CSCI 1120 (Low-Level Computing), Spring 2013

Homework 4

Credit: 20 points.

1 Reading

Be sure you have read the assigned readings for classes through 3/06.

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 4"). 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) In a previous assignment you completed a sort program we began in class. Revise this program so that rather than generating random data it reads the values to sort from a file and writes the sorted values to another file. The completed program should take two command-line arguments giving the names of the input and output files. The program should print appropriate error messages if it cannot open the input or output file or if the input file contains anything but a sequence of integers. Since we have not yet talked about how to make arrays larger at runtime, just write the program with a fixed-size array for holding input, and have the program print an error message if the number of input values exceeds the size of the array. It's up to you whether you keep the part of the earlier program that checks whether the sort succeeds; if you do, just have it print to standard output as before.

   Hints:

   - Sample program `sum-from-file.c`[1] illustrates reading a sequence of integers from an input file.

2. (10 points) A very simple way to encrypt text is to rotate each alphabetic character N positions. For example, if N is 1, "abc XYZ 1234" becomes "bcd YZA 1234". (This is obviously not industrial-strength encryption but is good enough to somewhat obscure the plaintext.) Write a C program that implements this scheme. The program should take three command-line arguments: the number of positions to rotate (which for simplicity should be a positive integer), the name of the input file, and the name of the output file. It should print error messages as appropriate (not enough command-line arguments, non-numeric N, input

[1] `http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2013spring/SamplePrograms/Programs/sum-from-file.c`

or output files cannot be opened). For valid arguments, it should encrypt the input file and write the result to the output file.

Hints:

- Library function `strtol` may be helpful in converting a command-line argument string into an integer.
- You don't need to try to read input a line at a time; you can just read and process it a character at a time using `fgetc`, `fputc`, and your own function that encrypts a single character.
- There are probably several ways you could approach encoding each character. One I like (because it doesn't rely on characters being encoded in ASCII — which on most systems these days they are, but C doesn't require it) begins by looking up the character in a string representing the alphabet. Starter code for such a scheme, to encode `int` variable `inchar`:

```
char* lc_alphabet = "abcdefghijklmnopqrstuvwxyz";
char* in_lc_alphabet = strchr(lc_alphabet, inchar);
if (in_lc_alphabet != NULL) {
  /* lower-case character */
  int position_in_alphabet = in_lc_alphabet - lc_alphabet;
  ....
}
```

  `lc_alphabet[position_in_alphabet+1]` then gives you the next character in the alphabet. You could do something similar for uppercase characters, with a string `uc_alphabet`.