CSCI 1120 (Low-Level Computing), Fall 2010

Homework 5

Credit: 20 points.

1 Reading

Be sure you have read, or at least skimmed, the readings for 10/04 and 10/18, linked from the "Lecture topics and assignments" page[1].

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. (10 points) Write a C program that performs some of the same functions as the UNIX command `grep` — searches in a file for all lines that match a given search string, and prints the ones that do. For example, if you have a file containing the following lines:

   ```
   Hello world!
   There is a German word for worldview that I have forgotten.
   World without end.
   Goodbye!
   ```

   and you search for `world`, the first and second lines should print. (It's easiest to do the match in a case-sensitive way, so the third line does not match.)

   The program should accept two command-line arguments, the first containing the text to search for and the second containing the name of the file to search. (For extra credit, make the program work more like `grep`, which allows searching any number of files, or none, in which case it searches for the specified text in standard input. If you do this, and more than one file is specified, print matching lines in a way that makes it clear which file they came from. `grep` does this by preceding each matching line with the name of the file it came from, e.g.,

   ```
   input.txt: There is a German word for worldview that I have forgotten.
   ```

[1] `http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2010fall/HTML/schedule.html`

but you may think of some format you like better.)

For this problem, read the file a line at a time, but be as careful as you can about reading text — it is very easy to read more characters than will fit into the character array you're putting them in. (I usually recommend using the library function `fgets` for reading text line by line.) If the input file contains lines that are too long, you can just print warning messages about them.

You may find the library function `strstr` useful.

2. (10 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) is to read the whole file into memory and then build 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;
    }
}
```