GTU

DEPARTMENT OF
COMPUTER ENGINEERING

CSE 344 – Spring 2022

MIDTERM REPORT

SÜLEYMAN GÖLBOL
1801042656

1. REQUIREMENTS

NONFUNCTIONAL REQUIREMENTS

1. Portability → The application should be portable. All computers that have Linux Distro and GCC compiler can run the program. Also, it can be run on Windows when Windows Subsystem for Linux 2 (v2 uses real Linux kernel) activated.

2. Maintainability of errors → In case of an error occurrence, the system uses perror and log file descriptor in order to give feedback on log file.

3. Performance → The system should initially be able to process as many entries as possible. Each request must be processed with different terminals. The system's performance should be fast enough to show user the feedback.

FUNCTIONAL REQUIREMENTS

In order to compile the program, user have to use "make" command that uses gcc. If make or gcc is not installed user can install it via *"sudo apt-get install build-essential"* command.
Make command runs 2 commands.

```
./client -s serverFifo.txt -o files/data.csv
./serverY -s serverFifo.txt -o logFile.txt -p 5 -r 4 -t 2
```

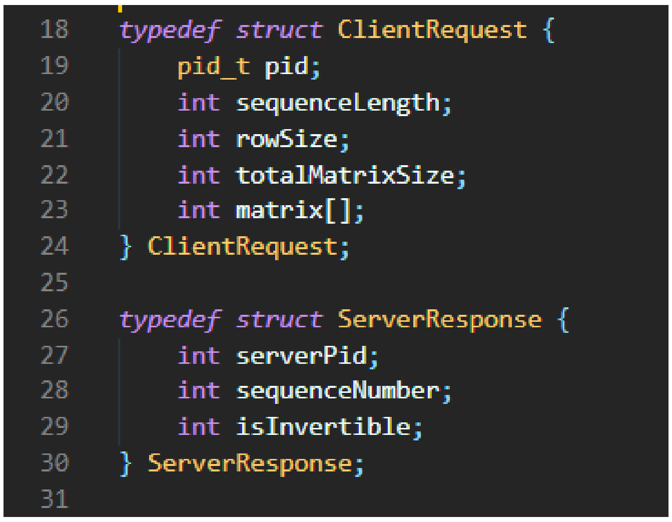
The reason that we are creating 2 executables is because of child – parent relation. In order to get the results we need to run executable like below with input and output paths as command line arguments.

```
./serverY -s serverFifo.txt -o logFile.txt -p 5 -r 4 -t 2
```

If we have all permissions in order to make fifo between server and client and we have permission to create/save the log file, the executable will run successfully.

2. PROBLEM SOLUTION APPROACH

In the midterm project, I encountered so many problems.
Firstly, the first big problem for me was to create fifo between client and server. To ensure the connection and information sending between them , I created 2 different structs.



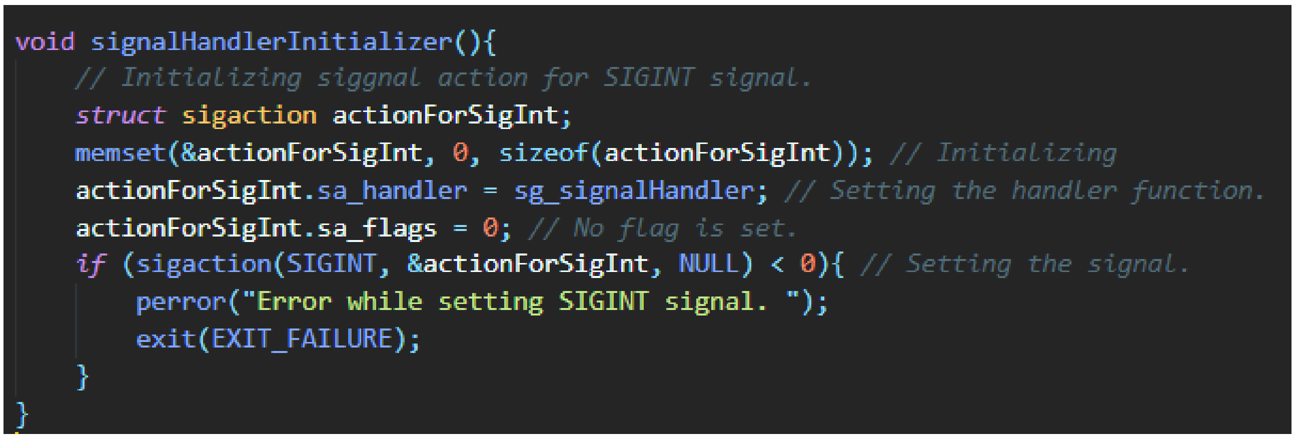
```
typedef struct ClientRequest {
    pid_t pid;
    int sequenceLength;
    int rowSize;
    int totalMatrixSize;
    int matrix[];
} ClientRequest;

typedef struct ServerResponse {
    int serverPid;
    int sequenceNumber;
    int isInvertible;
} ServerResponse;

```

ClientRequest holds details about the matrix that is readed and and at the end it holds a matrix[] array. The problem was, firstly I used integer pointer but sending the matrix failed. Server couldn't read the matrix. So, I used FAM which standardized from C99.

FAM means Flexible Array Member and in order to use it, in the struct there must be at least one other data member and the flexible array member must be last declared variable in the struct. (like matrix[] in the image.)

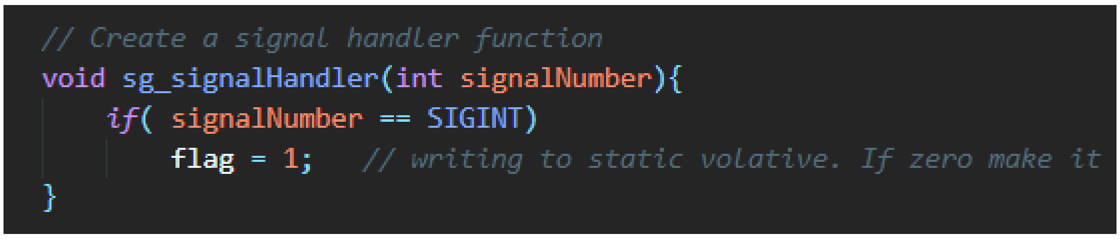
Another problem was printing timestamp without new line character so I manipulated last index of string that comes from time stamp's asctime(localtime()).

Another problem was signal handling. To get the signal I used it for checking SIGINT.



```c
void signalHandlerInitializer(){
    // Initializing siggnal action for SIGINT signal.
    struct sigaction actionForSigInt;
    memset(&actionForSigInt, 0, sizeof(actionForSigInt)); // Initializing
    actionForSigInt.sa_handler = sg_signalHandler; // Setting the handler function.
    actionForSigInt.sa_flags = 0; // No flag is set.
    if (sigaction(SIGINT, &actionForSigInt, NULL) < 0){ // Setting the signal.
        perror("Error while setting SIGINT signal. ");
        exit(EXIT_FAILURE);
    }
}
```

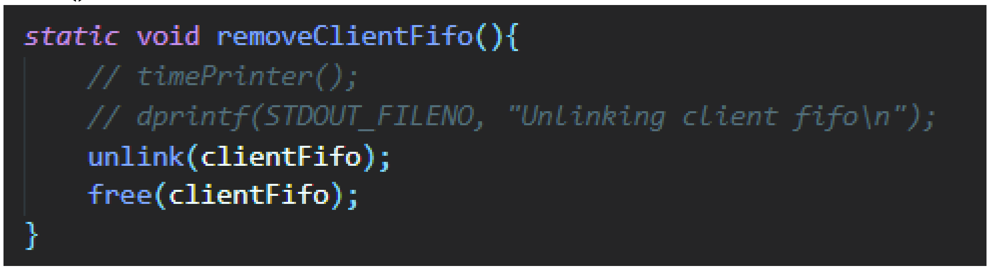
Inside signal handler I just changed the value of the flag which is static integer in order to access without parameter to it.



```c
// Create a signal handler function
void sg_signalHandler(int signalNumber){
    if( signalNumber == SIGINT)
        flag = 1;   // writing to static volative. If zero make it
}
```

Another problem I’ve encountered was to create a name for fifo. It should be unique so I needed to merge with pid. But pid is integer and to concatenate with path, I needed convert it to string. Because of itoa() function is not standard, I used my own itoa function which works for positive numbers.

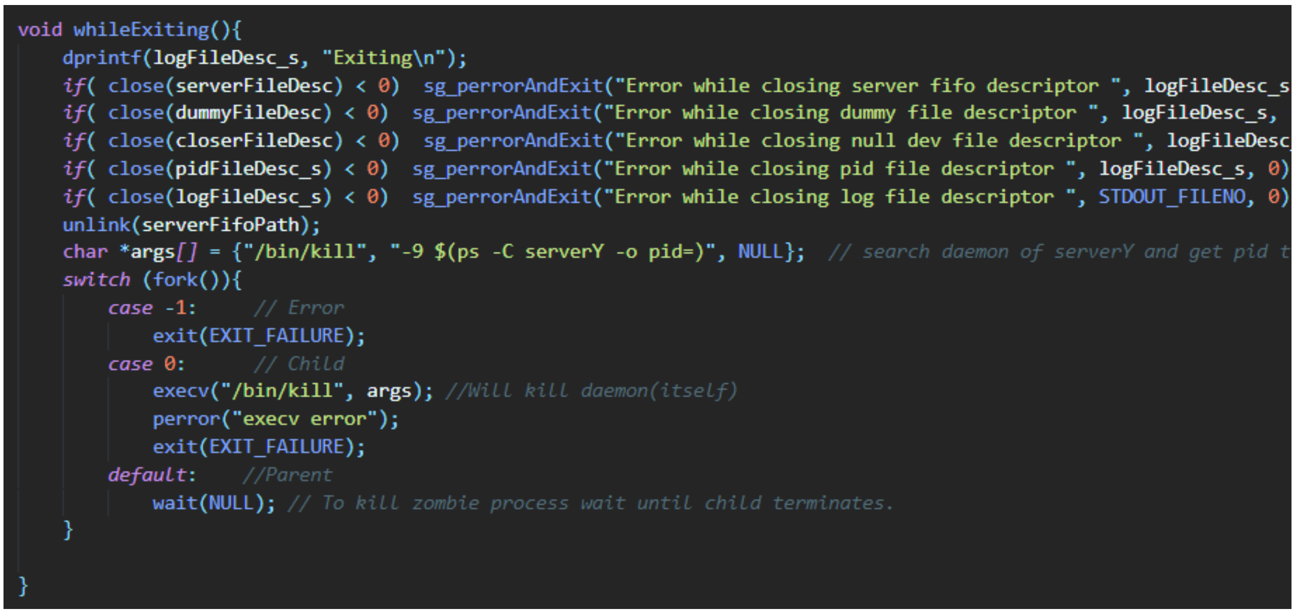
At the end of the program fifo must be unlinked so I used atexit to connect exit() to function to unlink the clientFifo.



```c
static void removeClientFifo(){
    // timePrinter();
    // dprintf(STDOUT_FILENO, "Unlinking client fifo\n");
    unlink(clientFifo);
    free(clientFifo);
}
```

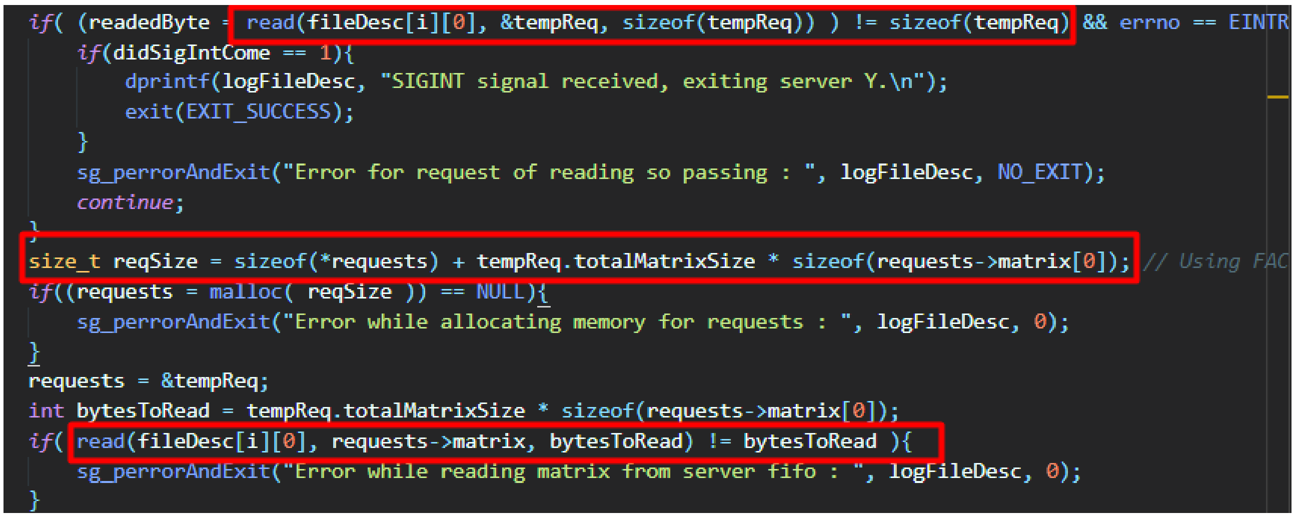
In serverY file, I had so much more problems and I couldn’t solve some of them.
First problem was if clientX gets the SIGINT signal, I needed to send this information to server. Because of serverY is daemon. I cannot directly use Ctrl C for it.

First, I used static int pointer variable and shared memory to solve it. I put this variable inside another header file which both client and server includes.



```c
void whileExiting(){
    dprintf(logFileDesc_s, "Exiting\n");
    if( close(serverFileDesc) < 0)  sg_perrorAndExit("Error while closing server fifo descriptor ", logFileDesc_s
    if( close(dummyFileDesc) < 0)  sg_perrorAndExit("Error while closing dummy file descriptor ", logFileDesc_s,
    if( close(closerFileDesc) < 0)  sg_perrorAndExit("Error while closing null dev file descriptor ", logFileDesc
    if( close(pidFileDesc_s) < 0)  sg_perrorAndExit("Error while closing pid file descriptor ", logFileDesc_s, 0)
    if( close(logFileDesc_s) < 0)  sg_perrorAndExit("Error while closing log file descriptor ", STDOUT_FILENO, 0)
    unlink(serverFifoPath);
    char *args[] = {"/bin/kill", "-9 $(ps -C serverY -o pid=)", NULL};  // search daemon of serverY and get pid t
    switch (fork()){
        case -1:      // Error
            exit(EXIT_FAILURE);
        case 0:       // Child
            execv("/bin/kill", args); //Will kill daemon(itself)
            perror("execv error");
            exit(EXIT_FAILURE);
        default:     //Parent
            wait(NULL); // To kill zombie process wait until child terminates.
    }

}
```

If signal flag was worked, I already had the exiting daemon function. I used “ps -C serverY -o pid=” command to list pid of server daemon, and inside execv, I called kill command to kill the daemon with pid.



```c
if( (readedByte = read(fileDesc[i][0], &tempReq, sizeof(tempReq)) ) != sizeof(tempReq) && errno == EINTR
    if(didSigIntCome == 1){
        dprintf(logFileDesc, "SIGINT signal received, exiting server Y.\n");
        exit(EXIT_SUCCESS);
    }
    sg_perrorAndExit("Error for request of reading so passing : ", logFileDesc, NO_EXIT);
    continue;
}
size_t reqSize = sizeof(*requests) + tempReq.totalMatrixSize * sizeof(requests->matrix[0]); // Using FAC
if((requests = malloc( reqSize )) == NULL){
    sg_perrorAndExit("Error while allocating memory for requests : ", logFileDesc, 0);
}
requests = &tempReq;
int bytesToRead = tempReq.totalMatrixSize * sizeof(requests->matrix[0]);
if( read(fileDesc[i][0], requests->matrix, bytesToRead) != bytesToRead ){
    sg_perrorAndExit("Error while reading matrix from server fifo : ", logFileDesc, 0);
}
```

While reading request coming from client inside server, to use FAM (flexible array member) I read 2 times. First I read the request without the matrix[], then I put the matrix size inside reqSize, then I created a ClientRequest* pointer with size 1 to read the all of matrix.

Another problem I’ve encountered was printing (or saving to log file) the integers (non strings).

I used system calls to create file descriptor for logfile. But I couldn’t use this descriptor with fprintf because that needs a standard C FILE*. So I used dprintf().

dprintf() comes since glibc 2.10 and it’s totally POSIX compliant (_POSIX_C_SOURCE >= 200809L) .

It’s first argument is file descriptor and second argument is buffer and it’s still variadic function so I passed integers using %d.

For the communication between serverY parent and children, I used unidirectional pipe. I used pipe() method to get the file descriptors. Then I closed write end of child and read end of parent so the pipe was successful. When the request comes to parent, it reads and sends to child in order to handle and write the response.

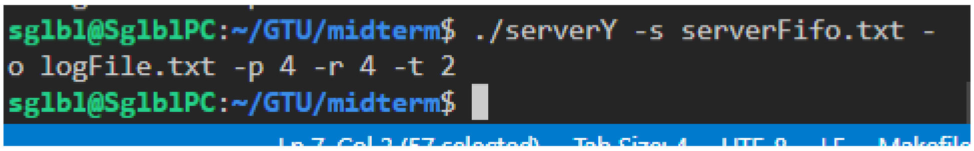
If user tries to open server in other terminal, because of serverFifo, it doesn’t let multiple instances.

MISSING PARTS

I encountered problems while using file locks to obtain sync of logFile between serverY and serverZ. So I didn’t use file lock for log file.

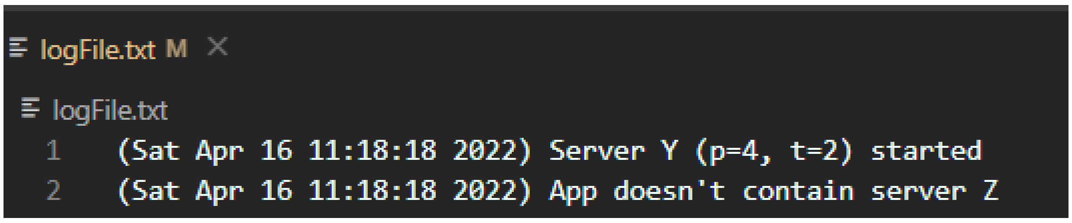
3) TEST CASES AND RESULTS

```
./serverY -s serverFifo.txt -o logFile.txt -p 5 -r 4 -t 2
```



```
sglbl@SglblPC:~/GTU/midterm$ ./serverY -s serverFifo.txt -
o logFile.txt -p 4 -r 4 -t 2
sglbl@SglblPC:~/GTU/midterm$
```

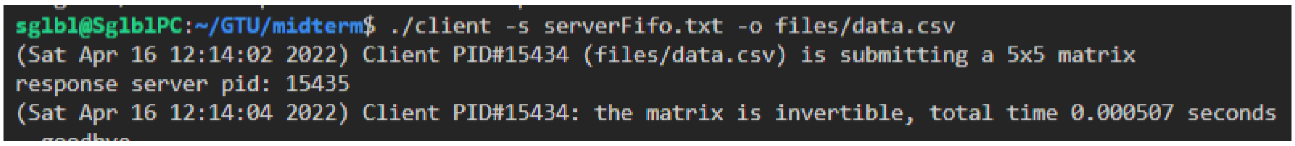
Because of daemon, disconnects from terminal.



```
(Sat Apr 16 11:18:18 2022) Server Y (p=4, t=2) started
(Sat Apr 16 11:18:18 2022) App doesn't contain server Z
```

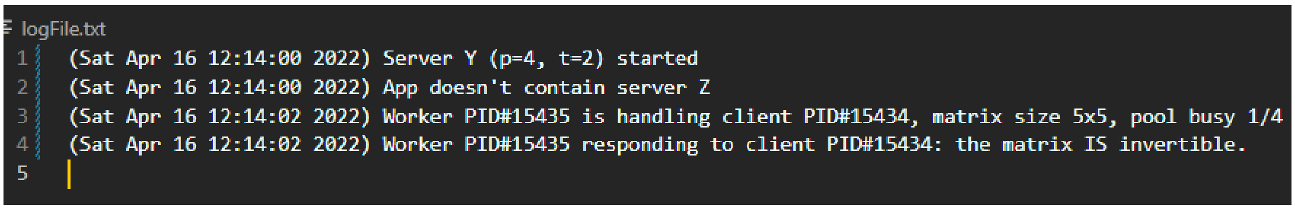
In the log file it shows the output.

```
./client -s serverFifo.txt -o files/data.csv
```



```
sglbl@SglblPC:~/GTU/midterm$ ./client -s serverFifo.txt -o files/data.csv
(Sat Apr 16 12:14:02 2022) Client PID#15434 (files/data.csv) is submitting a 5x5 matrix
response server pid: 15435
(Sat Apr 16 12:14:04 2022) Client PID#15434: the matrix is invertible, total time 0.000507 seconds
```

Client runs sends matrix and returns back the information.

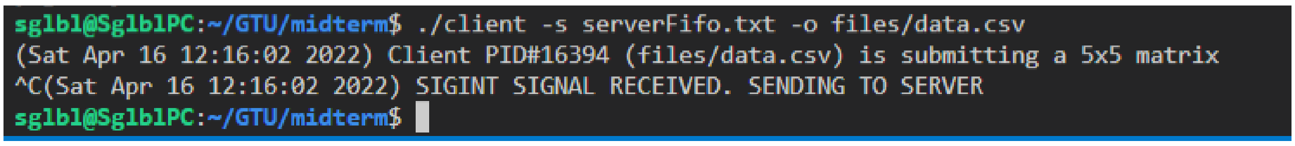


```
logFile.txt
(Sat Apr 16 12:14:00 2022) Server Y (p=4, t=2) started
(Sat Apr 16 12:14:00 2022) App doesn't contain server Z
(Sat Apr 16 12:14:02 2022) Worker PID#15435 is handling client PID#15434, matrix size 5x5, pool busy 1/4
(Sat Apr 16 12:14:02 2022) Worker PID#15435 responding to client PID#15434: the matrix IS invertible.
```

ServerY also writes the worker/client and matrix information to log file.

3.c) **Interrupting with Signals**

For handling SIGINT signal I used "`static volatile sig_atomic_t`" flag

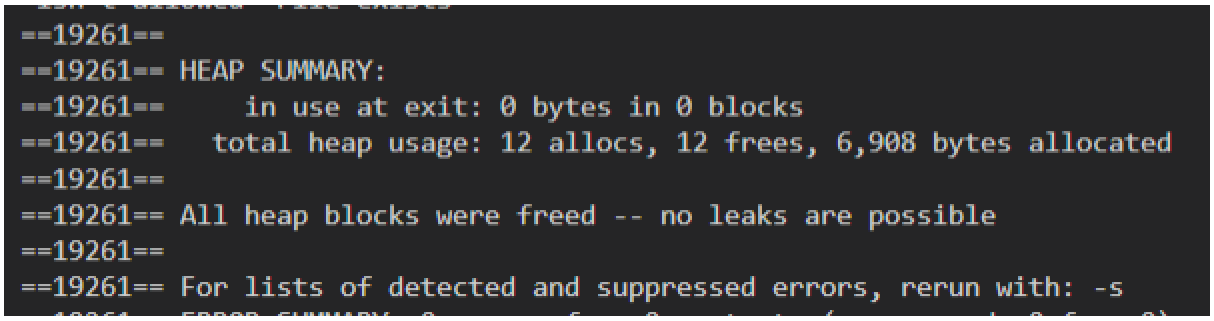


```
sglbl@SglblPC:~/GTU/midterm$ ./client -s serverFifo.txt -o files/data.csv
(Sat Apr 16 12:16:02 2022) Client PID#16394 (files/data.csv) is submitting a 5x5 matrix
^C(Sat Apr 16 12:16:02 2022) SIGINT SIGNAL RECEIVED. SENDING TO SERVER
sglbl@SglblPC:~/GTU/midterm$
```

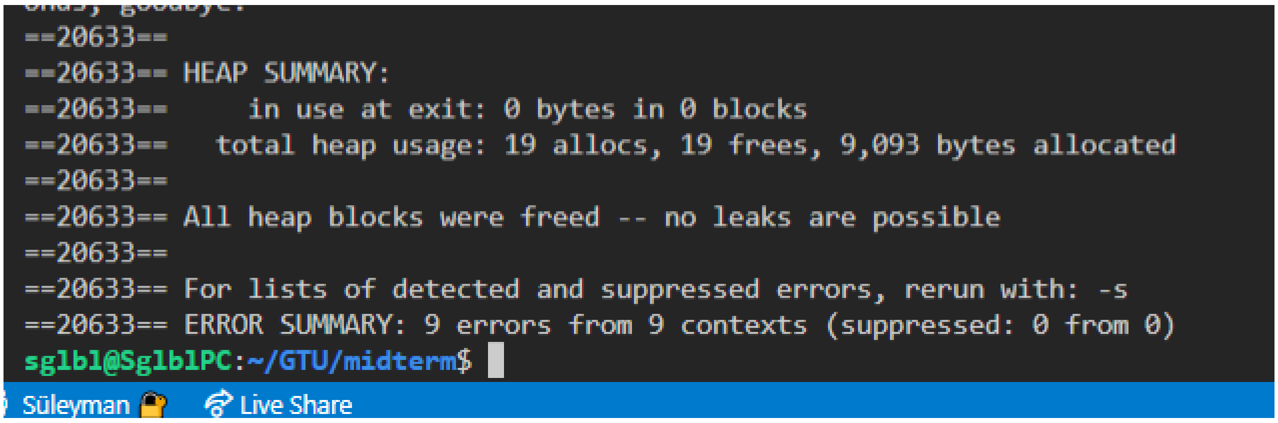
Client gets signal and gracefully exits with the method registered on atexit but serverY doesn't get the signal.

VALGRIND MEMORY RESULTS

The output from valgrind about heap and leaks is like below:

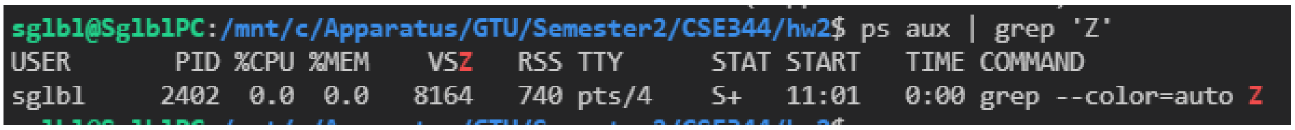


```
==19261==
==19261== HEAP SUMMARY:
==19261==     in use at exit: 0 bytes in 0 blocks
==19261==   total heap usage: 12 allocs, 12 frees, 6,908 bytes allocated
==19261==
==19261== All heap blocks were freed -- no leaks are possible
==19261==
==19261== For lists of detected and suppressed errors, rerun with: -s
```



```
==20633==
==20633== HEAP SUMMARY:
==20633==     in use at exit: 0 bytes in 0 blocks
==20633==   total heap usage: 19 allocs, 19 frees, 9,093 bytes allocated
==20633==
==20633== All heap blocks were freed -- no leaks are possible
==20633==
==20633== For lists of detected and suppressed errors, rerun with: -s
==20633== ERROR SUMMARY: 9 errors from 9 contexts (suppressed: 0 from 0)
sglbl@SglblPC:~/GTU/midterm$
```

CHECKING FOR ZOMBIE PROCESSES



```
sglbl@SglblPC:/mnt/c/Apparatus/GTU/Semester2/CSE344/hw2$ ps aux | grep 'Z'
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
sglbl       2402  0.0  0.0   8164   740 pts/4    S+   11:01   0:00 grep --color=auto Z
```