CS 4414 Spring 2011

Programming Assignment 1

Due February 13, 2011 12:05 am

Getting Started

All exercises are to be completed using C and `make` on the CSLAB Linux interactive systems (labunix01-labunix03.cs.virginia.edu). You may develop your software on another computer, but the source code for the version you turn in must reside on the lab machines, compile from a makefile, and execute correctly there.

You should already have received your CSLAB user ID (same as your UVA computing ID) and initial password. Please be sure to login to one of the lab machines and change your password as soon as possible; you can use any of labunix01, labunix02, or labunix03 - your account and home directory are available on all 3 systems. You can login to the Linux machines from your own computer or a lab workstation using Secure Shell (SSH).

To complete this assignment, you should:

1. Be familiar with the C language (or C++ and the differences between it and C)
2. Have a basic understanding of how to use Linux (including reading `man` pages)
3. Know how to run Linux commands using a shell (`bash` is the default login shell on the lab machines)
4. Know how to use a text editor (Vi/Vim, Emacs, Nano, etc.)
5. Understand the basics of `make`
6. Know how to use the `getfacl` and `setfacl` commands

Tutorials for C, make, and Access Control Lists (ACLs) are available on the Collab class site under Resources. For this assignment you can skip the Subversion portion of the Access Control Lists and Subversion (ACL and SVN) tutorial.

C Programming Exercises

In exercises 2-6, use the `list_t` and `list_item_t` structures as defined in the C tutorial. The information given in 2-6 should be sufficient to determine functionality. In all cases, gracefully handle user errors with a displayed message followed by termination.

1. Write a program that takes two command line arguments. The first will be a file name. The second should be one of "echo", "sort", "tail", and "tail-remove". Open the specified file, read the input, one line at a time, until `EOF` is read. If the second parameter is "echo", display the input, one word (contiguous string of non-whitespace characters) per line. Do not display white space.

2. Implement a constructor for the list class. Its prototype should be:

```
void list_init(list_t *l,
      int (*compare)(const void *key, const void *with),
      void (*datumdelete)(void *datum))
```

3. Implement `void list_visit_items(list_t *l, void (*visitor)(void *v))`, which takes a pointer to a function, `visitor`, and calls it on each member of `l` in order from `l->head` to `l->tail`. You will need to use this to print your list.

4. Implement `void list_insert_tail(list_t *l, void *v)`. Extend your program from Exercise 1 to insert the input, line by line, when the second parameter is "tail", and print the contents of the list, one member per line before exiting.

5. Implement `void list_insert_sorted(list_t *l, void *v)`. Elements inserted in the list should be in ascending order as determined by the comparison function. Further extend your program from Exercise 1 to insert the input in sorted order, line by line, when the second parameter is "sort", and print the contents of the list, one member per line before exiting.

6. Implement `void list_remove_head(list_t *l)`. Further extend your program from Exercise 1 to insert the input using `list_insert_tail()`, line by line, when the second parameter is "tail-remove". Then remove 3 items from the head of the list and print the contents of the list, one member per line; repeat until the list is empty. Be sure to "print" the empty list. Display a row of hyphens between each iteration.

Other than error messages, do not print anything not specified above. You may print debugging messages while you are developing and testing your code, but debugging should be removed or disabled by default in the submitted version. Error messages must be printed to `stderr` (not to `stdout`). **If your program detects valid command-line arguments and is able to successfully process the input file, it must exit with a status of 0 (zero), otherwise it must exit with a non-zero status if there are any errors.**

Creating and Submitting the Assignment Software

Your software for this assignment must reside in your CSLAB Linux account under ∼`/cs4414/assign1`. (If you are working in a team of 2 or 3 persons, choose one student's directory to hold the files to be submitted for the team.) This directory must allow read, write, and execute by the TA (`djb4p`) through an ACL that you create. You must use *make* to build your software, and your Makefile

must create an executable binary file called `assign1`. The makefile must also include a `clean` target that will remove any object (.o) files along with the executable binary file. The TA will `cd` to your project directory and type `make clean` followed by `make`. The TA will then test various inputs which may or may not comply with the specification of the exercises. Your program should correctly handle well-formed input and fail gracefully on malformed input.

To submit your assignment, go to the Assignments section of the Collab class site and enter the following information for Assignment 1:

- A list of the students on your team (first and last names, UVA computing ID)
- The full path of the directory that has the assignment software located in it (e.g., `/home/djb4p/cs4414/assign1`)

You must submit the assignment for the entire **team** (not each team member).

The assignment will be graded on the ability of your software to build and execute correctly when tested by the TA. Other factors that will be considered in the grading include:

- How well your code is designed and organized
- Consistent coding style and neatness of your source code
- Documentation (Is there an appropriate number of comments? Are the comments easy to understand?)