CSCI 1120 (Low-Level Computing), Fall 2013

Homework 5

Credit: 20 points.

1 Reading

Be sure you have read the assigned readings for classes through 11/13.

2 Programming Problems

Do the following programming problems. You will end up with at least one code file per problem. Submit your program source (and any other needed 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 5"). You can develop your programs on any system that provides the needed functionality, but I will test them on one of the department's Linux machines, so you should probably make sure they work in that environment before turning them in.

1. (20 points) Write a C program that sorts the lines in a text file using the library function `qsort`. The program should take the name of the file to sort as a command-line argument (and print appropriate error messages if none is given or the one given cannot be opened) and write the result of the sort to standard output.

   To do this, I think you will need to read the whole file into memory. There are various ways to do this, but the method I have in mind (for learning purposes) involves reading the whole file into memory and then building an array of pointers to individual lines. Here is a function you can use (on Linux systems anyway) to determine how much memory to allocate for the file:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
/* returns size of file *filename in bytes, or -1 on error */
int filesize(char * filename) {
    struct stat status;
    if (stat(filename, &status) == -1) {
        return -1;
    }
    else {
        return (int) status.st_size;
    }
}
\end{verbatim}
```

(The above description is deliberately not very detailed. More detailed hints about how to proceed available on request, but I want you to think about the problem yourself first.)

Hints:

- You can use the library function `strcmp` to compare two strings.
- The "sample programs" page contains an example of using `qsort`. (Notice that while it checks the result of the sort for correctness, your program does not need to do that; instead you are to print the results of the sort.)
  sort-improved.c[1]

[1] `http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2013fall/SamplePrograms/Programs/sort-improved.c`