GEBZE TECHNICAL UNIVERSITY
COMPUTER ENGINEERING

SYSTEM PROGRAMMING LECTURE

MIDTERM PROJECT

MIDTERM REPORT

AHMET FURKAN KURBAN

1801042674

APRIL 16

PROBLEM SOLUTION

I started by creating server and client files. First of all, I read the matrix from the file on the client side and sent it to the server with fifo. After creating the server children, it works in for. Thus, we can send clients to the server all the time. In addition, the children of the server also work in for. I separated them by pid. With the fork, I sent the matrix received by the server to the children with a pipe. In each pipe read, all children are blocked, so that only one child reads and continues to work. Each child working increases and decreases the number of clients using signal (sigusr1, sigusr2) by one. Thus, I control the transition to server z.

The matrix received by the child is sent to the determinant formula. Then it sends the result to the client with a fork. Since we are in the loop, the children and the parent can work again. The parent is blocked in the fork, the children are blocked while reading the pipe. Meanwhile, the child who has finished his work is writing to the file. While finding the determinant, I blocked the sigint with the mask because I prevented the memory work done within the function from getting interrupts.

When the number of clients reached pool size, I switched to server z. There are the same operations here, the only difference is that the server communicates with the children using shared memory.

In this part, I prevented 2 children from reading from memory at the same time by using semephore and preventing the other data from coming in without processing the previous data. I made my process deamon using the deamon function. When the sigint signal came, I closed the files and unlinked the semephores. I made the memoris free and exited.

DESIGN DECISIONS

This is how I thought of the design. The server should work continuously and the one that finishes its job should work again. That's why I constantly fork in the for and send it to a child with a pipe. Therefore, the one who finished the job was able to work. I created the processes before for and put a break so that the children do not create processes again. I tried to catch the synchronization with the Semephore and signals.

REQUIREMENTS THAT I ACHIVED

The system works in general, writes to file sequentially and works synchronously with semephore. Signals are handled, determinants are found and response is returned to client. Server Y and Z are being created, and not shutting down until sigint arrives.

REQUIREMENTS THAT I FAILED

As for sigint, the handler is running more than once and I couldn't prevent it from writing to the file. I used too much static, I couldn't send the pointer array with the pipe, so I made it static. When the sign came, I tried the method of increasing the global value in the handler and checking it in the code with the check function, but I was unsuccessful. Since there are more than one child process, I think I've been constantly getting file discriptor, fifo opening, file opening errors. so I wrote everything in sighandler.