Systems Programming Midterm Project Report

In this project, we are supposed to simulate server-client communication using signals, pipes and multiple processes. We will have 3 different programs, timerServer, seeWhat and showResults. Program timerServer is a server program receiving request from clients and handling them. Program seeWhat is a client program making requests for data from server, doing operations on it and sending it to the printer. And finally, showResults is a printer program writing all data received to the standard output and log file.

Program timerServer needs to do three main things, that is handle client requests, keep track of all running processes, and write its logs to a log file. All these three operations are done in child processes running parallelly with timerServer. To keep track of the all new processes I fork a child at the start of the timerServer calling keepTrackOfNewConnections function which creates a fifo to which new programs will write their pids, these pids will be read and saved by timerServer and processes with these pids will be sent a SIGUSR2 signal together with server's pid. This child runs parallelly with timerServer. Thus, all new processes will first exchange pids with timerServer. In this structure only timerServer is aware of the all running processes. When SIGINT is received, timerServer distributes it to all known processes. Thus, if any of the running processes receives SIGINT, in order to signal all the other processes, the only thing needed to do is resend it to the timerServer. Other child process call writeLogs function and runs parallelly with timerServer, this child just writes logs for timerServer. Program timerServer itself blocks all signals and waits for CPU ticks entered, then unblocks SIGINT and SIGUSR1(request signal) and checks for any. If request is received it checks if there is a pid in mainfifo, then if there is it forks a new process which would generate an invertible matrix for the client requesting it. The client requesting a matrix (seeWhat) needs to take turn for the server, this is done by writing it's pid to the mainfifo, so timerServer reads this clients pid and generates a child to serve this client. If there is no pid in the mainfifo, that means that the signal is an old signal and there is actually no client waiting to be served, so timerServer goes on with its routine.

Program seeWhat is cinchonized with server, and waits for server to start, when server starts it exchanges pids with server and then starts sending request to server until data is received, if data is received seeWhat starts reading its data, then forks 2 processes to do operations on this data and when they are done writes its logs and send results to showResults program. Program seeWhat has only one log file, to which it writes all its data operations results.

Program showResults is also cinchonized with server and also waits for server to start. If server has started it creates a fifo to which all seeWhat programs will write their

results and then starts reading this fifo and writing its own logs and output messages to stdout.

I have made a lot of testing of the whole system, running it with different ticks in second (in range from 1 to 10000), different matrix sizes. In almost all of the cases system run as expected. In some few cases after receiving SIGINT I had some of the seeWhat processes left blocked on pipe IO. Later on after a lot of more testing this problem never appeared again. The cause of the problem could not be found. I have extensively tested matrix library which does operations on matrices. Inverse matrix generator function is guaranteed to generate invertible matrix, and all of the quarter of this matrix are also guaranteed to be invertible. Function for finding 2d convolution uses identity kernel, so after this operation on a matrix, matrix stay intact, so result2 is always 0.

Some functions for finding matrix inverse and determinant were taken from Wikipedia page : http://www.songho.ca/dsp/convolution/convolution.html.
Function for finding 2d convolution was taken from this website : http://www.songho.ca/dsp/convolution/convolution.html.
All of the wrapper functions and helper functions were written by me.