

System Programming Midterm Report

Name: Mustafa Tokgöz

Student Id: 171044077

1 HOW TO SOLVE THIS PROBLEM

Firstly, In client program, At the begin of the program I initialize the signal handler for SIGINT signal. I read the arguments from argv and separate them as variables. I start the time for calculating execution time for client. After that I read file from given argument with function read from file. In this function I read the file character by character and I merge the characters and convert them integer value for my array. I use this function 2 times because one of them is for counting array size and the other is assigning them to new sized array. After that I create client fifo like in slayts. Set the struct values like pid and n (size of square matrix). After that I open the server file and write the struct and matrix array to there. After that I read the result from client fifo and print them as result. I also write timestamp function to show the time and date. Then I close the files. In server program, I create a struct for shared memory for keeping track of server. Firstly, I create a file and at the end of the program I remove the file. So if the file is there, then program ends because only 1 instantiation can occur. After that I write the become daemon function to have no controlling terminal like in book. After that I create signal handler for SIGINT signal. To send SIGINT to server program we should send kill -2 pid of server because server program is daemeon. After that I creates pipes for sending matrix to children. After that I separete the arguments from argv as variables. After that I open log file to write output of the server. After that I creates the named posix semaphores with sem_open. I will use this semaphores for synchronization for child processes. After that I initilize the shared memory for keeping number of request, number of inveritble etc. After that I create children of the serverY. I also create semaphrores for children. After that I call serverY function. In this function I create server side fifo and read matrix from there and send it to its pipe in infinite loop. After that I post the seamphore Y value for start any of child begin to calculate. Also I use semaphore W for counting unavailable children. If value of semaphore W is equal to poolsize then I will forward them to server Z. In child processes, There is infinite loop and in this infinite loop I wait the seamphore Y to start the reading from pipe. If semaphore Y posted, then one of children post the semaphore W for counting

working children and start to read matrix from pipe. After reading I print to log file and increase the request number of shared memory element. After that I calculate the determinant of the matrix. Then sleep the given time. After that It sends the result to client with client fifo. Then print the result and wait the semaphore W which we posted. In SIGINT handler , I close files, shared memories, semaphores and send resource back.

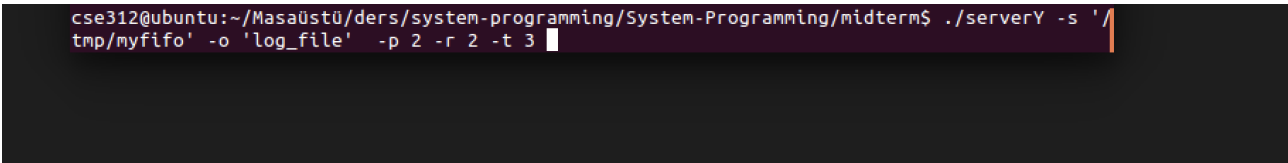
2 DESIGN DECISIONS

- I use `dprintf` to print to log file
- I named posix semaphores to synchronize to each other.
- I use only parent pipe to communicate with its children
- I use read and write for communication between server and client fifos.
- I use shared memory for keeping track of server.
- I use cofactor method to find if the matrix is invertible or not.
- I use semaphore Y to start the children.
- I use semaphore W for counting working process.
- I create semaphore Z for start the server Z children.

3 WHICH REQUIREMENTS I ACHIEVED AND FAILED

I achieved Server Y and Client but I didn't achieved server Z. I achieved become daemon, communicate with fifo, calculating stuffs, signal handler, creating children, synchronize the children and parent, communicate with fifo, request and response, server client.

4 INPUT-OUTPUT EXAMPLE



```
cse312@ubuntu:~/Masaüstü/ders/system-programming/System-Programming/midterm$ ./serverY -s '/tmp/myfifo' -o 'log_file' -p 2 -r 2 -t 3
```

```
cse312@ubuntu:~/Masaüstü/ders/system-programming/System-Programming/midterm$ ./run.sh
Sat Apr 16 11:35:03 2022 , Client #3779 (readfile) is submitting a 3x3 matrix
Sat Apr 16 11:35:06 2022 , Client #3779: the matrix IS NOT invertible, total time 3.00 second, goodbye
Sat Apr 16 11:35:06 2022 , Client #3790 (readfile2) is submitting a 4x4 matrix
Sat Apr 16 11:35:09 2022 , Client #3790: the matrix is invertible, total time 3.00 second, goodbye
Sat Apr 16 11:35:09 2022 , Client #3791 (readf.csv) is submitting a 3x3 matrix
Sat Apr 16 11:35:12 2022 , Client #3791: the matrix IS NOT invertible, total time 3.00 second, goodbye
cse312@ubuntu:~/Masaüstü/ders/system-programming/System-Programming/midterm$
```

```
Sat Apr 16 11:35:03 2022 , Server Y (log_file, p=2, t=3) started
Sat Apr 16 11:35:03 2022 , Worker PID#3776 is handling client PID#3779, matrix size 3x3, pool busy 1/2
Sat Apr 16 11:35:06 2022 , Worker PID#3776 responding to client PID#3779: the matrix IS NOT invertible.
Sat Apr 16 11:35:06 2022 , Worker PID#3777 is handling client PID#3790, matrix size 4x4, pool busy 1/2
Sat Apr 16 11:35:09 2022 , Worker PID#3777 responding to client PID#3790: the matrix is invertible.
Sat Apr 16 11:35:09 2022 , Worker PID#3776 is handling client PID#3791, matrix size 3x3, pool busy 1/2
Sat Apr 16 11:35:12 2022 , Worker PID#3776 responding to client PID#3791: the matrix IS NOT invertible.
Sat Apr 16 11:35:25 2022 , SIGINT received, terminating Z and exiting server Y. Total requests handled: 3,
```

```
, t=3) started
ling client PID#3779, matrix size 3x3, pool busy 1/2
ling to client PID#3779: the matrix IS NOT invertible.
ling client PID#3790, matrix size 4x4, pool busy 1/2
ling to client PID#3790: the matrix is invertible.
ling client PID#3791, matrix size 3x3, pool busy 1/2
ling to client PID#3791: the matrix IS NOT invertible.
ating Z and exiting server Y. Total requests handled: 3, 1 invertible, 2 not. 0 requests were forwarded.
```