ICS SS09 Lab0: MiniSort

Hand out: Sun. June. 13
Final Deadline: Sun. June. 27 20:00:00 No Extension!

1. Introduction

This lab requires the students to implement a simple character based sort program using C Language.

2. Hand out

The file you will need for this and subsequent lab assignments in ICS are distributed using the SVN version control system like the bits lab.

The files checked out from server should be "lab0.pdf", "gen_sort", "minisort_base", and "minisort.c".

3. Task

The program, named "**gen_sort**", generates an input file with multiple 100 byte records. The first 10 bytes is a random key of 10 printable characters. The next 10 bytes is the record number, 0, 1, ... with leading blanks. The remaining 80 bytes contain varying padding characters, like "AAAAAAAAAA", and the "enter" character, which is '\r' and '\n' in linux. The command to execute "**gen_sort**" is "**./gen_sort <#record> <filename>**". You need provide the number of record you want create, and the file name to store the records.

Your task is to implement a program, named "**minisort**", in the file "***minisort.c***". It sorts the records in the input file generated by above program "**gen_sort**". The sorted result is in the **ascending** order of "key", the first 10 bytes of record. If the key of multiple records are the same, then the record with smaller record number is prior.

We also provide the executable file of our implementation, named "**minisort_base**", to help you to verify the result of your implementation. The result should be the same. **NOTE**: If user doesn't provide the name of output file, the result should be printed to console. If the output file is existed, you should **overwrite** it, otherwise you should **create** it. The command to execute "minisort" is "**./minisort <input_filename> [output_filename]**". You need provide the input file, and the output file is optional.

It is not enough to your program that it only works with correct input. You should deal with the mistake from user or system, e.g. wrong input and memory work out. You also should release all resources allocated before program exit, e.g. the open file and dynamic allocated memory.

4. Grading

Part 2 (40' + 30' = 70'): 40' from the correctness and 30' from the robustness

Your code will be compiled with GCC and run and tested on one of the class machines. Your score will be computed out of a maximum of 75 points based on the following distribution:

40: Correctness of program.

- 20: correctly sort the input file
- 5: correctly sort the records with the same key
- 5: correctly deal with the command line
- 5: support to output result to a file
- 5: check the input file, e.g. the size of input file

30: robustness of program.

- 10: no limitation to the size of input (e.g., from 100 Bytes to 100 MBytes)
- 5: release resource before program exit, e.g. free memory and close file
- 5: check error from system, e.g., out of memory error from malloc()
- 5: performance, don't 5 times slower than "minisort_base".
- 5: coding style, e.g., annotation of code

5. Hand in

You only need to commit files "minisort.c" before deadline. But I strongly recommend you to commit it multiple times. **NOTE:** please ensure the success of commit operation. You have to bear all consequences.

!!! *No plagiarism is permitted. If spotted, both plagiarist and the one being plagiarized losses all scores for the lab.*