Computer Science 381
Programming Unix in C

The College of Saint Rose
Fall 2013

Lab 2: C Basics

Due: 10:25 AM, Wednesday, September 11, 2013

In this week's lab you will learn how to use Emacs in a better way, then focus on some of the basics of C programming.

Before you start, create a directory on a Unix or Mac system to contain your work for this lab, and open a file `lab2.txt` where you will place your answers for this week's lab questions.

X11 and Emacs

Now that you have seen the basics of using Emacs and Unix, we'll try to make it a more pleasant and somewhat user-friendly experience. A huge part of this is being able to use the mouse to reposition the cursor, select text, *etc.*.

We will do this using an old graphical user interface system called the *X Window System*, or *X11*. The "11" here is the major version number of thr system, though I can't imagine there will ever be an "X12", since it's been "X11" for the 20+ years I've been using it. It is the default (and generally, only) GUI for Unix-like systems, and can also be used on the Mac or Windows, with the proper software installed.

To try this out on a lab Mac, open a Terminal, and type any X11-aware command. I recommend "`xeyes`". If all is set up correctly, you will see "XQuartz" launch and a pair of eyes that follow your mouse will appear on your screen.

In many ways, this is similar to the windowing systems you'll find on other operating systems like Mac OS X and Windows. But the great value-added here, and a major reason that X11 has withstood the test of time is the ability to run a program on one computer and display and interact with its windows on another computer. Let's try this with mogul.

In your terminal on the Mac, connect to mogul with the command:

```
ssh -Y mogul.strose.edu -l yourusernameonmogul
```

The `-Y` flag is what tells ssh that you would like to set up an X11 tunnel to display programs running on the remote computer on your local screen.

To make sure this works, again run `xeyes`. If the eyes appear again, you're all set. This time, the `xeyes` program is running on mogul, but the windows it creates are displayed on your local machine.

Now, to the point for today, you can also run Emacs with its X11 GUI from mogul. You don't need to do anything special – if you run `emacs` there and your connection was set up with X11

forwarding, it will launch Emacs in a new X11 window, in which you can click around with the mouse, and use the menus at the top (though I strongly recommend learning the keystrokes we used last week instead).

The last tip for today is about running commands in the "background". In a Terminal on the Mac or on mogul, launch `xeyes` again. Notice that while `xeyes` is running, you can't type any commands in that terminal. In fact, you don't even get a prompt back. If you kill `xeyes` (you can do this by typing `Ctrl-C` in the terminal), you will then get your prompt back.

However, when a program is launched using X11, you normally interact with it through the GUI windows it brings up rather than through the command-line terminal. So we often launch X11 programs in the background. Run `xeyes` again, but this time, add the & character after the command. The eyes should appear again, but this time, you will also get your prompt back.

You can do the same when you launch Emacs on mogul. This means you can keep Emacs running in the background, and interact with it through its GUI, and use the command line to run your compile commands and to run your programs.

C Basics

For the rest of this lab, you may work on any C-capable Unix-like system (such as a lab Mac or mogul).

If you have not already done so, finish reading Chapter 1 of K&R.

Lab Question 1:

Consider any C program that uses the `printf` function. What happens if you leave out the `#include <stdio.h>` line? Explain briefly. (2 points)

Lab Question 2:

Suppose one of the programs from the running example about conversion from Fahrenheit to Celsius is to be modified to print conversions for every degree Fahrenheit from -100 to 1000 and you wish to print 3 digits after the decimal point for each Celsius temperature. What `printf` function call could be used to print this so that the printed temperatures are aligned nicely? (2 points)

Practice Program:

Write a program `charcount.c`, similar to that in Exercise 1-14 on p. 24 of K&R. Your program should deal only with numeric and alphabetic characters, and should treat upper or lower case characters as equivalent. You also need not print a histogram, you can just print a table of the results. Like the example program on p. 22, your program should read characters from the standard input and print the results to standard output. (10 points)

Run your practice program on the input file on mogul in `/home/cs381/labs/c-basics/whosonfirst.t`
and capture your output in `whocounts.txt` with a command such as

```
./charcount < whosonfirst.txt > whocounts.txt
```

Output Capture:
`whocounts.txt` for 2 point(s)

Practice Program:
Write the program in Exercise 1-19 on p. 31 of K&R. Call your program `revlines.c`. (10 points)

Also run this program on the `whosonfirst.txt` file, this time saving the output in `whorev.txt`.

Output Capture:
`whorev.txt` for 2 point(s)

Lab Question 3:
Briefly describe the major differences between arrays in C, as presented in Chapter 1 of K&R, and arrays in Java. (2 points)

Programming Assignment

Write the program in Exercise 1-20 on p. 34 of K&R. Call your program `detab.c`. Use a defined constant for the value n, the number of columns between tab stops. Include This program is worth 20 points, broken down as shown at the end of this document.

Reference solutions to all programs are available on mogul in `/home/cs381/labs/c-basics`.

Submission

Please submit all required files as email attachments to *terescoj@strose.edu* by 10:25 AM, Wednesday, September 11, 2013. Be sure to check that you have used the correct file names and that your submission matches all of the submission guidelines listed on the course home page.

Grading

Grading Breakdown	
Lab questions and output captures	10 points
Practice program `charcount.c` correctness	10 points
Practice program `revlines.c` correctness	10 points
`detab.c` correctness	10 points
`detab.c` design	3 point
`detab.c` documentation	3 points
`detab.c` style	3 point
`detab.c` efficiency	1 point