CMPS 12L
Introduction to Programming
Lab Assignment 1

The purpose of this assignment is threefold: get a basic introduction to the Unix operating system, to learn how to create and edit text files using either the Vi or Emacs text editors, and to learn to compile and run a java program.

Preparation
Before attempting this assignment, begin reading one of the Unix tutorials linked on the course website. You need not complete the tutorial, but find one that you like, and bookmark it for future reference. Vi and Emacs are two very popular text editors available on all Unix systems. Start reading one of the Vi tutorials, or the Emacs tutorial that are linked on the course website.

Unix
In order to submit any work in this class, you must first logon to your Unix Timeshare account (unix.ucsc.edu). For those running any version of Windows, you can do this by downloading a program called PuTTy. Go to http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html, download and then run the file putty.exe. Fill in `unix.ucsc.edu` as the Host Name, `22` as the port number, `SSH` as the connection type, then press open. Type your CruzID at the login prompt and give your Blue password. If you are running either Mac OSX or Linux, open a terminal window and type the command:

```
ssh cruzid@unix.ucsc.edu
```

where `cruzid` is your CruzID, then respond with your Blue password. If all this was done correctly you are now logged on to the Unix Timeshare. You will see a prompt that most likely looks like: `bash-4.1$`. This is the Unix command prompt which indicates that the command interpreter is waiting for you to type a command. In the examples that follow I will represent this prompt by the single character: `%`. If any of the above steps failed and you cannot logon, you'll need to attend a lab session and get help.

Type `ls` to list the contents of your home directory. Use the command `mkdir` to create a new directory called `cs12a` in which you will place all work for this class. Type `ls` again to see the new `cs12a` directory listed. Make `cs12a` your current working directory by typing `cd cs12a` at the command prompt.

```
% ls
% mkdir cs12a
% ls
% cd cs12a
```

Remember that `%` here represents the Unix command prompt and you do not type it. You can learn about any Unix command by typing `man` at the command prompt. Try:

```
% man mkdir
% man ls
% man cd
% man man
```

Man pages are notorious for being cryptic and even impenetrable, especially for beginners. Typically they assume a great deal of background knowledge. Nevertheless, you must get used to reading them since they are an invaluable resource. Use the man pages in conjunction with the tutorial to build up your vocabulary of Unix commands. Also try using Google to find Unix commands. For instance a Google

search on the phrase "unix copy" brings up a reference to the `cp` command. Research the following Unix commands, either through the tutorial, or man pages, or Google: `man`, `ls`, `pwd`, `cd`, `mkdir`, `more`, `less`, `cp`, `cat`, `rm`, `rmdir`, `mv`, `echo`, `date`, `time`, `alias`, `history`. You can also try just typing the command and see what happens. Create a subdirectory of `cs12a` called `lab1` and `cd` into it, then type `pwd` to confirm your location.

```
% mkdir lab1
% cd lab1
% pwd
```

The output of the last command should look something like

```
/afs/cats.ucsc.edu/users/f/cruzid/cs12a/lab1
```

where `cruzid` is your CruzID and the letter `f` may be different for you. This is the full path name of your current working directory. See http://docstore.mik.ua/orelly/unix2.1/lrnunix/ch03_01.htm for more on the Unix directory structure. It is highly recommended that you create separate subdirectories of `cs12a` for every lab and programming assignment in this class.

Editors

Using either the Vi or Emacs text editor create a file in your `lab1` directory called `HelloWorld.java` containing the following lines. This file can be found on the course website under `Examples/lab1`.

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

This is a java *source file*. Type `more HelloWorld.java` at the command prompt to view the contents of the file.

Java

In order to run the program we must first compile it. A *compiler* is a program that translates *source* code into *executable code*, which is what the computer understands. To compile the above program type

```
% javac HelloWorld.java
```

You should see the unix prompt (`%`) disappear for a few seconds, while it works, then reappear. List the contents of `lab1` again to see the new file `HelloWorld.class`. This is a java *executable file*. You can now run the program by typing

```
% java HelloWorld
```

This command should cause the words

```
Hello, world!
```

to be printed to the screen, followed by a new command prompt on the command line. We will have a lot more to say about the proper use and syntax of the Java programming language, but for now just note that what is printed to the screen is exactly what appears between quotes in the line

```
      System.out.println("Hello, world!");
```

in the source file `HelloWorld.java`. Also note that everything after `//` on a line constitutes a comment and is ignored by the compiler. Every program you write in this class must begin with a comment block of the following form.

```
// filename
// your Name
// your CruzID
// the assignment name (like lab1 or pa1)
// a very short description of what the program does
```

Open up your editor and change the comment block in HelloWorld.java to conform to the above format. Also change the body of the program so that it prints out your name:

```
Hello, my name is Foo Bar.
```

where `Foo` is your first name and `Bar` is your last name. Compile the new program and run it. If it does not compile, i.e. if you get error messages when you run `javac`, look for some stray character that you might have inserted into the file inadvertently, or perhaps a required character you failed to type.

Now create a new text file called `HelloWorld2.java` containing the lines

```
// HelloWorld2.java
// your Name
// your CruzID
// lab1
// prints greeting and some system information.
class HelloWorld2{
   public static void main( String[] args ){
      String os = System.getProperty("os.name");
      String osVer = System.getProperty("os.version");
      String jre = System.getProperty("java.runtime.name");
      String jreVer = System.getProperty("java.runtime.version");
      String jvm = System.getProperty("java.vm.name");
      String jvmVer = System.getProperty("java.vm.version");
      String home = System.getProperty("java.home");
      double freemem = Runtime.getRuntime().freeMemory();
      long time = System.currentTimeMillis();

      System.out.println("Hello, World!");
      System.out.println("Operating system: "+os+" "+osVer);
      System.out.println("Runtime environment: "+jre+" "+jreVer);
      System.out.println("Virtual machine: "+jvm+" "+jvmVer);
      System.out.println("Java home directory: "+home);
      System.out.println("Free memory: "+freemem+" bytes");
      System.out.printf("Time: %tc.%n", time);
   }
}
```

Compile and run this program by doing

```
% javac HelloWorld2.java
% java HelloWorld2
```

You will see that it prints something like

```
Hello, World!
Operating system: Linux 2.6.32-504.12.2.el6.x86_64
Runtime environment: OpenJDK Runtime Environment 1.7.0_75-mockbuild_2015_01_20_23_39-b00
Virtual machine: OpenJDK 64-Bit Server VM 24.75-b04
Java home directory: /usr/lib/jvm/java-1.7.0-openjdk-1.7.0.75.x86_64/jre
Free memory: 1.22431496E8 bytes
Time: Sun Mar 29 14:05:04 PDT 2015.
```

The exact output you get will depend on the date and time you run it, as well as the platform you are working on. You can see that the extra lines in this version of the program have the effect of collecting and printing certain platform specific information. The meaning of these lines may be discussed in class at some point.

Now edit this file once more so that the comment block contains your name and CruzID, and alter the greeting so that it prints

```
Hello, my name is Foo Bar
```

where `Foo Bar` is your name, as before. Recompile your program, wring out any typographical errors you might find, then test it.

What to turn in

Read the instructions on the website concerning the use of the submit command. Briefly, the syntax of the submit command is

```
% submit class_name assignment_name  file1 file2 file3 …
```

Here `class_name` will be `cmps012a-pt.f15` for all Lab and Programming assignments you turn in. The `assignment_name` in this case is `lab1`. Submit the two source files `HelloWorld.java` and `HelloWorld2.java`. Thus your submit command will be

```
% submit cmps012a-pt.f15 lab1 HelloWorld.java HelloWorld2.java
```

This command must be typed from within your lab1 directory where these source files reside, or you will get error messages. Start early and ask questions in the lab sessions or office hours if anything is unclear.