CS342 Operating Systems
Fall 2019
Project #1

Assigned: Oct 16, 2019, Wednesday.
Due date: Oct 30, 2019, Wednesday, 23:55.

- *Submit through Moodle. Make sure you start submitting one day before the deadline. You can overwrite your submission as many times as you wish. Late submissions will not be accepted (no excuse; no email will be accepted).*
- *You can only learn if you think and do yourself.*

Assignment

Part 1. Develop a program that will create N child processes, besides the main process, to process N input files to find out the k largest integers in those files. N input files will be processed concurrently by N processes. Each input file can contain a very large number of positive integers. The result wil be written to an output file in sorted order, from largest to smallest. k can be a number between 1 and 1000. N can be a number between 1 and 5. Input and output files will be ascii text files. The program will be invoked as followes:

findtopk <k> <N> <infile1> ...<infileN> <outfile>

Each child process will read and process one input file and write its result to an intermediate file. When all child processes finish, the main process can read and process those intermediate files.

You could read and write files using fscanf and fprintf, but in this project you will use **low-level I/O** routines, read() and write(), so that you can practice low level I/O. You will open a file using open() function and close the file using close() function. You can read an input file one character at a time (or few characters). An input file will contain integers seperated by whitespace characters (space, tab, newline). The output file will contain one integer per line. You program will delete the intermediate files when it terminates.

Part 2. Do the same project this time using POSIX message queues to pass information from child process to the parent process. No intermediate files will be used. The name of the program will be findtopk_mqueue in this case.

Part 3. Do the same project using POSIX threads. For each input file a seperate thread will be created. The name of the program will be findtopk_thread in this case. Thread global variables will be used to pass information from a thread to another thread. No need to use message queues or intermediate files.

Part 4. Experimentation. Measure the running time of your programs for various k, N, and filesizes (number of integers in a file). Report your results (in tables or plots).Try to draw some conclusions. Fix k and number of integers in a file. For example, k can be 1000, and number of integers in a file can be 1000000. Measure the running time of your programs for N = 1, 2, and 3. Report your results. Try to draw conclusions. Write a report at the end of experiments.

Submission

Put all your files into a directory named with your Student Id. In a README.txt file write your name, ID, etc. The set of files in the directory will include README.txt, Makefile, findtopk.c, findtopk_mqueue.c, findtopk_thread, report.pdf. When we type make, all your programs should be complied and the related executables should be generated. The tar and gzip the directory. For example a student with ID 21404312 will create a directory named “21404312” and will put the files there. Then he will tar the directory (package the directory) as follows:

tar cvf 21404312.tar 21404312

Then he will gzip the tar file as follows:

gzip 21404312.tar}

In this way he will obtain a file called 21404312.tar.gz. Then he will upload this file in Moodle.

Late submission will not be accepted (no exception). A late submission will get 0 automatically (you will not be able to argue it). Make sure you make a submission one day before the deadline. You can then overwrite it.

Tips and Clarifications

- Work incrementally. Be persistent.