CS342 Operating Systems – Fall 2021
Project #1 – Processes, IPC, and Threads

Assigned: Oct 13, 2021.
Due date: Oct 28, 2021. Document version: 1.1

This project will be done individually. You will use the C programming language. You will develop your programs in Linux.

Part A (50 pts): In this project you will implement an application that will compute some simple statistics over an integer data set. The application will have two parts: **client** and **server**. The client program will be called as statclient. The server program will be called statserver. The client will get a request from a user about what statistics to compute and the related parameters. Then this request will be sent to the server over a POSIX **message queue**. The server will receive this request and compute the requested statistics using multiple **child processes** that will act on data concurrently. When needed, child processes will pass data to the parent using **pipes**. There will be one pipe per child. After computing the result, the server (parent) will send the result to the client using another message queue. After receiving the result, client will print that out.

The following are the **requests** that a user can type at the client prompt (meaning of the request is self-explanatory). After completing one request, the client will wait for another request.

count
avg
max
count <start> <end>
avg <start> <end>
range <start> <end> <K>

The request "count" is to find the number all integers in the whole data set. The request "avg" is to find the average of all integers in the whole data set. The request "count <start><end>" is to find the number of integers in range [<start>, <end>], inclusive. The request "avg <start><end>" is to find the average of all integers in range [<start>, <end>]. The request "range <start> <end> <K>" is to find the top <K> integers in range [<start>, <end>] in ascending order. In this request, <K> is the maxiumum number of integers to return. If the [<start>, <end>] has more integers than <K>, only <K> largest of them will be returned in ascending sorted order. <K> can be at most 1000, and at least 1.

Between client and server, there will be two message queues; one for sending the request, and the other for receiving the result. The server will compute the needed result from the integer data stored in multiple files. The names of the files will be indicated to the server at server start time. The following is how we will invoke the server:

statserver <N> <filename1> <filename2> … <filenameN>

Here <N> is the number of input data files. The min value of <N> is 1. The max value of <N> is 10. For example, if <N> is 3, then:

./statserver 3 infile1.txt infile2.txt infile3.txt

An input file is an ascii file and will contain positive integers (one integer per line) There may be a very large number of integers (millions) in a file. The client will be invoked as follows:

./statclient

Then the client program will print out a command prompt asking a request to be typed by the user. Some example requests are:

avg
avg 100 3000
count
count 75 340000
range 20000 25000 50
max

When the request "range 20000 25000 50" is entered, no statistics will be computed, but the largest 50 numbers between 20000 and 25000 will be returned from the server to the client in sorted order (from smaller to larger). Duplicates will not be eliminated.

When the server receives a request, it will create as many child processes as the number of input files. That means, each input file will be processed by another child process. Hence the children will process the files concurrently. The results they obtain, if any, will be sent to the parent process via the pipes. When a child finishes sending data, it will indicate this by sending -1. So, when the server main process receives -1 form a pipe, it understands that no more data will arrive from that pipe and that child.

Part B (30 pts): Do the same project using **threads** now, instead of child processes. This time you will not use pipes. Global variables can be used by all server threads (main thread and the created threads).

Part C – Experiments (20 pts): Compare the running time of child process usage and thread usage. Which one is faster. Do experiments and show results. Support your conclusions with your experiment results. Plot some graphs.

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, statserver.c, statclient.c, and 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

- Start early, work incrementally.
- In server, create the pipes first, before creating the children using fork().