CSCI 1120 (Low-Level Computing), Fall 2014

Homework 1

Credit: 10 points.

1 Reading

Be sure you have read the assigned readings for classes through 9/10.

2 Programming Problems

Do the following programming problems. You will end up with at least one file per problem (text for the first problem, source code for the second). Submit these files by sending mail to `bmassing@cs.trinity.edu`, with each file as an attachment. Please use a subject line that mentions the course number and the assignment (e.g., "csci 1120 homework 1"). You can do this assignment on any system that provides the needed functionality, but I will test the program on one of the department's Linux machines, so you should probably make sure it works in that environment before turning it in.

1. (5 points) (Not really a programming problem, but one that requires you to use a computer.) For this problem your mission is to learn a little more about traditional UNIX text editors `vi` and/or `emacs`. Do one or both of the following (full credit for doing one, extra credit if you do both):

   - Do something to improve your ability to use `vi` (or, more properly, `vim`, since that's what we have installed). Options include:
     - Start the interactive tutorial by opening a terminal window and typing `vimtutor`. Work through at least the first lesson, more if you have time.
     - Think about your past use of `vi` and identify something you find particularly annoying (e.g., not knowing how to cut and paste). Then try to find a way to reduce the annoyance. You may find something helpful in the tutorial, or in the online help (which you start from within `vi` by typing `:help` and pressing the Enter key), or you may prefer to use your favorite search engine.

     Use what you've learned to write, in a text file, a paragraph or two reporting on what you learned and what you still wish you knew about this editor.
   - Learn something about `emacs`. If you've never used it, start it by typing `emacs -nw` in a terminal window. This should give you a page of instructions. Press control-h and then t to start an interactive tutorial. Work through as much of this tutorial as you need to in order to create and save a text file. (Starting the program by just typing `emacs` starts a graphical version of the program, which you may prefer for use in our labs, but which isn't as useful if you're working remotely.) If you already know something about `emacs`, either work through some of the tutorial, or do the second option for `vi` above (identify an annoyance and try to figure out a way to reduce it), but for `emacs`. Use what you've learned to write, in a text file, a paragraph or two reporting on what you learned and what you still wish you knew about this editor.

Turn in the resulting text file(s).

2. (5 points) Write a C program to convert seconds into years, days, hours, minutes, and seconds. Your program should prompt the user for a number of seconds, get the number entered, and print the equivalent number of years, days, etc. (e.g., 100 seconds is 0 years, 0 days, 0 hours, 1 minute, and 40 seconds). Assume 365 days in a year (not quite right but can make the calculations simpler). We have not talked in detail about how to do conditional execution in C, but the example of getting user input on the sample programs page `simple-io.c`[1] should be enough to let you do simple error checking: If what is entered is not an integer, or is less than zero, print an error message and stop.

   Hint: Probably the best way to do the required calculations is with integer-division (`/`) and remainder (`%`) operators.

[1] `http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2014fall/SamplePrograms/Programs/simple-io.c`