Basic C Programming

CS 350: Computer Organization & Assembler Language Programming

Lab 1, due Wed Jan 20 (2400 hrs)[1]

[1/14: Sorry, alpha isn't up yet, so we'll have to postpone this lab]

A. Why?

- You'll be writing your programs for CS 350 in C and you'll definitely be writing programs for CS 351 in C.
- One of our later topics will be seeing how high-level programs in C are implemented as lower-level programs in machine code (the instructions that the hardware understands).

B. Outcomes

After this lab, you should be able to:

- Log into the `alpha.cs.iit.edu` machine and compile and run a simple C program.

C. Discussion

- C is a "lower-level" language than Java: its constructs more easily map to the data and operations found on typical hardware.
- You should have accounts on `alpha.cs.iit.edu`; if not, let me know (Piazza would be useful here).
- As part of the zip file that makes up this lab, you should find `Lab1.c`.

D. Logging Into alpha and Compiling

- The `alpha` machine runs Linux; if you don't already know how to use Linux, it'll be good for you to learn how to. The `linux-account.pdf` file that's part of this

[1] You get an automatic one-day extension if you attend lab the previous week; if you didn't already know that, you should read the syllabus.

lab will show you the basics of Linux; the version attached refers to an old computer; substitute `alpha.cs.iit.edu` everywhere you see `dijkstra.cs.iit.edu`. (Thanks to Dr. Beckman for sharing his handout!)

- If you already had an account on `alpha`, just continue using it. If you didn't already have an account on `alpha`, you should receive an email from it/Dr. Beckman telling you about it.
- If you need help, your Lab TAs can show you how to log into `alpha` from the using a secure shell session (ssh) via `PuTTY` (on Windows), `Terminal` or `iTerm2` (Mac OS X) or `ssh` (Linux).
- To transfer files to `alpha`, you'll probably want an SFTP (secure file transfer protocol) program; `FileZilla` seems popular.
- For this lab, practice logging into the `alpha` machine and compiling and running the `Lab1.c` program. Once you have a copy of the program in your current directory, the Linux command to compile the program is

  ```
  gcc -Wall -std=c99 -lm Lab1.c
  ```

  "`gcc`" means "GNU [pronounced Guh-Noo] C compiler," the standard compiler for Linux environments[2]. The option `-Wall` says to print all error messages; `-std=c99` says to use the ISO C99 standard; the `-lm` says to include the math library (so you can use `sqrt`). Depending on your setup, you may not need the `-lm`; if you get a complaint about a missing `sqrt` routine when you compile your program, then you need the `-lm`. It may also be possible to put the `-lm` after the *filename*`.c`: `gcc -Wall -std=c99 -lm Lab1.c -lm`
- If the compile succeeds, it produces an executable file named `a.out`. To run your program, execute that file with the command `./a.out`
- **Optional:** If at some point during the semester, you get tired of typing in all the `gcc` compile options, use a text editor to edit (or create) your `~/.bashrc` file, which contains initializations used by the `bash` "shell" program that you type your Linux commands into. Add the line

[2] "GNU" stands for "GNU's Not Unix", a reference to GNU being different from the versions of Unix that existed when the GNU project was started


```
alias gcc="gcc -Wall -std=c99 -lm"
```

to the `~/.bashrc` file. Close the file and log out and log back in. Now you can just type `gcc` *filename*`.c` when you want to compile, and the bash shell will substitute the `gcc` with options for the `gcc` in your typed-in input.

E. The Sample Program

- Read through the sample program `Lab1.c`. You'll find much of C is similar to Java, but there are some fairly large differences too. [Ignore the problems for now.]
- The program contains a number of constructs, including:
  - Declarations of variables of basic types (`int`, `double`, `char`) and arrays of basic types of values.
  - The `printf` (print formatted) function for printing out values to the screen. Some basic formats (`%d`, `%f`, `%c`, and `%s`) are used.
  - String constants and strings stored as character arrays.
  - The `scanf` (scan formatted) function for reading values from the keyboard.
  - The `sscanf` (string scan formatted) function for reading values from a string.
  - The type `long int` (long integer), which is like regular `int` but can store larger values.
  - Hunt down some reference material on basic C programming as necessary to understand how the program works.

F. Problems [50 points]

- There are problem descriptions in the comments of the `Lab1.c` program. Write out answers to the problems in a separate text or pdf document and submit it using Blackboard: Find Lab 1 under Assignments and press the link for uploading your solution. You don't need to include program runs with your answers.
- In any case, don't bother including an object file or executable file. (See http://cs.iit.edu/~cs350 → Syllabus.)

