Faculty of Computer Science, Dalhousie University *06-Sep-2018*

CSCI 2132 — Software Development

Lab 1: Getting Started in Unix Environment

Location: Teaching Labs Instructor: Vlado Keselj
Time: Thursday

Lab 1: FCS Computing Environment

Lab Overview

– Learning Objectives: In this lab, you will:
– Learn how to start using UNIX (i.e., Linux or Unix-style system)
– 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 bit about the 'head' command

In this lab, you will learn how to start using 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 general, such as Linux, and their common commands, we will use the word Unix without full capitalization to refer to this general environment.

You will also be asked to 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. In this step, you will learn basic use of 'emacs', a common text file editor in Unix environment.

In the process, you will make basic steps in navigating the file directory structure, creating new directories, and copying files and directories. The main commands that you will use for this are:
pwd — to print your current working directory,
cd — to change your current working directory,
mkdir — to make a new directory,
mv — to rename or move a file or directory,
ls — to list contents of a directory, and
chmod — to change permissions of a file or directory.

The command 'chmod' is a complex command and we will learn more about it in the class later, but we will use it here only for one basic step.

You will learn a basic use of a few SVN commands needed to submit your work:
svn co — or 'svn checkout' to get a working copy of your SVN course repository,
svn add — to add files or directories (folders) to the set of files and directories that are to be submitted to SVN, and
svn commit — to submit your work to SVN.

At the end, you will use the 'man' command to learn a bit about the 'head' command and use the 'head' command.

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

Step 1. Logging in to server bluenose

- You can choose Windows or Mac environment in some labs
- Windows: you will use `putty` program
- On Mac: open a Terminal and type:
  `ssh` *CSID*`@bluenose.cs.dal.ca`
- Instead of *CSID* use your CS userid (CSID)
- On Linux: similarly to Mac, you open the terminal and type the same command:
  `ssh` *CSID*`@bluenose.cs.dal.ca`

You can generally use your computer, such as laptop, or a lab computer for lab work. If you are using a lab computer, your first step is to login to the lab computer, and after that login to the `bluenose` server. If you are using your own computer, then you can directly login to the `bluenose` server.

You may generally work from a Windows, Mac, or Linux computer, and you can use an 'ssh' login from any of them to the `bluenose` server.

On Windows: If you use a Windows environment, you will need to use a program named `putty` to login to the `bluenose.cs.dal.ca` server. This will be explained in the next step. The PuTTY program has already been installed in the lab machines. You can also install it on any other machine since it is freely available on the Internet.

On Mac: In Mac OS environment, you can click on the search image in the upper right corner and type 'Terminal' to find the `Terminal` application. Another way to find the Terminal application is to look into the Mac applications folder. Once you open the terminal, you can login to the `bluenose` server by typing:

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

where `CSID` is your CS userid.

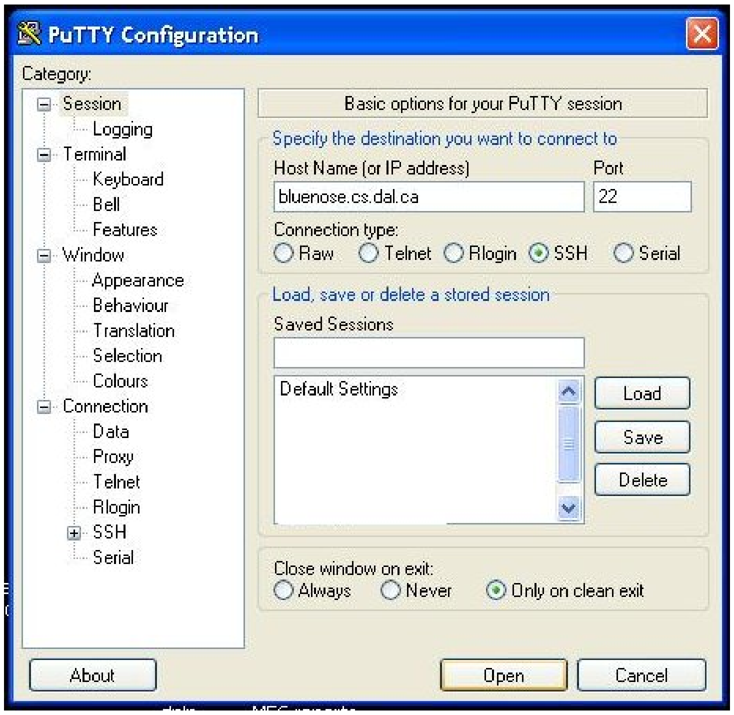
On Linux: In a Linux environment, you should, similarly to Max, open a terminal and type:

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

where `CSID` is your CS userid.

Running PuTTY

- Double-click the PuTTY icon, and the following window should appear:



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): bluenose.cs.dal.ca
Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions

Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

You should fill in the basic information: `bluenose.cs.dal.ca` for the Host Name. Make sure that the port number is 22; i.e., Connection type is SSH. You click 'Open' and the login process should start. You are likely to receive a warning about an unknown host key. Normally, this is something that you should be careful about and try to make sure that the offered fingerprint matches the fingerprint of the server, but in a relatively secure network you can accept this connection. Once accepted, the host key is stored with PuTTY and this warning should not appear again.

Step 2: pwd. Now, by default, you are in your home directory (or home folder). In other words, after logging in your current working directory is your home directory. One role of the current working directory is that when you use a command that uses a file name and you specify a file name without giving a path to it, the command will try to locate the file in the current working directory. Enter the command:

```
pwd
```

to show the path name of your current working directory. Write down the output of this command displayed in your terminal.

Step 3: mkdir, ls, chmod. Enter the following command to make a new directory in your working directory for your CSCI 2132 course work:

```
mkdir csci2132
```

Enter the command:

```
ls
```

to display the files (including subdirectories) in your working directory. Do you see the name of the directory you just created?

Now, enter the command:

```
chmod go-rx csci2132
```

This command will prevent other users from entering and seeing the content of this directory. You can check that the 'r' and 'x' permissions are off on this directory by running the command:

```
ls -l
```

We will explain later in more detail the information shown by the command "`ls -l`", but for now, you can just take a look at the beginning of the line containing the directory name `csci2132`. At the beginning of this line, you should see `drwx------.`, which means that the owner (yourself) have rigths to read, write, and enter this directory, and other users on the system do not have these rights.

Step 4: lab1 directory. Enter the command:

```
  cd csci2132
```

to change your current working directory to the directory you created in the previous step. Which command among those learned in previous questions, can you use to verify that you have performed this task successfully?

Create a directory with the name `lab1` in your current working directory, and change the current working directory to the directory `lab1`. Verify that you have performed these tasks successfully.

Step 5: Prepare `HelloWorld.java` in emacs. Now we are going to write a Java application on Unix using Unix utilities and applications. This Java program will be the typical "Hello world!" program which prints a line saying "Hello, world!".

The first step is to write the source code, and we will use the `emacs` editor for this task. It should be easy to start using emacs, based on a few basic instructions that we will explain soon, but you should learn more about it and below are some references.

Read pages 69–75 in the Unix textbook to learn how to use `emacs`. If you do not have the textbook yet, read the following longer tutorial (read enough till you feel comfortable about using `emacs` for a small program):
`http://www2.lib.uchicago.edu/keith/tcl-course/emacs-tutorial.html`
Third way that you can learn Emacs is to simply type `emacs` in your PuTTY terminal. This will start the editor, with an initial 'splash' screen explaining basic commands. You can notice that Emacs uses notation C-x for Ctrl-x, i.e., Control-X key. Additional special way for pressing keys is M-x, which stands for Meta-x, and this is obtained by pressing Alt key and 'x' in the same time, or by pressing Esc and 'x' one after another (not in the same time!). You can also notice in the 'splash screen' that by pressing `C-h t` ; i.e,., Control-h and then 't', you can start a tutorial, and this is also a way to learn emacs.

If you started emacs, you may want to know how to get out of it, and this is `C-x C-c` sequence.

There is another Unix editor called `vi` that can also be used to edit text files. If you are already familiar with `vi`, but do not know how to use emacs, you can choose to learn `emacs`, or to use `vi` instead.

The official editor of this course is `emacs`. Therefore, we will provide support for any questions with `emacs`. If you choose to use `vi`, then whenever there is a question in a lab that asks you to perform a task using `emacs`, perform the same task using `vi` instead.

If you do not know how to use `emacs` or `vi`, but know how to use other Unix editors such as `nano` and `pico`, do learn to use `emacs`, which is more suitable for serious programming than `pico` and `nano`.

Now, we can go back to editing the Java program, which we will call `HelloWorld.java` and which should print string "`Hello, world!`" on a line by itself. You can start by typing:

```
emacs HelloWorld.java
```

in the shell command line. If you forgot Java, your program could look as follows:

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

After finishing the program, you can exit emacs using `C-x C-c`. You will need to confirm that you want to save the file by pressing 'y' on the emacs question "Save file…".

Step 6: Compiling and running a Java program. Exit emacs and enter:

```
javac HelloWorld.java
```

to compile your code. If there are compile errors, edit you source code again and fix them. Once your code has been compiled successfully, enter:

```
java HelloWorld
```

to execute your Java Application. The output should be one line of text: '`Hello, world!`' In other words, this is the string that this Java program produces at the standard output. The program does not read any standard input and should not print anything to the standard error channel.

Step 7: Using emacs for search-and-replace. We will learn here how to replace a string with another string throughout a file using an emacs command. To do this, we make a copy of the file `HelloWorld.java`, and change it to a program that prints "Hi, world!" by following the instructions below:

7-a) Enter the command:
```
cp HelloWorld.java HiWorld.java
```
to create a copy of `HelloWorld.java` and name it `HiWorld.java`. The two arguments of this command specify the source and then the destination files.

7-b) Open `HiWorld.java` using `emacs`, using command:
```
emacs HiWorld.java
```

7-c) Press the key combination `Alt+x` and the cursor should appear in the status line (bottom line) following the string "`M-x`". If `Alt+x` does not work, you can try `Esc` followed by `x`.

7-d) Enter command "`replace-string`" and press `Enter`

7-e) Type the word "`Hello`", press `Enter`, type the word "`Hi`", and press `Enter` again. This should change all occurrences of "Hello" to "Hi".

7-f) Save, compile and run this new program to verify that we have successfully completed this task. To save the program you can use `C-x C-s`; then exit Emacs with `C-x C-c`; compile the file using `javac HiWorld.java` and run the command using `java HiWorld`.

You could achieve a similar effect by using the hot key for `Replace`, which is `M-%` i.e., Meta followed by the percentage character. The meta key is a key on a Unix keyboard. On PC, you can use the Alt key, or the Escape key, but followed by the percent sign instead of pressing it in the same time.

Step 8: Using SVN to submit files. Now, let us learn how to submit our work. We are going to submit a folder named `lab1` which contains two files: `HelloWorld.java` and `HiWorld.java`, without submitting any other files.

Follow the following sub-steps:

8-a) Your current working directory is supposed to be `lab1`. Change your current working directory to `csci2132` by entering:
```
cd ..
```
in the command line. The two dots in this command refers to the parent directory of the current working directory. This is, or will be, covered in class as well.
If you want to check to be sure that you are in the right directory, you can try the following commands:
```
pwd
```
will print your current directory. Based on this, you could check that you are in the directory `csci2132` just under your home directory. You could also do:
```
 cd ~/csci2132
 pwd
```
and you should see the same directory path being printed.

8-b) Rename the directory `lab1` to `lab1.bk`. Use the command:
```
mv lab1 lab1.bk
```
for this renaming task. This directory will keep a backup of our work, and we will later copy the Java files into a new directory. Use the `ls` command to verify that this task has been done successfully.

8-c) Now, in the `csci2132` directory, there is no subdirectory called `lab1`. Before recreating this directory, you will create a working copy of your SVN repository. Before this, you should make sure that you are

in the directory `csci2132` just under your home directory; in other words, you must be in the directory `~/csci2132`. (We will use character ~ (tilde) to denote your home directory, which is a standard as we will see later.)
First create a subdirectory named 'svn' using command:

```
mkdir svn
```

Go into this directory (by this we mean "change your current directory to this directory") using command:

```
cd svn
```

Now, you can create a working copy of your SVN repository by running the command:

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

where *CSID* is your CSID. You will need to enter your CSID password. The SVN command will ask you whether to "`Store password unencrypted (yes/no)?`", and you **should answer 'no'.** We will discuss later SVN to more details and what these commands mean. For now, you can understand this command as simply making a copy of your course SVN repository. You will soon add a directory and some files to it, and then you will copy them back to the SVN server.

Important Note: The above step will not work for you if you are not registered in the course and you will not be able to finish SVN submission. If you want access, or believe that you should have access, please send an email to the course instructor.

8-d) Go into the directory *CSID* (named after you CSID) using command:

```
cd CSID
```

Your current directory should be: `~/csci2132/svn/`*CSID*
Create now the directory '`lab1`' using the command:

```
mkdir lab1
```

Copy your Java files from `lab1.bk` to `lab1` using:

```
cp ../../lab1.bk/*.java lab1/
```

You can check that the files are in `lab1` using:

```
ls lab1
```

You should get an output as follows:

```
HelloWorld.java  HiWorld.java
```

8-e) Submit the content of `lab1` to the SVN repository using the following two commands:

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

Do not use any spaces in the above line within the string '`-mlab1submitted`'.
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 *CSID* is your CSID.

Step 9: The command `head` etc. In this step, you can learn how to display the first three lines of `HelloWorld.java` using a command named `head`. Use the Unix `man` command to get the Unix manual page for `head`, and find out how to perform the above task. You can use the command as follows:

```
man head
```

Check the description of this command, and read what the option `-n` does. This manual page may however be confusing for beginners. Note that you can get out of listing the man page by pressing key 'q' on the keyboard. To help you understand better, read the following web page as well, and compare it with the manual page to understand the format of the manual page: `http://en.wikipedia.org/wiki/Head_%28Unix%29`

By now, you have finished the required work of this lab. Although you will learn more about `emacs` in future labs, it is a good idea to learn more about it now. Practice using emacs yourself. You can read about it in the textbook or read the tutorial given earlier in the lab. You can create bigger files using emacs. You could write longer Java programs and run them on Unix, or write a long diary.

End of Lab 1