CS342 Operating Systems – Spring 2022
Project #1 – Processes, IPC, and Threads

Assigned: Feb 10, 2022.
Due date: Mar 3, 2022, 23:59. Document version: 1.0

This project will be done in groups of two students. You can do it individually as well. 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 a specified histogram over an integer data set that resides in a set of input ascii files. A histogram is an approximate representation of the distribution of numerical data. A histogram has ranges of values (intervals or buckets) and for each range (interval) it gives the count (frequency) of the data set values falling into that range. In computing the histogram, the application will use the integers that are relevant to the specified histogram.

The application will have two parts: client and server. The client program will be called as histclient. The server program will be called as histserver. The client will be invoked with a set of parameters that specify the requested histogram, i.e., the number of intervals, the width of an interval, and the start of the first interval. Then the client will prepare a request and send it to the server over a message queue. The server will receive the request and will compute the specified histogram. When started, the server will take the names of the input files constituting the data set as command line parameters. To compute the histogram, the server will use multiple child processes. If there are N input files, then N child processes will be used. Each child process will act on a separate input file and will compute its part of the histogram by considering the integer values stored in that input file. That means, for each integer read from the file, the child will increment the count (frequency) for the histogram interval into which the integer value is falling. Child processes will run concurrently. After computing its part of the histogram, a child process will send the histogram to the parent via a message queue.

Child processes will use a single message queue to pass their histogram data to the parent. Hence the message queue will be shared by all children. Therefore, while sending data to the parent, a child can also include its pid in the data message, so that the server can know from which child the message is coming. After collecting the information from the children, the server parent process will combine the results and will obtain the final histogram data. Then the server will send this histogram data to the waiting client using another message queue. Hence there will be two message queues used between the client and server. And there will be a third message queue used between the server child processes and the server parent. In total the application will use 3 message queues. After receiving the histogram information, the client will print that out.

The client program will take the following parameters:

histclient <intervalcount> <intervalwidth> <intervalstart>

<intervalcount> is the number of intervals in the histogram. <intervalwidth> is the width (integer range) of each interval (a half-open range). <intervalstart> is the integer at which the first interval starts. An example invocation can be like the following:

./histclient 5 200 1000

Here, histogram specification says that there are 5 intervals each having a width of 200 and the first interval starts at integer 1000. Then the intervals will be: [1000,

1200), [1200, 1400), [1400, 1600), [1600, 1800), [1800, 2000). An integer 1534, for example, will fall into the interval [1400, 1600). An integer 1600 will fall into the interval [1600, 1800). An integer 2500 will not fall into any of the intervals of the histogram, therefore it will not be considered (not counted).

The server will have the following parameters:

histserver <N> <infile1> <infile2> … <infileN>

<N> is the number of input files. <infile1> … <infileN> are the names of the input files. 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 min value of <N> is 1. The max value of <N> is 10. The following is a sample invocation of the server for <N> being 3.

./histserver 3 infile1.txt infile2.txt infile3.txt

The server will find for each interval, how many integers are falling into that interval. Then, the server will send this information to the client. Finally, the client will print out intervals and the counts (frequencies). An example client output can be like the following:

[1000, 1200): 23
[1200, 1400): 346
…
[1800, 2000): 57

In this example, there are 346 integers in the interval [1200, 1400). Your client should print out the result in this format. You will not plot the histogram. You will just print out the related histogram data as above without plotting. Nothing else will be printed out by the programs.

After printing the result, the client will send another message to the server indicating that the client has finished. Then the client will terminate. The server, upon receiving this message, will also terminate. But before terminating, the server parent will make sure that all children have terminated (will wait for their termination) and will remove all the message queues from the system. In this way, we leave a clean system behind for proper running of the programs again.

Normally, we may not implement such an application in a client-server manner. But here the goal is to practice what we have learned in the classroom. In this way, besides seeing the communication happening between children and parent, you will also see the communication happening between different programs.

Part B (25 pts): Do the same project using **threads** now, instead of child processes. Each input file will be processed by another server thread. If there are N input files, there will be N new threads (worker threads) created by the server, besides the main thread. In the server program you do not need to use a message queue between main thread and the worker threads. Instead, global variables can be used by the threads to pass information. This time the program names will be: histserver_th and histclient_th. The histclient_th program will essentially be the same with histclient program, but anyway we should still have a separate histclient_th program (and its source, histclient_th.c).

Part C – Experiments (25 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 experimental results. Plot some graphs.

Submission

Put all your files into a directory named with your Student ID (if done as a group, the ID of one of the students will be used). In a README.txt file, write your name, ID, etc. (if done as a group, all names and IDs will be included). The set of files in the directory will include README.txt, Makefile, **histserver.c, histclient.c, histserver_th.c, histclient_th.c,** report.pdf, and any other file you find necessary (like one or more header - .h - files). Do not include executable files in your tar package. Then 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 into Moodle.

Tips and Clarifications

- You will use four .c files. Your Makefile should compile the four C programs by just typing make.
- POSIX message queues will be used. Not System V message queues.
- Start early, work incrementally.
- Max value for <intervalcount> can be 1000. Min value is 1.
- Max value for <intervalwidth> can be 1000000. Min value is 1.
- Server will create all message queues. They can be created before the children are created. The client program will know the names of the message queues that will be used between the client and server.
- You can assume that the value of an integer in a file will not be bigger than 2^30. It is a positive number.
- histclient.c and histclient_th.c will have essentially the same code. But anyway, have them both. For processes, use histclient.c, for threads use histclient_th.c.
- You can use any method you like to indicate the end of data from a child to the parent.
- You can use more than one header file (.h file) if you wish.