your working directory. So, immediately after logging in, `file.txt` refers to the file `file.txt` in your home directory and `Documents/doc.tex` refers to the file `doc.tex` in the subdirectory `Documents` of your home directory. Type

```
$ pwd
```

and write down the output of this command displayed in your terminal.

Step 3: `mkdir`, `ls`, and `chmod`

Create a subdirectory of your home directory named `csci2132`:

```
$ mkdir csci2132
```

You will use this directory to store all your work on labs and assignments in this course. If you enter

```
$ ls
```

to display the contents of your home directory, you should see `csci2132` as part of the output.

The next step is **very important**. After running `mkdir csci2132`, the directory `csci2132` is accessible by **everyone** who has access to `bluenose`. Since you will store your assignment answers in this directory, you should ensure that only you can access this directory; **your classmates should not be able to access it.** To ensure this, run:

```
$ chmod go-rwx csci2132
```

This removes all access rights to this directory from everybody except you. To check that the operation succeeded, run

```
$ ls -ld csci2132
```

The output should look something like this:

```
drwx------. 2 nzeh csfac 2 Dec 26 13:37 csci2132
```

The user name `nzeh` and the group `csfac` will be replaced with your CSID and your user group. The important part is the start of the line `drwx------.`, which indicates that `csci2132` is a directory (`d`), you can read from it (`r`), write to it (`w`), and change into it (`x`), while everybody else cannot access this directory at all (`------`).

Step 4: Create the `lab1` directory

The next step is to create a directory where you store the work you do in this lab. This directory will be called `lab1`. First switch into your `csci2132` directory:

```
$ cd csci2132
```

Verify that your working directory is now `csci2132`. Which command have you learned to check this?

Create a subdirectory named `lab1` inside the `csci2132` directory. Recall which command you use to create a directory. Use `ls` to check that the `lab1` directory has been created successfully.

Finish this step by switching into the `lab1` directory:

```
$ cd lab1
```

Step 5: Write `HelloWorld.java` using `emacs`

Throughout this course, you have two choices of editors to work with, `emacs` or `vi` (`vim`). If you are familiar with `vi`, you can complete all your labs using `vi`. However, `emacs` is the official editor used in this course, so TAs are only expected to be able to answer questions concerning `emacs`, not `vi`. If you are familiar with Unix and have used `nano` or `pico` before, you should learn to use `emacs`. `pico` and `nano` are very simple editors that may be suitable for occasionally changing a few lines in a file but are not up to the task of more serious text manipulations required when programming or editing larger bodies of text.

Your task in this step is to implement a simple Java program, the ubiquitous `HelloWorld.java`, whose source code looks something like this:

csci2132/lab1/HelloWorld.java

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

You should use `emacs` to create this file. You have three ways to learn how to take your first step in `emacs` (apart from a plethora of `emacs` tutorials available online):

- Read pages 69–75 in the Unix textbook.
- Read enough of the tutorial found at `http://www2.lib.uchicago.edu/keith/tcl-course/emacs-tutorial.html` to feel comfortable using `emacs`.
- Start `emacs` from the command line using the command `emacs` and then follow the built-in tutorial. To start this tutorial, type `C-h` `t`, where `C-h` means you press the `Control` key and `h` simultaneously. Most `emacs` commands are accessed by using the `Control` or `Alt`/`Meta` key as a modifier. (The `Alt` key on common keyboards is referred to as the `Meta` key in `emacs` parlance.) For example, to access a command prompt where you can enter arbitrary `emacs` commands, press `M-x` (`Meta` + `x`). You leave this "minibuffer" at the bottom of the screen by pressing `C-g`. This is also the key you use to abort any other emacs command. To exit emacs, press `C-x` `C-c`.

Now go ahead and create the above file `HelloWorld.java` in `emacs`: Start editing the file using

```
$ emacs HelloWorld.java
```

Once you are done editing, exit `emacs` using `C-x` `C-c` and answer `y` when `emacs` asks you whether you want to save the file.

Step 6: Compile and run a Java program

Back at the command line (after exiting `emacs`), compile your program using

```
$ javac HelloWorld.java
```

If there are compilation errors, open `emacs` again and fix the errors until `javac HelloWorld.java` completes without errors. You should now have a file `HelloWorld.class` in your `lab1` directory, which you can check using `ls`. To run your program, run

```
$ java HelloWorld
Hello, world!
```

Step 7: Use `emacs` for search-and-replace

The next exercise lets you practice replacing strings in `emacs`. You will rename your `HelloWorld` program to `HiWorld` and, accordingly, make it print "Hi, world!". Perform the following steps:

- Make a copy of `HelloWorld.java`:

  ```
  $ cp HelloWorld.java HiWorld.java
  ```

  The two arguments specify the source and destination files of the copy operation.

- Open `HiWorld.java` in `emacs`:

  ```
  $ emacs HiWorld.java
  ```

- Enter `emacs`'s command prompt by pressing `M-x`. The cursor should jump to the bottom of the screen and you should see the prompt `M-x`. If this does not work, pressing `Esc` `x` should work. (Note that this means that you press `Esc` first and then `x`; you do not press them together.)
- Now enter `replace-string` and press `Enter`.
- You will be asked for a string to replace and then for a string to replace it with. Enter `Hello` as the string to replace and press `Enter`. Then enter `Hi` as the string to replace it with and press `Enter`. This should replace all occurrences of "Hello" with "Hi" throughout the file.
- Save the file using `C-x` `C-s` and then exit `emacs` using `C-x` `C-c`.
- Compile the file `HiWorld.java` using `javac HiWorld.java` and run the resulting program using `java HiWorld`.

Instead of calling the `emacs` function `replace-string` from `emacs`'s command prompt, you could also start the search-and-replace process by pressing `M-%` or `Esc` `%`. Try it out.

Step 8: Prepare for using Subversion (SVN) to submit files

In this step, you will learn how to submit your work. You will submit the folder `lab1` containing the two files `HelloWorld.java` and `HiWorld.java` but no other files that were created in the previous steps.

First create a subdirectory `svn` of `csci2132`, which you will use to interact with SVN throughout the term. Since your current working directory is `csci2132/lab1`, you should go up one level and check that you are in the right directory:

```
$ cd ..
$ pwd
/users/cs/CSID/csci2132
```

Now create the `svn` subdirectory and change into it:

```
$ mkdir svn
$ cd svn
```

Check out a working copy of your course SVN repository using

```
$ svn co https://svn.cs.dal.ca/csci2132/CSID
```

Again, CSID needs to be replaced with your CSID. I, for example, would write

```
$ svn co https://svn.cs.dal.ca/csci2132/nzeh
```

You need to enter your CSID password. The command will ask you whether to `Store password unencrypted (yes/no)?`. **Answer no!** We will discuss Subversion in detail later in the course. For now, it is sufficient to know that the SVN server at `svn.cs.dal.ca` stores the repository you will use to record your assignment answers. The `svn co` line creates a working copy of this repository under your `csci2132/svn` directory.

Important note: *The above step will not work if you are not registered in the course. Only registered students have a course repository on* `svn.cs.dal.ca`*. If you are registered and still cannot complete this step, please send an email to* `nzeh@cs.dal.ca`*. After sending this email, it will still take a little while before your repository becomes accessible.*

Now, to finish preparing for submitting your work, change into the `CSID` subdirectory created using the above `svn co` line:

```
$ cd CSID # Again, CSID needs to be replaced with your CSID
```

Your current directory should now be `/users/cs/CSID/csci2132/svn/CSID`. Check using `pwd`.

Step 9: Submit your files

Create the subdirectory `lab1`:

```
$ mkdir lab1
```

Copy the Java source files from `~/csci2132/lab1` to `~/csci2132/svn/CSID/lab1` and check that they have been successfully copied:

```
$ cp ../../lab1/*.java lab1/
$ ls lab1
HelloWorld.java HiWorld.java
```

Submit your work to the SVN repository:

```
$ svn add lab1
$ svn commit -mlab1submitted
```

Do not use any spaces in the `-mlab1submitted` part of the second line! If you want, you can check that you properly submitted the files by using your web browser and opening the URL `https://svn.cs.dal.ca/csci2132/CSID`, where once again, CSID is your CSID. We will later discuss other ways to check that you successfully submitted your work directly from the command line.

Step 10: head and `tail`

As a final exercise in this lab, you should familiarize yourself with the `head` command. This command allows you to display the first few lines in a file, which is often enough when trying to find a file based on its content. Its counterpart `tail` can be used to display the last few lines in a file and also to monitor changes to a file, which is useful for observing the output written to a file by a running program.

Learn about both programs:

```
$ man head
$ man tail
```

To finish displaying a manpage, press `q`. Find out what the `-n` option of `head` and `tail` does. A description that may be easier to understand than the manpage can be found at `http://en.wikipedia.org/wiki/Head_%28Unix%29`.

The next steps

The official part of this lab is finished. You are strongly encouraged to continue practicing the use of `emacs`. Use Google and the built-in documentation to learn about more advanced editing commands and get ready for editing larger files as you will do in future labs and assignments.