CSC161 2011S *Imperative Problem Solving*

Assignment 2: C Basics

Assigned: Friday, 28 January 2011
Due: 11:00 p.m., Wednesday, 2 February 2011

Summary: In this assignment, you will begin exploring the C programming language through a few simple examples.

Purposes: To get you started thinking about C.

Expected Time: Two to three hours.

Collaboration: I would encourage you to work in groups of two or three on this assignment. However, you may also work alone or work in groups of up to size four. Each group should turn in only one copy of the assignment. You may certainly work with other students as you review the course materials and answer the questions. You may discuss the assignment with anyone you wish. You may obtain help from anyone you wish, but you should clearly document that help.

Note: Because K&R is an important book, answers to all of these exercises are readily available online. My academic honesty guidelines do indicate that you can copy any answers you find online. However, you will find that you learn much more if you do the work on your own.

Submitting: Submit a tarball via Pioneerweb. See further details below.

Warning: So that this assignment is a learning experience for everyone, I may spend class time publicly critiquing your work.

Preliminaries

Executable Shell Programs

We've experimented with writting shell commands on the command line. However, you might want to save a command for later use. Typically, we put the command into a file for processing.

You can create files with GEdit. (Type `gedit &` on the command line.)

Your file should have the following form:

```
#!/bin/bash

instruction-1
instruction-2
...
```

You need to type `chmod 755 filename` to make it executable.

You run it with `./filename`.

For example, if I want to write a script that gets the first names of faculty in the three divisions, I might put the following in a file called `facnames`.

```
#!/bin/bash

echo "SCIENCE"
cut -d' ' -f2 /home/rebelsky/share/linux/sciencefac.txt | sort -u
echo ""
echo "SOCIAL STUDIES"
cut -d' ' -f2 /home/rebelsky/share/linux/socialfac.txt | sort -u
echo ""
echo "HUMANITIES"
cut -d' ' -f2 /home/rebelsky/share/linux/humfac.txt | sort -u
echo ""
```

I would then run that with

```
./facnames
```

Recording Sessions with `script`

At times, you'll want to provide a record of what you've done. For example, you may want to run a C program on different inputs and show me the results of those commands. (In effect, you want something like the interactions pane from MediaScheme.) In Linux parlance, these are "scripts". It is fairly easy to produce scripts using the Linux `script` command.

In the terminal window, type:

```
script filename
```

This will begin a script session, the output of which will be written to *filename*. Be careful to use different filenames (or directories) for each script you want to submit.

Now continue to type commands (ie, run your program and enter test data for each of your test cases) in the same terminal window. Note that it will make your script easier to read if you enter a few Enter/Return characters between each test.

When you have completed your test runs, type `exit` to finish the script session. At this point all of your interaction during the session, including both input and output, will be saved into the script file.

Unfortunately, a few messy characters will also be entered into your script file for any special function keys, such as arrows, delete, backspace, and home, that you use. It would be nice if you could avoid these keys while producing your script, but I understand that it is hard to avoid them completely.

Please produce one test script for each problem on the assignment. You can look at the test script using either cat or less, but you must not edit your test script in any way.

Assignment

Part One: Fun with Grades

Professor Ric (that's, Gene Ric, in case you were wondering) stores grades for classes in text files in which each line records one grade, using the following format:

uid:*assignment*:*grade*:*notes*

For example,

```
student1:hw1:23:
student2:hw1:9:significant problems with syntax
student1:hw2:30:lots of extra credit
student1:hw3:5:surprisingly bad
student1:reading5:3:
student3:hw1:20:late, but nice
student4:hw1:27:late, but exceptionally nice
```

a. Write instructions for finding the highest grade on homework 1 (named `hw1`). Store those in an executable called `besthw1`. You will probably need a pipeline that includes `grep`, `sort`, and other commands.

b. Write instructions for finding the rank of `student1` on `hw1`. (In the example above, student 1 is ranked second.) Once again, you will find `grep` and `sort` helpful. You will also need to find other commands that will help you identify a line number.

For this problem, you can assume that no two students have the same grade.

Part Two: Problems from K&R

Do exercises 1-3, 1-4, 1-7, 1-8, and 1-13. In your answer to 1-13, you may assume that no word has more than ten characters.

Test each of your answers to show that the code works to your satisfaction.

Do your best to follow the GNU Coding Standards (which are not the standards that K&R follow) in formatting your code.

Important Evaluation Criteria

I will primarily evaluate your programs on their correctness. I will also look at the quality of the tests that you write.

Submitting Your Homework

Create a directory for this assignment of the form `hw02.`*`names`*. Within that directory, create subdirectories for each part of the assignment (e.g., highest, ranking, kr1-3). Each subdirectory should contain:

- The code that you have written for that part of the assignment.
- A typescript or typescripts in which you demonstrate the utility of the code you have written.
- (Optional) A README file with any additional info you want to provide.

Make a gzipped tarball of the assignment directory. (If you don't know how to do that, let me know.) Upload it via the Assignment 2 link on Pioneerweb. If possible, eliminate any executables before you build that tarball.

Citations

The instructions for creating a typescript are taken from similar instructions by Marge Coahran.
