Dalhousie University
CSCI 2132 — Software Development
Winter 2017
Lab 3, January 26/27

In this lab, you will first learn how to use the UNIX `diff` command to compare two files and also get some experience of using pipes. You will then write and compile a hello world program in C using `emacs` and `gcc`.

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

1. First, perform the following steps to get started:

   (i) Login to server bluenose.cs.dal.ca via SSH from a CS Teaching Lab computer or from your own computer.

   (ii) Change your current working directory to the `csci2132` directory created in Lab 1.

   (iii) Create a subdirectory named lab3.

   (iv) Change your current working directory to this new directory.

   (v) Copy the two java files we created in Lab 1 from directory `lab1` to the current directory using one single command (Hint: wildcards).

   (vi) Make a copy of the file `HelloWorld.java` and name it `Hello`.

   (vii) Verify that in your current working directory, you have the following three text files: `HelloWorld.java`, `HiWorld.java` and `Hello`.

2. The utility `diff` is a very useful tool. It compares two files and prints a list of editing changes that will convert the first file into the second. Later this term, you will use this command to compare the output of your program with output files given in assignment specifications. This utility is also useful for source code management: If you and another person worked on the same source file, and both of you made changes to different parts of this file, you can use `diff` to help you merge your work.

   To run this command, simply enter `diff` followed by the pathnames of two files. To understand its output, read pages 93 and 94 on the UNIX text book. The textbook does not explain what "lineCount" means in the description of the "Deletions" on page 93. It means that after the deletion is made, these two files will be in synchronization, starting at line lineCount.

   To try this command, perform the following tasks:

   (i) Run `diff HelloWorld.java Hello`. There should be no text in stdout, as these two files are identical.

(ii) Run `diff HelloWorld.java HiWorld.java`. What does the output mean?

(iii) Use `emacs` to completely remove three consecutive lines in HelloWorld.java. Do not leave any empty lines (use Control-k four times).

(iv) Run `diff HelloWorld.java Hello` again. What does the output mean?

3. Next, let's get some practice on using pipes. Create a file named `halloffame2` containing the following 5 lines:

```
Albert Newton
Isaac Newton
Alan Turing
Albert Newton
Isaac Newton
```

Develop a single command line that prints the number of distinct last names in the above file on the screen. The output should be 2 in this example.

4. Now, let's start coding in C. Use `emacs` or `vi` to create the following C program which prints "hello, world" on a separate line. Use `hello.c` as its filename. Make sure to include the return statement in the main function.

```
#include <stdio.h>

int main(void) {
  printf("hello, world\n");
  return 0;
}
```

We will learn more about this program in the next week, and we will just get some hands-on experience in this lab. For now, it is sufficient to know that the statements in the `main` function will be executed when we run the program, the function `printf` in the main function will print "hello, world", and `#include <stdio.h>` includes some necessary stuff from the standard library so that we can call `printf` in this program.

5. Exit `emacs`, enter the following command to compile it: `gcc -o hello hello.c`

In this command, `gcc` is the compiler, which means "GNU Compiler Collection". Originally named "GNU C Compiler", this has become the standard compiler for many UNIX or UNIX-like operating systems. The "-o" option specifies the executable file name. Thus, after you enter the command, if the file has been successfully compiled, there will be an executable file named `hello` in the current working directory. The compiler automatically sets the permissions for execution, since this is an executable file. Use a UNIX command to check the permissions of `hello`. After this, enter `./hello` to run it.

Why do we have to give the pathname of this program to run it?

If there are error messages, compare your program to the source code given above, and make sure that they are exactly the same.

6. Try to compile your program without the "-o" option, i.e., do not specify `-o hello` in your command line. Are you able to find the executable file in your current working directory? It is not the file `hello` since we did not specify the name for the executable file. After you locate this file, execute it.

7. When testing our programs, we often modify them. If each time we modify a C program, we exit `emacs` and compile it, it will not be very productive. One solution is to open two terminal windows on the same computer, change the current working directories of both to the same directory, and use one for editing and the other for compiling / running the program.

   Alternatively, we can learn two more hot keys of `emacs` without opening another terminal. Let's practice this.

   We first learn how to compile our program without exiting `emacs`. Following the steps below:

   (i) Open `hello.c` using `emacs`.

   (ii) Press Esc-x (if you have a UNIX keyboard, you are supposed to use Meta instead of the escape key), type "compile", and then press enter.

   (iii) You will be asked to enter the compile command. Use backspace key to delete the default command entirely, and then enter the `gcc` command given in Question 3.

   (iv) This will split the current `emacs` window into two, and the bottom window will show the message generated by the compiler.

   (v) To go back to the single window mode so that we can see more lines of source code, enter Control-x, then 1 (this is the digit 1).

   (vi) You can invoke any shell command without exiting `emacs`, and this includes running the compiled code. To do this, enter `Esc` key followed by the `!` key (again, if you have a UNIX keyboard, you are supposed to use Meta instead of the escape key). Then you can enter any command. You can now enter `./hello` and you will see the result in `emacs`.

8. What does the statement `return 0;` do in the main function? This is about exit code in UNIX. Each UNIX command will return an exit code (a nonnegative integer) to the shell after its execution to indicate whether it has been executed successfully. Most UNIX utilities return 0 to indicate successful execution, and 1 for failure. This exit code is then stored in the `$?` built-in variable of the shell, which can be displayed using `echo $?`.

Run our program again, and use the above command to check the exit code. Then, change the value that the return statement returns (say, change it to 10), compile, run the program, and check again.