COSC 220 - Computer Science II
Lab 1

Dr. Joe Anderson

Due: September 2

1 Objectives

In this lab you will focus on the following objectives:

1. Develop familiarity with the GNU/Linux operating system
2. Use the GCC compiler to compile some simple C++ programs
3. Review C++ arrays and pointers

2 Tasks

1. Open a terminal window (the "Terminal" application if your Desktop Environment is Gnome, or "Konsole" if your desktop environment is KDE)
2. Look at the command prompt. Your starting location should be your home directory, represented by simply the symbol `~`.
3. Use the `pwd` command ("Print Working Directory") to print the full path of your current location.
4. Use the `ls` and `ls -al` commands ("List") to list the files in your home directory. Notice the different outputs from the two commands.
5. Use the `mkdir` command to create a folder in your home directory – this will be used for future labs so you may want to call it "Development" or "COSC-220". Don't use spaces in file or folder names, for now. For example:

   ```
   $ mkdir COSC-220
   ```

6. Use the `ls` command to see that your new folder is in your home folder.
7. Use the `cd` command ("Change directory") to move into your new folder, e.g.

   ```
   $ cd COSC-220
   ```

8. Create a new folder called "Lab-1" inside your folder for this class. This folder will be zipped and turned in later.
9. Use the command `touch` to create a new file called "`arrays.cpp`". For example:

   ```
   $ touch arrays.cpp
   ```

10. Use `ls` to verify that your file is there.

11. The `cat` command can print the contents of your file (`cat arrays.cpp`) – but it should be empty at this point

12. Open the file in a text editor. Some options you should try:

    (a) `nano` (text, simplest) – note that `^X` means to press `ctrl+x`

    (b) `vim` (text, steep learning curve)

    (c) `emacs` (text, steep learning curve)

    (d) Kate (graphical)

    (e) gedit (graphical)

13. In your `arrays.cpp` file:

    (a) Create a main function that asks the user for several integers and stores them in an array.

    (b) Write a function called `mean` that takes an integer array argument, along with its length, and returns a `double` representing the mean of the passed array.

    (c) Write a second function called `mean2` which is the same as `mean`, but does not use the `[]` operator anywhere.

        i. Recall that with arrays, accessing a single element, e.g. with `arr[i]`, is the same as dereferencing a pointer to that same address: `*(arr + i)`.

14. **To compile and run your program:**

    (a) Open a (second, to make things easier) terminal and navigate to the directory containing your source file.

    (b) Use the command `g++ arrays.cpp` to compile your code. This will output a binary file called `a.out`. If there are compilation errors, they will be shown here.

        i. To change the name of the output executable, add the option: `-o MyExecutable`

        ii. You can change `MyExecutable` to whatever name you like.

    (c) Run your program as follows:

        ```
        $ ./a.out
        ```

    (d) At this point, you should be able to interact with your program as you would expect.

    (e) To submit your lab, see the instructions below.

15. Finally (do not turn this in):

    (a) Try using the `man` command to see the documentation for the commands you ran above. E.g.

        ```
        $ man pwd
        $ man cd
        $ man mkdir
        $ man g++
        $ man man
        ```

    (b) If you want to try using `vim`, try using the `vimtutor` command first.

    (c) Try using one of the other text editors to write another small `c++` program.

    (d) See the course webpage for more information and links to programming resources to make development on Linux easier!

3 Submission

The following are due by the beginning of lecture on the due date.

Upload your project files (for this project only!) to the course canvas system in a single `.zip` file.

Some graphical interfaces allow you to right click a folder and select "compress" or "archive". Or, to zip from the command line:

1. To zip a single folder into a single archive called "myZip.zip":

```
zip -r myZip.zip folderName
```

2. To zip multiple files into a zip file called "myZip.zip":

```
zip myZip.zip file1.cpp file2.h file3 file4.cpp
```

Turn in (stapled) printouts of your source code, properly commented and formatted with your name, course number, and complete description of the code. To print, you will want to use either the `lpr` command (see in-class instruction) or the graphical interface.

Also turn in printouts reflecting several different runs of your program (you can copy/past from the terminal output window). Be sure to test different situations, show how the program handles erroneous input and different edge cases.

4 Bonus

(Up to 10 pts) Allocate and store your array using *dynamic* memory, i.e. allocated with `new` and allow the user to, in principle, enter an unlimited amount of data (depending of course on the amount of memory in the machine!). Note that you'll want to allocate a fixed-size to start, then dynamically increase the size of the array as needed; be sure to avoid memory leaks during this procedure!