GTU Department of Computer Engineering
CSE 344 - Spring 2022
Midterm Project Report

Hasan Mutlu
1801042673

How I Solved This Problem?

I have solved this problem by implementing all the techniques we have learned so far. I used file open/close and read/write, file locking, signal handling, creating multiple processes, FIFO's, pipes and shared memory.

Design Decisions

- ➔ All file read/write operations except stdout outputs are done by using read() and write() functions, along with snprintf(), mostly.
- ➔ Files are locked and unlocked when valid output will be written.
- ➔ When creating multiple processes, their process ID's and pipe file descriptors are saved into a worker struct array in serverY. In order to create a pool of processes, I am influenced by the book UNIX® Network Programming, Volume 1, where creating preforked processes is described in detail.
- ➔ In order to see the status of workers, whether they are processing client data or are they available, I used shared memory. I created a worker status tables array to hold status information. At the end of this array, current number of busy processes and number of invertible and non-invertible matrices information are also hold.
- ➔ Client sends requests using FIFO, as it should be. Requests are sent as fixed size, using a request struct.
- ➔ ServerY reads the FIFO. When a new request arrives, it checks the pool of workers to find an avilable worker. When it finds, it marks that worker as busy, increments busy worker's counter and sends the request to its worker using that workers pipe file descriptor.
- ➔ If there isn't an available worker, it keeps checking the pool until it finds one. It doesn't forward the request to serverZ, unfortunately.
- ➔ When a new request arrives to worker, it calculates the determinant of matrix, then prints whether the matrix is invertible or not. After that, it sends the determinant of matrix to client, using client's FIFO. After that, it sleeps for seconds provided in command line argument, as it is states in PDF. It increments the number of handled matrices information in the shared table, marks itself as available and proceeds to wait for next request.
- ➔ When determinant of matrix arrives to client, it prints the output message depending on the determinant and exits.
- ➔ When ServerY recieves signal, it terminates its workers, prints required information, closes files, frees allocated memory and exits gracefully.

Achieved and Failed Requirements

- ➔ All requirements of ServerY and its workers are working fine, except forwarding requests to serverZ when all workers are available.
- ➔ I implemented ServerZ but it didn't work as it should be. So I excluded it from the program. Its codes are still present but thefunction call to create ServerZ is commented out.