# System Programming Midterm Project Report

Mustafa Karakaş 18.04.2022 Instructor: ErchSeran Aptoula

## ServerY

When servery starts running, it opens a shared memory area with specific name and O_EXCL flag ,which makes shm_open give an error if shared memory with given name exists, to be able to understand if another servery currently runs or not. Then servery creates pipe to communicate workery processes and serverz process. Then creates workerY processes and serverz process. Then creates fifo with the given name to communicate with the clients. Then program loop starts and it will be blocked until client sends process id and its matrix's data via fifo. When server takes this information, it checks the number of workerY available processes. If there is no available workerY process, then it will send data to serverZ.

How does serverY keeps track of number of available workerY processes ?

It actually holds number of busy processes. When the program starts, available number of workerY processes is known and there is no busy process. Each time when serverY assigns a job to workerY, it increases the number of busy processes. Each workerY sends a SIGUSR2 signal to serverY when it finishes given job. So, serverY decreases the number of busy processes.

How does serverY sends data to available workerY process if there is any ?

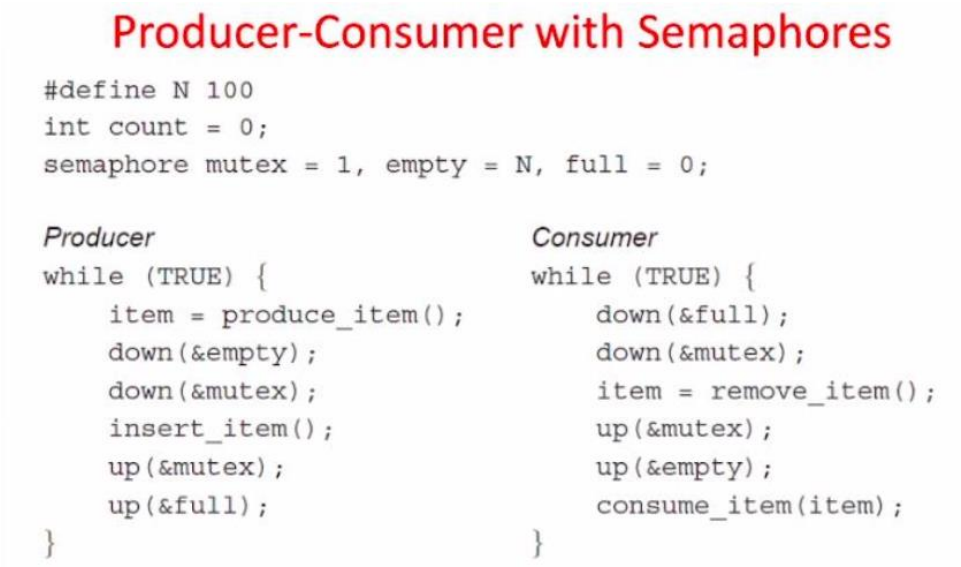
It puts taken data to pipe, and creates a mutex. So, no two processes can read at the same time. Each available workerY waits for the mutex and if it passes the mutex, it is blocked until data is available. We need mutex here because, there are multiple read operations. And pipe does not guaranties that, the process makes the first read operation will also make the next one. Mutex guaranties that. After all read calls, mutex is up and other processes can also read.

## WorkerY

Takes client's data and checks if the given matrix is invertible or not and sends result to client via fifo whose name contains client's id.

## ServerZ

Creates a shared memory, and puts data taken from pipe to here. Creates workerZ processes to take care of calculations. ServerZ and WorkerZ has procucer customer relationship. ServerZ puts the data and WorkerZ processes takes the data. To be able to avoid race conditions, semaphores are used with the help of template below.




ServerZ puts –1 at the end of the queue. So that consumers understand there is no more data. When data is placed to the shared memory, id of client is placed first. Because of that –1 is not a valid process id, no conflict happens. When the consumer takes data from shared memory, it takes the first one and removes it. If the first index holds –1, that means there is no data in shared memory. Shared memory has fixed size.

## Client

First of all client reads the matrix from given file. Then creates fifo whose name contains process id of client. Client communicates with serverY via given server fifo. It gives its data to serverY and waits until result is given from the fifo created.

SIGINT signal is handled. Program lefts no zombie processes.

```
mustafa@mustafa-Aspire-A515-52G:~/Desktop/hw3/hw3$ ./serverY -s '/tmp/seqnum_sv' -o out.txt -p 2 -r 3 -t 3
2022-04-18 11:26:10 Server Y (out.txt, p=2, t=3) started
2022-04-18 11:26:10 Instantiated server Z
2022-04-18 11:26:10 Z:Server Z (out.txt, t=3, r=3) started
2022-04-18 11:26:27 Worker PID#29860 is handling client PID#29929, matrix size 6x6, pool busy -1/2
2022-04-18 11:26:30 Worker PID#29860 responding to client PID#29929: the matrix IS NOT invertible.
2022-04-18 11:26:30 Worker PID#29861 is handling client PID#29930, matrix size 6x6, pool busy -1/2
2022-04-18 11:26:33 Worker PID#29860 is handling client PID#29931, matrix size 6x6, pool busy -1/2
2022-04-18 11:26:33 Worker PID#29861 responding to client PID#29930: the matrix IS NOT invertible.
2022-04-18 11:26:33 Worker PID#29861 is handling client PID#29932, matrix size 6x6, pool busy -1/2
2022-04-18 11:26:36 Worker PID#29860 responding to client PID#29931: the matrix IS NOT invertible.
2022-04-18 11:26:36 Worker PID#29861 responding to client PID#29932: the matrix IS NOT invertible.
2022-04-18 11:26:37 Worker PID#29860 is handling client PID#29935, matrix size 6x6, pool busy -1/2
2022-04-18 11:26:40 Worker PID#29860 responding to client PID#29935: the matrix IS NOT invertible.
2022-04-18 11:27:26 Worker PID#29861 is handling client PID#29971, matrix size 6x6, pool busy -1/2
2022-04-18 11:27:26 Worker PID#29860 is handling client PID#29972, matrix size 6x6, pool busy -1/2
2022-04-18 11:27:27 Forwarding request of client PID#29973 to serverZ, matrix size 6x6, pool busy 2/2
2022-04-18 11:27:27 Z:Worker PID#29862 is handling client PID#29973, matrix size 6x6, pool busy -1/3
2022-04-18 11:27:27 Forwarding request of client PID#29974 to serverZ, matrix size 6x6, pool busy 2/2
2022-04-18 11:27:27 Z:Worker PID#29862 is handling client PID#29974, matrix size 6x6, pool busy -1/3
2022-04-18 11:27:28 Forwarding request of client PID#29975 to serverZ, matrix size 6x6, pool busy 2/2
2022-04-18 11:27:28 Z:Worker PID#29862 is handling client PID#29975, matrix size 6x6, pool busy -1/3
2022-04-18 11:27:29 Worker PID#29861 responding to client PID#29971: the matrix IS NOT invertible.
2022-04-18 11:27:29 Worker PID#29860 responding to client PID#29972: the matrix IS NOT invertible.
2022-04-18 11:27:30 Z:Worker PID#29863 responding to client PID#29973: the matrix IS NOT invertible.
2022-04-18 11:27:30 Z:Worker PID#29864 responding to client PID#29974: the matrix IS NOT invertible.
2022-04-18 11:27:31 Z:Worker PID#29865 responding to client PID#29975: the matrix IS NOT invertible.
```

## Features not implemented:

ServerY is not deamon. Program only writes small portion of its output to given log file. It writes all of them to stdout. Number of busy processes is not given. At the end of the program it does not writes number of requests handled and how much of them are invertible and not invertible.