CMSC 246 – Lab 0

Jan 19, 2016
Updated Jan 21, 2016

Objective:

Basic familiarity with the Linux system. Compile and execute a short C++ program to ensure that setup is working properly. Compile and execute a program that uses an external library and object file.

Starting up:

1. Log into a Linux workstation if you have not done so already. Your username is the same as your brynmawr email account name (without the @brynmawr.edu portion). This is also known as your UNIX account name on this campus. Your password should be the same one you use to check your email.

2. Once you have logged in, be sure to let the desktop start up. At the bottom (or on the top) of the desktop you will see a toolbar of applications that you can use to access the parts of the system. Start a **terminal** window. Be sure to take notes if you need to.

3. Try out the UNIX commands on the UNIX cheat sheet. Use the `man` command to find out more about them.

4. Try out the either emacs or vi for editing.

File editing:

You are strongly encouraged to get to know and use **Emacs**. It is a very powerful text editor, it has a lot of useful features and it's fairly easy to use. The **Emacs** tutorial sheet is a good place to start. You can also use the menu on the top of the window and the built-in help. You can find some good guides on the web: http://www.gnu.org/software/emacs/tour/ . See the emacs reference card for more info.

Alternatively, vi/vim is another good text editor for coding.

Creating, versioning, and archiving. Compiling and running c++ programs.:

1. Create a file in your home directory (~/ *or* /home/username/) called .hgrc and add the following contents to it. Also put your Name and email address in the same format as Jane Doe after username =
   Note: lines that start with # are comments

```
[ui]
# name and email, e.g.
# username = Jane Doe <jdoe@example.com>
username =

[extensions]
# uncomment these lines to enable some popular extensions
# (see "hg help extensions" for more info)
#
# pager =
# progress =
# color =
```

   Note: All monospaced text under a number should be typed at the command prompt.
2. Create a `CS246` directory in your home directory (the one you logged into). Use the `mkdir` command to do this. Move into the newly created directory (the `cd` command).

3. Create another directory named `lab0` for this week's work.

4. Move into lab0 (using cd)

5. Initialize a mercurial repository, `hg init` . You will only need to do this once.

6. Make a file called emptyLib.cpp

```
touch emptyLib.cpp
```

7. Compile the empty library

```
g++ -o emptyLib.o -c emptyLib.cpp
```

8. This will make an object file called emptyLib.o
9. Make a file called emptyApp.cpp
10. Try to compile the app with the following command:

```
g++ -o emptyApp emptyApp.cpp
```

11. Note the failure.

12. Edit emptyApp.cpp with the following command:

    ```
    echo "int main() { return 0; }" >> emptyApp.cpp
    ```

13. Try the compile command from step 10 again:

    ```
    g++ -o emptyApp emptyApp.cpp
    ```

14. Run emptyApp using the following command:

    ./emptyApp

15. Note that there is no output, since the command runs an empty program.

16. Enter the program listed as Program listing 1 below (on page 5) into a file called HelloGraphics.cpp (Make sure you update the comments appropriately including changing the course number)
    Use Emacs (or vi) to do this. Notice how it helps you with the indentation and pairing of parentheses. Don't forget to save the file! Hint: type '`emacs HelloGraphics.cpp &`' to start up Emacs. You won't need to quit your editor in order to compile your program, but you will need to save the file.

17. copy EasyWindow.h, EasyWindow.cpp, EasyColors.h from /rd/cs246s2016/shared/lab0/ to your `lab0` directory using the following command

    ```
    cp /rd/cs246s2016/shared/lab0/*  ./
    ```

18. Compile EasyWindow.cpp using the command

    ```
    g++ -o EasyWindow.o -c EasyWindow.cpp
    ```

    The general rule is:

    ```
    g++ [ -o outputfilename.o ] -c sourcefile
    ```

19. Compile your program using the command

    ```
    g++ -o HelloGraphics HelloGraphics.cpp EasyWindow.o -lSDL2
    -lSDL2_gfx
    ```

    The general rule is:

    ```
    g++ [ -o outputfilename ] sourcefile [ objectfile ] [ -l Library
    Name ]
    ```

    If you receive errors during the compilation phase, re-edit the source code file and attempt to correct them.

20. Once the file successfully compiles, execute it using the command:

    ```
    ./HelloGraphics
    ```

21. Commit your changes to the mercurial versioning system. Type:

    ```
    hg add
    hg commit -m "log message for lab 0"
    ```

22. Go up a directory, and submit your lab with the following command:
    (Note: the ` character is typed with the same key as ~)

    ```
    X=`umask`; echo $X; umask 0007; hg clone --pull lab0
    /rd/cs246s2016/${USER}/lab0 ; umask $X;
    ```

23. Once you have cloned the repository, you can push future commits with hg push like so:
    (Note: USER is an environment variable, and to access any variable, you can use the ${[varname]} syntax.

    ```
    hg push /rd/cs246s2016/${USER}/lab0
    ```

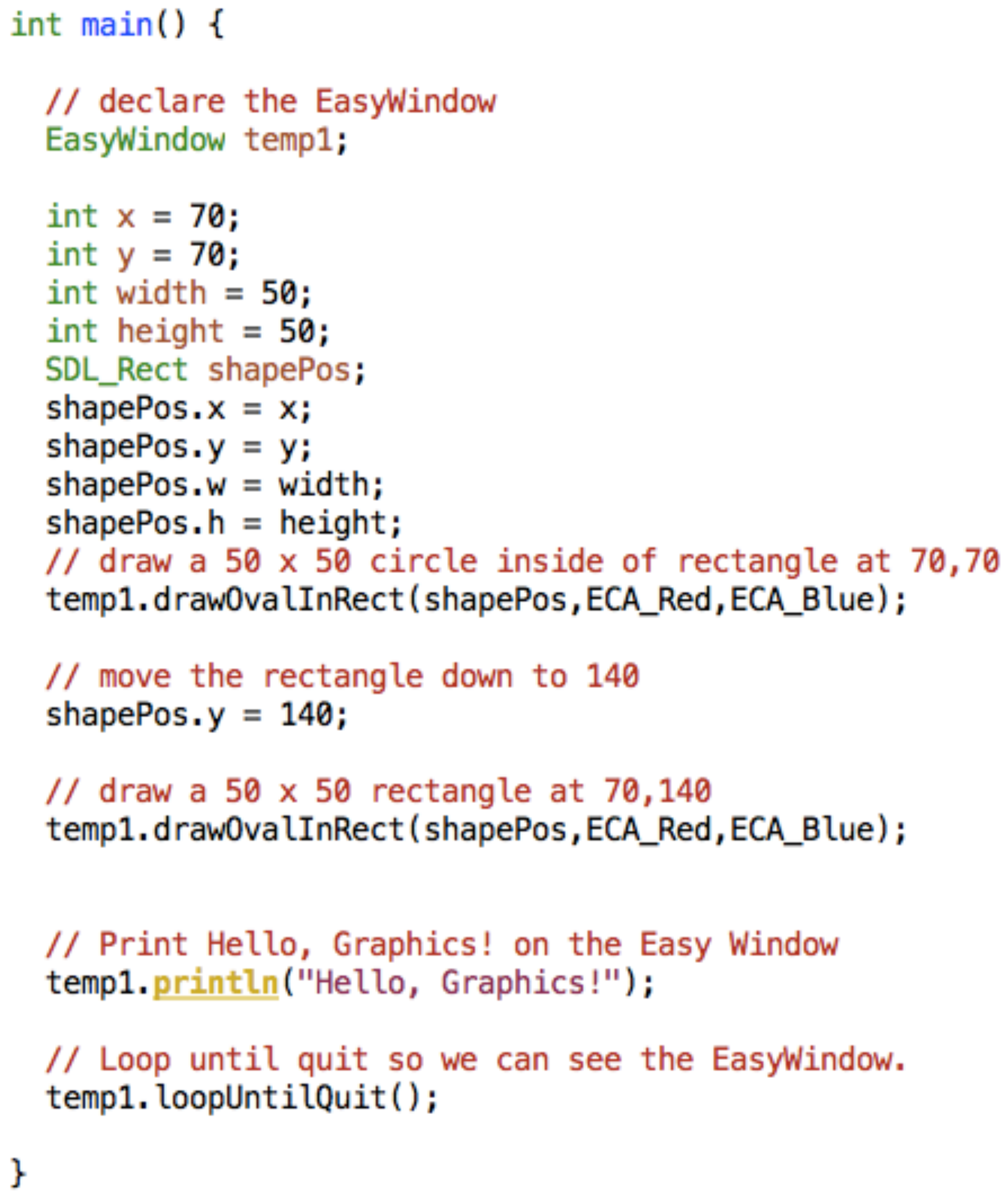
24. With your partner, add your graphical design from the exercise during the lecture. (compile, run, commit, and push to `/rd/cs246s2016/${USER}/lab0`). See program listing 2 (page 6) for an example of the graphics calls. Available color variable names are in `EasyColors.h`

25. The second part should be submitted by the end of day on Sunday, Jan 24, 2016.

Program Listing 1:

```
/*
 * HelloGraphics.cpp
 * Programmer(s): ???,???
 * Date Submitted: ??/??/2015
 * Instructor: Dr. Cooper
 * Course: CSC215
 * Assignment: Lab ??
 */


#include "EasyWindow.h"
#include "EasyColors.h"
#include <unistd.h>



int main() {

  // declare the EasyWindow
  EasyWindow temp1;

  // Print Hello, Graphics! on the Easy Window
  temp1.println("Hello, Graphics!");

  // Loop until quit so we can see the EasyWindow.
  temp1.loopUntilQuit();

}
```

Program Listing 2



```
int main() {

  // declare the EasyWindow
  EasyWindow temp1;

  int x = 70;
  int y = 70;
  int width = 50;
  int height = 50;
  SDL_Rect shapePos;
  shapePos.x = x;
  shapePos.y = y;
  shapePos.w = width;
  shapePos.h = height;
  // draw a 50 x 50 circle inside of rectangle at 70,70
  temp1.drawOvalInRect(shapePos,ECA_Red,ECA_Blue);

  // move the rectangle down to 140
  shapePos.y = 140;

  // draw a 50 x 50 rectangle at 70,140
  temp1.drawOvalInRect(shapePos,ECA_Red,ECA_Blue);


  // Print Hello, Graphics! on the Easy Window
  temp1.println("Hello, Graphics!");

  // Loop until quit so we can see the EasyWindow.
  temp1.loopUntilQuit();

}
```