CSE344 SYSTEM PROGRAMING

MIDTERM

REFİK ORKUN ARSLAN
151044063

CLİENT.C

Read line by line data csv and add bufferk

```
while((len = readline (buffer, 1024, argv[4], &offset)) != -1 && !signal_flag)
	{
                                        z=commaCounter(buffer);

                                        if (first == 0)
                                        {
                                                first = z;
                                        }

                                        if (z > 0)
                                        {
                                                strcat(buffer1[k],buffer);
                                                if (i == first)
                                                {
                                                        strcat(buffer1[k],"\n");
                                                        counter[k]=i+1;
                                                        k += 1;
                                                        i = 0;
                                                        first = 0;
                                                }
                                                else {
                                                        strcat(buffer1[k],"\n");
                                                        i++;
                                                }



                                        }


                                        memset(&buffer,0,sizeof(buffer));


	}
```

Create Child process

```
if (parent == getpid() && !signal_flag)

pid_t child= fork();
```

Open fifo and need assigned variable (we will need it later.)
Sent with request struct via fifo

```
struct request req;
struct response resp;
char clientFifo[CLIENT_FIFO_NAME_LEN];

 umask(0);
snprintf(clientFifo, CLIENT_FIFO_NAME_LEN, CLIENT_FIFO_TEMPLATE,(long) getpid());
                                                    if (mkfifo(clientFifo, S_IRUSR | S_IWUSR | S_IWGRP)
        == -1 && errno != EEXIST)
perror("mkfifo");
if (atexit(removeFifo) != 0)
perror("atexit");

 req.pid = getpid();
req.seqLen = strlen(buffer1[n]);
strcpy(req.buf,buffer1[n]);
req.matrixcounter=counter[n];
```

Sent with request struct via fifo
Wait response via fifo

```
if (write(serverFd, &req, sizeof(struct request)) !=sizeof(struct request))
perror("Can't write to server");
clientFd = open(clientFifo, O_RDONLY);
if (read(clientFd, &resp, sizeof(struct response))!= sizeof(struct response))
perror("Can't read response                    from server");
```

serverY.C

We created a child process and submitted it with execve.
Parent continues.

```
pid_t execpid=fork();
        if(execpid==0)
        {
                char* arg[] = { word1,word2,word3,pipeExec,word4,NULL };
                 int ret =execve("./serverZ",arg,envp);
                if(ret ==-1)
                {
                        perror("execve failed");
                }
        }
```

In case of CTRL+C, it keeps the number of requests up to that point.

```
glob_var=mmap(NULL,sizeof *glob_var,PROT_READ|PROT_WRITE,MAP_SHARED |
MAP_ANONYMOUS,-1,0);
*glob_var=0;
```

In case of CTRL+C, it keeps the number of requests up to that point.

```
glob=mmap(NULL,sizeof *glob,PROT_READ|PROT_WRITE,MAP_SHARED |MAP_ANONYMOUS,-1,0);
```

```
*      *glob=0;
```

Keep pid of the request

```
int xx;
int *x;

xx = shmget(IPC_PRIVATE, numberOfProcesses*sizeof(int), IPC_CREAT | 0666);
x = (int *) shmat(xx,NULL,0);
```

Keep idle of process

```
int zz;
int *z;

zz = shmget(IPC_PRIVATE, numberOfProcesses*sizeof(int), IPC_CREAT | 0666);
z = (int *) shmat(zz,NULL,0);
```

FORKİNG
OPEN FİFO
It then enters an infinite loop.

First control if(signal_flag)

If it enters here, the CTRL + C command is entered and how many requests have come so far and how many of them are invertible or not, the log file is pressed and it breaks and exits the infinite loop.

Second control if(parentpid==getpid() && !signal_flag)

Enter parent process ,read request comes from client and write pipe

```
read(serverFd, &req, sizeof(struct request)) != sizeof(struct request)) {
fprintf(stderr, "Error reading request; discarding\n");
continue;
}
write(pfd[i][1], &req, sizeof(struct request)) !=sizeof(struct request))
perror("parent - partial/failed write");
                                    }
```

İdle control
if(z[i]==0) (idle) //sequentially checks and discards the idle task
{
z[i]=1;(running)
...

sleep(5); //Returns to idle state after sleep
z[i]=0

serverZ.C

The fifo tip, argument variable and necessary variables were assigned with execve.
Create fifo

Used for the same operations as the serverY

```
glob_var=mmap(NULL,sizeof *glob_var,PROT_READ|PROT_WRITE,MAP_SHARED |MAP_ANONYMOUS,-1,0);
 *glob_var=0;

  glob=mmap(NULL,sizeof *glob,PROT_READ|PROT_WRITE,MAP_SHARED |MAP_ANONYMOUS,-1,0);
 *glob=0;
    int xx;
            int *x;

            xx = shmget(IPC_PRIVATE, numberOfProcesses*sizeof(int), IPC_CREAT | 0666);
            x = (int *) shmat(xx,NULL,0);

            int zz;
            int *z;

            zz = shmget(IPC_PRIVATE, numberOfProcesses*sizeof(int), IPC_CREAT | 0666);
            z = (int *) shmat(zz,NULL,0);
```

Showing space via shared memory has been enabled. Parent and child locations have been marked.

```
z_plus = mmap(NULL, sizeof *z_plus,PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1,
0);
child_plus = mmap(NULL, sizeof *child_plus,PROT_READ | PROT_WRITE, MAP_SHARED |
MAP_ANONYMOUS, -1, 0);
*z_plus = 0;
*child_plus = 0;
```

Our shared memory, which is the main thing, was opened with the mmap method

```
oo = shmget(IPC_PRIVATE, 150*sizeof(struct request), IPC_CREAT | 0666);
o = (struct request*) shmat(oo,NULL,0);
```

Forking

It then enters an infinite loop.

First control if(signal_flag)

If it enters here, the CTRL + C command is entered and how many requests have come so far and how many of them are invertible or not, the log file is pressed and it breaks and exits the infinite loop.

Second control if(parentpid==getpid() && *z_plus==*child_plus && ! signal_flag)

Since the processes are blocked during reading,Doing the operations here. Because the server can receive requests continuously.
Requests are written to shared memory.

Third control if(parentpid==getpid() && *z_plus==(*child_plus +1) && ! signal_flag)
This is where the parent enters to continue when the read is blocked.
Since all requests are received, only the necessary appointments are made.

Forth Control else if (z[keep_id] == 1 && !signal_flag)
where child processes come in
Make fifo
Calculating matrix

OUTPUT

Client

```
orkun@orkun:~/Desktop/son2$ ./client -s Orkun -o data.csv
Client output Mon Apr 18 11:41:32 2022

Client PID# 12082 (data.csv) is submitting a 5x5 matrix
Client 12082: the matrix is invertible,total time .784 seconds,goodbye
Client output Mon Apr 18 11:41:32 2022

Client PID# 12079 (data.csv) is submitting a 3x3 matrix
Client 12079: the matrix is invertible,total time .2123 seconds,goodbye
Client output Mon Apr 18 11:41:34 2022

Client PID# 12091 (data.csv) is submitting a 5x5 matrix
Client 12091: the matrix is invertible,total time 2.2266 seconds,goodbye
Client output Mon Apr 18 11:41:34 2022

Client PID# 12090 (data.csv) is submitting a 3x3 matrix
Client 12090: the matrix is invertible,total time 2.7611 seconds,goodbye
Client output Mon Apr 18 11:41:36 2022

Client PID# 12092 (data.csv) is submitting a 3x3 matrix
Client 12092: the matrix is invertible,total time 4.6119 seconds,goodbye
Client output Mon Apr 18 11:41:36 2022

Client PID# 12093 (data.csv) is submitting a 4x4 matrix
Client 12093: the matrix is invertible,total time 4.7740 seconds,goodbye
Client output Mon Apr 18 11:41:38 2022

Client PID# 12084 (data.csv) is submitting a 4x4 matrix
Client 12084: the matrix is invertible,total time 6.5620 seconds,goodbye
Client output Mon Apr 18 11:41:38 2022

Client PID# 12085 (data.csv) is submitting a 3x3 matrix
Client 12085: the matrix is invertible,total time 6.52841 seconds,goodbye
Client output Mon Apr 18 11:41:40 2022

Client PID# 12080 (data.csv) is submitting a 4x4 matrix
Client 12080: the matrix is invertible,total time 8.14622 seconds,goodbye
Client output Mon Apr 18 11:41:40 2022

Client PID# 12087 (data.csv) is submitting a 3x3 matrix
Client 12087: the matrix is invertible,total time 8.52859 seconds,goodbye
Client output Mon Apr 18 11:41:42 2022

Client PID# 12083 (data.csv) is submitting a 3x3 matrix
Client 12083: the matrix is invertible,total time 10.16387 seconds,goodbye
Client output Mon Apr 18 11:41:42 2022

Client PID# 12094 (data.csv) is submitting a 3x3 matrix
Client 12094: the matrix is invertible,total time 10.60296 seconds,goodbye
Client output Mon Apr 18 11:41:44 2022

Client PID# 12081 (data.csv) is submitting a 3x3 matrix
Client 12081: the matrix is invertible,total time 12.47639 seconds,goodbye
Client output Mon Apr 18 11:41:44 2022

Client PID# 12089 (data.csv) is submitting a 4x4 matrix
Client 12089: the matrix is invertible,total time 12.60305 seconds,goodbye
orkun@orkun:~/Desktop/son2$
```

Log File

```
arslan
(Mon Apr 18 11:41:29 2022) Server Y (arslan, p=2, t=2) started
(Mon Apr 18 11:41:29 2022) Instantiated server Z
(Mon Apr 18 11:41:29 2022) Z:Server Z (arslan, p=2, t=2) started
(Mon Apr 18 11:41:32 2022) Worker PID#12074 is handling client PID#12079 matrix size 3X3, pool busy 1/2
(Mon Apr 18 11:41:32 2022) Worker PID#12076 responding to client PID#12079: the matrix IS: invertible.
(Mon Apr 18 11:41:32 2022) Worker PID#12074 is handling client PID#12082 matrix size 5X5, pool busy 2/2
(Mon Apr 18 11:41:32 2022) Forwarding request of client PID#12090to serverZ, matrix size 3X3, pool busy 2/2
(Mon Apr 18 11:41:32 2022) Worker PID#12077 responding to client PID#12082: the matrix IS: NOT invertible.
(Mon Apr 18 11:41:32 2022) Z:Worker PID#12075 is handling client PID#12090 matrix size 3X3, pool busy 1/2
(Mon Apr 18 11:41:32 2022) Z:Worker PID#12075 is handling client PID#12091 matrix size 5X5, pool busy 2/2
(Mon Apr 18 11:41:32 2022) Forwarding request of client PID#12091to serverZ, matrix size 5X5, pool busy 2/2
(Mon Apr 18 11:41:32 2022) Forwarding request of client PID#12092to serverZ, matrix size 3X3, pool busy 2/2
(Mon Apr 18 11:41:32 2022) Z:Worker PID#12075 is handling client PID#12092 matrix size 3X3, pool busy 2/2
(Mon Apr 18 11:41:32 2022) Forwarding request of client PID#12093to serverZ, matrix size 4X4, pool busy 2/2
(Mon Apr 18 11:41:32 2022) Forwarding request of client PID#12084to serverZ, matrix size 4X4, pool busy 2/2
(Mon Apr 18 11:41:32 2022) Forwarding request of client PID#12085to serverZ, matrix size 3X3, pool busy 2/2
(Mon Apr 18 11:41:32 2022) Forwarding request of client PID#12080to serverZ, matrix size 4X4, pool busy 2/2
(Mon Apr 18 11:41:32 2022) Forwarding request of client PID#12087to serverZ, matrix size 3X3, pool busy 2/2
(Mon Apr 18 11:41:32 2022) Forwarding request of client PID#12083to serverZ, matrix size 3X3, pool busy 2/2
(Mon Apr 18 11:41:32 2022) Forwarding request of client PID#12094to serverZ, matrix size 3X3, pool busy 2/2
(Mon Apr 18 11:41:32 2022) Forwarding request of client PID#12081to serverZ, matrix size 3X3, pool busy 2/2
(Mon Apr 18 11:41:32 2022) Forwarding request of client PID#12089to serverZ, matrix size 4X4, pool busy 2/2
(Mon Apr 18 11:41:34 2022) Z:Worker PID#12088 responding to client PID#0: the matrix IS: NOT invertible.
(Mon Apr 18 11:41:34 2022) Z:Worker PID#12075 is handling client PID#12093 matrix size 4X4, pool busy 2/2
(Mon Apr 18 11:41:34 2022) Z:Worker PID#12086 responding to client PID#0: the matrix IS: NOT invertible.
(Mon Apr 18 11:41:34 2022) Z:Worker PID#12075 is handling client PID#12084 matrix size 4X4, pool busy 2/2
(Mon Apr 18 11:41:36 2022) Z:Worker PID#12088 responding to client PID#0: the matrix IS: invertible.
(Mon Apr 18 11:41:36 2022) Z:Worker PID#12075 is handling client PID#12085 matrix size 3X3, pool busy 2/2
(Mon Apr 18 11:41:36 2022) Z:Worker PID#12086 responding to client PID#0: the matrix IS: invertible.
(Mon Apr 18 11:41:36 2022) Z:Worker PID#12075 is handling client PID#12080 matrix size 4X4, pool busy 2/2
(Mon Apr 18 11:41:38 2022) Z:Worker PID#12088 responding to client PID#0: the matrix IS: invertible.
(Mon Apr 18 11:41:38 2022) Z:Worker PID#12075 is handling client PID#12087 matrix size 3X3, pool busy 2/2
(Mon Apr 18 11:41:38 2022) Z:Worker PID#12086 responding to client PID#0: the matrix IS: NOT invertible.
(Mon Apr 18 11:41:38 2022) Z:Worker PID#12075 is handling client PID#12083 matrix size 3X3, pool busy 2/2
(Mon Apr 18 11:41:40 2022) Z:Worker PID#12088 responding to client PID#0: the matrix IS: invertible.
(Mon Apr 18 11:41:40 2022) Z:Worker PID#12075 is handling client PID#12094 matrix size 3X3, pool busy 2/2
(Mon Apr 18 11:41:40 2022) Z:Worker PID#12086 responding to client PID#0: the matrix IS: invertible.
(Mon Apr 18 11:41:40 2022) Z:Worker PID#12075 is handling client PID#12081 matrix size 3X3, pool busy 2/2
(Mon Apr 18 11:41:42 2022) Z:Worker PID#12088 responding to client PID#0: the matrix IS: invertible.
(Mon Apr 18 11:41:42 2022) Z:Worker PID#12075 is handling client PID#12089 matrix size 4X4, pool busy 2/2
(Mon Apr 18 11:41:42 2022) Z:Worker PID#12086 responding to client PID#0: the matrix IS: NOT invertible.
(Mon Apr 18 11:41:44 2022) Z:Worker PID#12088 responding to client PID#0: the matrix IS: NOT invertible.
(Mon Apr 18 11:41:44 2022) Z:Worker PID#12086 responding to client PID#0: the matrix IS: invertible.
```

Ctrl+C terminate

```
) SIGINT received, exiting server Z. Total requests handled 15, 2 invertible, 13 , not invertible.
) Z:SIGINT received, exiting server Z. Total requests handled 13, 10 invertible, 3 , not invertible.
```