CSE344
System Programming

Midterm Project

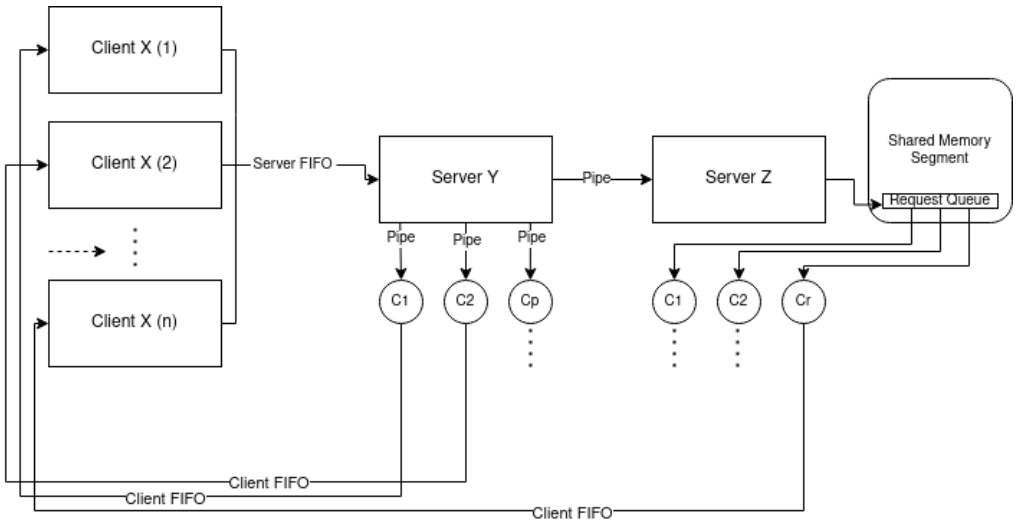
Author
Gökbey Gazi KESKİN
Date
April 16, 2022

Table of Contents

Requirement Analysis....3
System Diagram....3
Requirements....3
Requirement 1: Each Client should have its distinct Client FIFO and name of the FIFO should be propagated to the children of servers....3
Requirement2: When Server Z receives SIGINT, all its children, ServerZ and ServerZ's children should clear their resources and exit....3
Requirement3:There should be only 1 running instance of processes Y and Z....3
Requirement4: POSIX does not provide synchronization on shared memory. Synchronization and mutual exclusion should be done manually....4
Requirement 5: Server Y should know if their children are available....4
Requirement 6: ServerY and ServerZ should know how many requests were invertible....4
Requirement 7: ServerY and ServerZ doesn't have a controlling terminal and they should output to the log file....4
Requirement 8: There should be a precaution against zombie childrens....5
Requirement 9: Processes shouldn't write to log file at the same time....5
Requirement 10: Signals shouldn't interrupt Operations……..…….5
Step-by-Step Algorithm....5
Tests Cases & Results....6
Default flow of tests:....6
Test1....6
Test2....6
Test3....7
Test4....7
Test5....8

Requirement Analysis

System Diagram



Requirements

Requirement 1: Each Client should have its distinct Client FIFO and name of the FIFO should be propagated to the children of servers.

Solution: Each client creates it's Client FIFO and names it as its PID. This name is sent to servers and servers send this name to their children. This way, children knows where to send the result and also it is used for log output:

```
Usage1:                                  Usage2:
myPrint(2," is handling client PID#");   clientFD = open(clientFifo, O_WRONLY);
MyPrint(0,clientFIFO);
```

Requirement2: When Server Z receives SIGINT, all its children, ServerZ and ServerZ's children should clear their resources and exit.

Solution: When SIGINT is received by ServerY, it's Signal Handler sets the value sigint_received to 1. Then, in main function, before freeing other resources, closes the write end of the children pipes (so children while-read loops breaks and they free their resources and return) and forwards the SIGINT to ServerZ:

```
for(int i=0;i<poolSize;i++)          Than, ServerZ forwards this message to its
      kill(serverZPID,SIGINT);       children and exits:
                                     for(int i=0;i<poolSize;i++)
                                           kill(childrenPid[i],SIGINT);
```

Lastly, children of ServerZ break their main loop when SIGINT is received:

```
if(sigint_received)
    break;
free their resources and return.
```

Requirement3: There should be only 1 running instance of processes Y and Z.

Solution: On instantiation, ServerY creates a uniquely named file and removes it when SIGINT is received. If the file already exists, ServerY prints an error message and exits (ServerZ never gets created).

Requirement4: POSIX does not provide synchronization on shared memory. Synchronization and mutual exclusion should be done manually.

Solution: Used a binary semaphore for mutual exclusion and 2 other semaphores for synchronization.

```
Child of Z:                          Z:
if(sem_wait(full_sem)==-1){          if(sem_wait(empty_sem)==-1){
      if(sigint_received)                  perror("wait_sem_empty");
            break;                         exit(EXIT_FAILURE);
      perror("wait_sem_full");       }
                                     if(sem_wait(mutex_sem)==-1){
}                                          perror(“wait_sem_empty”);
if(sem_wait(mutex_sem)==-1){               exit(EXIT_FAILURE);
      perror("wait_sem_mutex");      };
      exit(EXIT_FAILURE);            requestsHandled+=1;
}                                    enqueue(shared_mem_ptr,buffer);
                                     if(sem_post(mutex_sem)==-1){
shared_mem_ptr→busyChild+=1;               perror(“post_sem_mutex”);
request = dequeue(shared_mem_ptr);         exit(EXIT_FAILURE);
strcpy(buffer,request);              }
free(request);                       if(sem_post(full_sem)==-1){
                                           perror(“post_sem_mutex”);
if(sem_post(mutex_sem)==-1){               exit(EXIT_FAILURE);
      perror("sem_post");            }
      exit(EXIT_FAILURE);
}
if(sem_post(empty_sem)==-1){
      perror("sem_post");
      exit(EXIT_FAILURE);
}
```

Requirement 5: Server Y should know if their children are available.

Solution : Server Y has a struct child array called children. Struct child has a variable busy. When a request is sent to a child, its busy field is set to 1 and when a child sends SIGUSR1, it is set back to 0. Used a different signal handler for this task which has a info field.

```
void siguserHandler(int sig, siginfo_t *info, void *context){
      childReturned=1;
      availableChild=info->si_pid;
}
```

Requirement 6: ServerY and ServerZ should know how many requests were invertible

Solution: Children sends SIGUSR2 every time the matrix is invertible. (non-invertible amount = handledRequests-invertible)

Requirement 7: ServerY and ServerZ doesn’t have a controlling terminal and they should output to the log file.

Solution: In becomeDaemon function of ServerY STDOUT and STDERR file descriptors are replaced with log file descriptor and since ServerZ is a child of the ServerY, it inherits the descriptors.

```
dup2(logFD,STDOUT_FILENO);
dup2(logFD,STDERR_FILENO);
close(logFD);
close(0);
for (fd = 3; fd < maxfd; fd++)
      close(fd);
```

Requirement 8: There should be a precaution against zombie childrens.

Solution: Implemented SIGCHLD handler and processes wait everytime they receive it.

```
if(sigchld_received){
      wait(NULL);
}
```

Requirement 9: Processes shouldn't write to log file at the same time.

Solution: Used fcntl function to lock the file.

Requirement 10: Signals shouldn't interrupt Operations
Solution: ClientX, ServerY and ServerZ blocks all the signals except the ones they use (SIGINT, SIGUSR, SIGCHLD etc.) because they have their own handlers.

```
if(sigfillset(&mask)==-1 || sigdelset(&mask,SIGINT)==-1  || sigdelset(&mask,SIGCHLD)==-1 ||
        sigdelset(&mask, SIGUSR1)==-1 || sigdelset(&mask, SIGUSR2)==-1 || sigprocmask(SIG_SETMASK,
&mask, NULL)==-1){
        perror("Failed to block signals");
        exit(EXIT_FAILURE);
  }
```

Step-by-Step Algorithm

General Perspective:
1) Server Y creates a server FIFO and opens it in readonly mode
2) ServerY creates its children and ServerZ. ServerZ creates a pipe for each of its children (including ServerZ), closes its reading end and sends it to children (as a function variable for children, as an argv element to ServerZ)
3) ServerZ creates its children and a shared memory segment with a request queue
4) Children of ServerZ maps to shared memory segment
5) Each ClientX opens the server FIFO in writeonly mode
6) Each ClientX creates a client FIFO named as its PID followed by character and sends the name to ServerY through Server FIFO
7) Each Client reads it's data file and sends it to ServerY through Server FIFO

ServerY Perspective:
8) If ServerY has any available children, it sends the name of the client and the matrix to its available child as a package through its pipe, otherwise, it sends the same package to ServerZ.
9) Children of ServerY unpacks the data, opens the client fifo, checks if the given matrix is invertible and sends the result through client FIFO. During this operation, it also notifies the ServerY (SIGUSR2) when the matrix is invertible.

ServerZ Perspective:
10) ServerZ enqueues the package to the requests queue (which is in shared memory segment)
11) Children of Z dequeues packages from requests queue whenever they are available and does the same job with children of ServerY after this point.

General Perspective:
12) Each ClientX gets the result, prints it and returns.
13) ServerY and ServerZ continues waiting for input until they receive SIGINT. When ServerY receives SIGINT, forwards the signal to all its children (including ServerZ), frees its resources and exits. ServerZ forwards the SIGINT to its children, frees it resources and exits. All the children frees their resources and exits as well.

Tests Cases & Results

Default flow of tests:

Start serverY as daemon process:

```
gokbey@gokbey-ABRA-A5-V15-3:~/Desktop/system_midterm$ ./serverY -s /tmp/SVFIFO -o log -p 2 -r 2 -t 4
gokbey@gokbey-ABRA-A5-V15-3:~/Desktop/system_midterm$
```

Start n clients (specified at test case)with different matrices and wait for output:

```
gokbey@gokbey-ABRA-A5-V15-3:~/Desktop/system_midterm$ ./clientX -s /tmp/SVFIFO -o matrix.csv
Fri Apr 15 18:18:22 2022|Client PID#36428:(matrix.csv) is submitting a 3x3 matrix
Fri Apr 15 18:18:26 2022|Client PID#36428: the matrix is invertible, total time 4.002003, goodbye.
gokbey@gokbey-ABRA-A5-V15-3:~/Desktop/system_midterm$
```

Send SIGINT to serverY

```
gokbey@gokbey-ABRA-A5-V15-3:~/Desktop/system_midterm$ kill -2 36344
```

Test1

Only ServerY is used (sent #poolSize requests at the same time)
2 clients sent 2 requests at the same time.
PoolSize:2, SleepTime:4

Log File:

```
Y:Sat Apr 16 00:24:52 2022|Server Y (log, p=2, t=2) started
Y:Sat Apr 16 00:24:55 2022|Instantiated server Z
Y:Sat Apr 16 00:24:55 2022|pool busy, 1/2, Worker PID#20626 is handling client PID#20624, matrix size 4x4
Y:Sat Apr 16 00:24:55 2022|pool busy, 2/2, Worker PID#20627 is handling client PID#20631, matrix size 3x3
Y:Sat Apr 16 00:25:02 2022|SIGINT Received, terminating Z and exiting server Y Total requests handled: 2, 1 invertible, 1 not. 0 requests were forwarded.
Z:Sat Apr 16 00:25:02 2022|SIGINT Received, exiting server Z. Total requests handled 0, 0 invertible, 0 not.
```

Each ClientX took ~4 seconds

Test2

Only ServerY is used (sent #poolSize requests at the same time).
30 clients sent 30 requests with random time intervals.
PoolSize:4, SleepTime:2

Log File:

```
Y:Sat Apr 16 00:25:48 2022|Server Y (log, p=4, t=2) started
Y:Sat Apr 16 00:25:56 2022|Instantiated server Z
Y:Sat Apr 16 00:25:56 2022|pool busy, 1/4, Worker PID#20798 is handling client PID#20792, matrix size 3x3
Y:Sat Apr 16 00:25:56 2022|pool busy, 2/4, Worker PID#20799 is handling client PID#20802, matrix size 4x4
Y:Sat Apr 16 00:25:56 2022|pool busy, 3/4, Worker PID#20800 is handling client PID#20804, matrix size 3x3
Y:Sat Apr 16 00:25:57 2022|pool busy, 4/4, Worker PID#20801 is handling client PID#20806, matrix size 4x4
Y:Sat Apr 16 00:25:58 2022|pool busy, 3/4, Worker PID#20798 is handling client PID#20810, matrix size 3x3
Y:Sat Apr 16 00:26:00 2022|pool busy, 2/4, Worker PID#20799 is handling client PID#20815, matrix size 4x4
Y:Sat Apr 16 00:26:01 2022|pool busy, 2/4, Worker PID#20798 is handling client PID#20819, matrix size 3x3
Y:Sat Apr 16 00:26:02 2022|pool busy, 3/4, Worker PID#20800 is handling client PID#20822, matrix size 4x4
Y:Sat Apr 16 00:26:05 2022|pool busy, 1/4, Worker PID#20798 is handling client PID#20851, matrix size 3x3
Y:Sat Apr 16 00:26:06 2022|pool busy, 2/4, Worker PID#20799 is handling client PID#20854, matrix size 4x4
Y:Sat Apr 16 00:26:08 2022|pool busy, 2/4, Worker PID#20798 is handling client PID#20858, matrix size 3x3
Y:Sat Apr 16 00:26:08 2022|pool busy, 2/4, Worker PID#20799 is handling client PID#20862, matrix size 4x4
Y:Sat Apr 16 00:26:10 2022|pool busy, 2/4, Worker PID#20798 is handling client PID#20864, matrix size 3x3
Y:Sat Apr 16 00:26:11 2022|pool busy, 2/4, Worker PID#20799 is handling client PID#20868, matrix size 4x4
Y:Sat Apr 16 00:26:11 2022|pool busy, 3/4, Worker PID#20800 is handling client PID#20870, matrix size 4x4
Y:Sat Apr 16 00:26:13 2022|pool busy, 2/4, Worker PID#20798 is handling client PID#20875, matrix size 3x3
Y:Sat Apr 16 00:26:14 2022|pool busy, 2/4, Worker PID#20799 is handling client PID#20879, matrix size 4x4
Y:Sat Apr 16 00:26:15 2022|pool busy, 2/4, Worker PID#20798 is handling client PID#20884, matrix size 3x3
Y:Sat Apr 16 00:26:16 2022|pool busy, 2/4, Worker PID#20799 is handling client PID#20888, matrix size 4x4
Y:Sat Apr 16 00:26:17 2022|pool busy, 3/4, Worker PID#20800 is handling client PID#20891, matrix size 3x3
Y:Sat Apr 16 00:26:18 2022|pool busy, 3/4, Worker PID#20798 is handling client PID#20895, matrix size 4x4
Y:Sat Apr 16 00:26:19 2022|pool busy, 3/4, Worker PID#20799 is handling client PID#20899, matrix size 3x3
Y:Sat Apr 16 00:26:23 2022|pool busy, 1/4, Worker PID#20798 is handling client PID#20925, matrix size 3x3
Y:Sat Apr 16 00:26:23 2022|pool busy, 2/4, Worker PID#20799 is handling client PID#20933, matrix size 4x4
Y:Sat Apr 16 00:26:23 2022|pool busy, 3/4, Worker PID#20800 is handling client PID#20935, matrix size 3x3
Y:Sat Apr 16 00:26:24 2022|pool busy, 4/4, Worker PID#20801 is handling client PID#20961, matrix size 4x4
Y:Sat Apr 16 00:26:28 2022|pool busy, 1/4, Worker PID#20798 is handling client PID#20970, matrix size 4x4
Y:Sat Apr 16 00:26:29 2022|pool busy, 2/4, Worker PID#20799 is handling client PID#20973, matrix size 3x3
Y:Sat Apr 16 00:26:29 2022|pool busy, 3/4, Worker PID#20800 is handling client PID#20976, matrix size 4x4
Y:Sat Apr 16 00:26:30 2022|pool busy, 4/4, Worker PID#20801 is handling client PID#20978, matrix size 3x3
Y:Sat Apr 16 00:26:47 2022|SIGINT Received, terminating Z and exiting server Y Total requests handled: 30, 22 invertible, 8 not. 0 requests were forwarded.
Z:Sat Apr 16 00:26:47 2022|SIGINT Received, exiting server Z. Total requests handled 0, 0 invertible, 0 not.
```

Each ClientX took ~4 seconds

Test3

Both ServerY and ServerZ are used
PoolSize:2, PoolSize2:3, SleepTime:4
6 clients sent 6 requests at the same time

Log File:

```
Y:Sat Apr 16 00:28:34 2022|Server Y (log, p=2, t=3) started
Y:Sat Apr 16 00:28:37 2022|Instantiated server Z
Y:Sat Apr 16 00:28:37 2022|pool busy, 1/2, Worker PID#21209 is handling client PID#21207, matrix size 3x3
Y:Sat Apr 16 00:28:38 2022|pool busy, 2/2, Worker PID#21210 is handling client PID#21216, matrix size 3x3
Y:Sat Apr 16 00:28:38 2022|pool busy, 2/2, Forwarding request of client PID#21218 to serverZ, matrix size 4x4
Z:Sat Apr 16 00:28:38 2022|pool busy, 1/3, Worker PID#21211is handling client #PID21218, matrix size 4x4
Y:Sat Apr 16 00:28:38 2022|pool busy, 2/2, Forwarding request of client PID#21220 to serverZ, matrix size 3x3
Z:Sat Apr 16 00:28:38 2022|pool busy, 2/3, Worker PID#21212is handling client #PID21220, matrix size 3x3
Y:Sat Apr 16 00:28:39 2022|pool busy, 2/2, Forwarding request of client PID#21222 to serverZ, matrix size 4x4
Z:Sat Apr 16 00:28:39 2022|pool busy, 3/3, Worker PID#21213is handling client #PID21222, matrix size 4x4
Y:Sat Apr 16 00:28:39 2022|pool busy, 2/2, Forwarding request of client PID#21224 to serverZ, matrix size 3x3
Z:Sat Apr 16 00:28:42 2022|pool busy, 3/3, Worker PID#21211is handling client #PID21224, matrix size 3x3
Y:Sat Apr 16 00:28:56 2022|SIGINT Received, terminating Z and exiting server Y Total requests handled: 2, 0 invertible, 2 not. 4 requests were forwarded.
Z:Sat Apr 16 00:28:56 2022|SIGINT Received, exiting server Z. Total requests handled 4, 3 invertible, 1 not.
```

Important Result: ServerZ doesn't decline requests, it enqueues them to the queue and workers dequeues them when they are available. So, first 3 requests to ServerZ took ~4 seconds but the 4th one took 6 second as expected (since it is not processed immediately and waited in the queue until a worker is ready).
3rd request which is sent to serverZ:

```
gokbey@gokbey-ABRA-A5-V15-3:~/Desktop/system_midterm$ ./clientX -s /tmp/SVFIFO -o matrix2.csv
Sat Apr 16 00:16:41 2022|Client PID#19656:(matrix2.csv) is submitting a 3x3 matrix
Sat Apr 16 00:16:45 2022|Client PID#19656: the matrix is not invertible, total time 4.001381, goodbye.
```

4th request which is sent to serverZ:

```
gokbey@gokbey-ABRA-A5-V15-3:~/Desktop/system_midterm$ ./clientX -s /tmp/SVFIFO -o matrix2.csv
Sat Apr 16 00:16:42 2022|Client PID#19658:(matrix2.csv) is submitting a 3x3 matrix
Sat Apr 16 00:16:49 2022|Client PID#19658: the matrix is not invertible, total time 6.788201, goodbye.
```

Test4

Both ServerY and ServerZ are used
PoolSize:5, PoolSize2:3, SleepTime:5
10 clients sent 10 requests at the same time

Log File:

```
Y:Sat Apr 16 00:34:18 2022|Server Y (log, p=5, t=3) started
Y:Sat Apr 16 00:35:08 2022|Instantiated server Z
Y:Sat Apr 16 00:35:08 2022|pool busy, 1/5, Worker PID#21946 is handling client PID#21944, matrix size 4x4
Y:Sat Apr 16 00:35:08 2022|pool busy, 2/5, Worker PID#21947 is handling client PID#21966, matrix size 3x3
Y:Sat Apr 16 00:35:08 2022|pool busy, 3/5, Worker PID#21948 is handling client PID#21968, matrix size 3x3
Y:Sat Apr 16 00:35:09 2022|pool busy, 4/5, Worker PID#21949 is handling client PID#21970, matrix size 3x3
Y:Sat Apr 16 00:35:09 2022|pool busy, 5/5, Worker PID#21950 is handling client PID#21972, matrix size 3x3
Y:Sat Apr 16 00:35:10 2022|pool busy, 5/5, Forwarding request of client PID#21974 to serverZ, matrix size 4x4
Z:Sat Apr 16 00:35:10 2022|pool busy, 1/3, Worker PID#21952is handling client #PID21974, matrix size 4x4
Y:Sat Apr 16 00:35:10 2022|pool busy, 5/5, Forwarding request of client PID#21976 to serverZ, matrix size 4x4
Z:Sat Apr 16 00:35:10 2022|pool busy, 2/3, Worker PID#21953is handling client #PID21976, matrix size 4x4
Y:Sat Apr 16 00:35:10 2022|pool busy, 5/5, Forwarding request of client PID#21978 to serverZ, matrix size 3x3
Z:Sat Apr 16 00:35:10 2022|pool busy, 3/3, Worker PID#21954is handling client #PID21978, matrix size 3x3
Y:Sat Apr 16 00:35:11 2022|pool busy, 5/5, Forwarding request of client PID#21980 to serverZ, matrix size 3x3
Y:Sat Apr 16 00:35:11 2022|pool busy, 5/5, Forwarding request of client PID#21982 to serverZ, matrix size 3x3
Z:Sat Apr 16 00:35:15 2022|pool busy, 3/3, Worker PID#21952is handling client #PID21980, matrix size 3x3
Z:Sat Apr 16 00:35:15 2022|pool busy, 3/3, Worker PID#21953is handling client #PID21982, matrix size 3x3
Y:Sat Apr 16 00:35:48 2022|SIGINT Received, terminating Z and exiting server Y Total requests handled: 5, 2 invertible, 3 not. 5 requests were forwarded.
Z:Sat Apr 16 00:35:48 2022|SIGINT Received, exiting server Z. Total requests handled 5, 4 invertible, 1 not.
```

As in the previous test, last 2 requests took more time than other 3 since they waited at the queue until 2 other worker become available.

Test5

Both ServerY and ServerZ are used
PoolSize:3, PoolSize2:3, SleepTime:4
16 clients sent 16 requests at different time intervals.

Log File:

```
Y:Sat Apr 16 00:42:26 2022|Server Y (log, p=3, t=3) started
Y:Sat Apr 16 00:42:39 2022|Instantiated server Z
Y:Sat Apr 16 00:42:39 2022|pool busy, 1/3, Worker PID#22735 is handling client PID#22733, matrix size 3x3
Y:Sat Apr 16 00:42:39 2022|pool busy, 2/3, Worker PID#22736 is handling client PID#22743, matrix size 3x3
Y:Sat Apr 16 00:42:40 2022|pool busy, 3/3, Worker PID#22737 is handling client PID#22745, matrix size 4x4
Y:Sat Apr 16 00:42:40 2022|pool busy, 3/3, Forwarding request of client PID#22748 to serverZ, matrix size 4x4
Z:Sat Apr 16 00:42:40 2022|pool busy, 1/3, Worker PID#22738is handling client #PID22748, matrix size 4x4
Y:Sat Apr 16 00:42:40 2022|pool busy, 3/3, Forwarding request of client PID#22750 to serverZ, matrix size 3x3
Z:Sat Apr 16 00:42:40 2022|pool busy, 2/3, Worker PID#22739is handling client #PID22750, matrix size 3x3
Y:Sat Apr 16 00:42:41 2022|pool busy, 3/3, Forwarding request of client PID#22752 to serverZ, matrix size 3x3
Z:Sat Apr 16 00:42:41 2022|pool busy, 3/3, Worker PID#22740is handling client #PID22752, matrix size 3x3
Y:Sat Apr 16 00:42:41 2022|pool busy, 3/3, Forwarding request of client PID#22754 to serverZ, matrix size 3x3
Z:Sat Apr 16 00:42:44 2022|pool busy, 3/3, Worker PID#22738is handling client #PID22754, matrix size 3x3
Y:Sat Apr 16 00:42:45 2022|pool busy, 1/3, Worker PID#22735 is handling client PID#22763, matrix size 3x3
Y:Sat Apr 16 00:42:46 2022|pool busy, 2/3, Worker PID#22736 is handling client PID#22766, matrix size 3x3
Y:Sat Apr 16 00:42:47 2022|pool busy, 3/3, Worker PID#22737 is handling client PID#22769, matrix size 4x4
Y:Sat Apr 16 00:42:54 2022|pool busy, 1/3, Worker PID#22735 is handling client PID#22797, matrix size 3x3
Y:Sat Apr 16 00:42:55 2022|pool busy, 2/3, Worker PID#22736 is handling client PID#22800, matrix size 3x3
Y:Sat Apr 16 00:43:02 2022|pool busy, 1/3, Worker PID#22735 is handling client PID#22808, matrix size 3x3
Y:Sat Apr 16 00:43:02 2022|pool busy, 2/3, Worker PID#22736 is handling client PID#22811, matrix size 3x3
Y:Sat Apr 16 00:43:03 2022|pool busy, 3/3, Worker PID#22737 is handling client PID#22813, matrix size 3x3
Y:Sat Apr 16 00:43:03 2022|pool busy, 3/3, Forwarding request of client PID#22815 to serverZ, matrix size 4x4
Z:Sat Apr 16 00:43:03 2022|pool busy, 1/3, Worker PID#22739is handling client #PID22815, matrix size 4x4
Y:Sat Apr 16 00:43:31 2022|SIGINT Received, terminating Z and exiting server Y Total requests handled: 11, 8 invertible, 3 not. 5 requests were forwarded.
Z:Sat Apr 16 00:43:31 2022|SIGINT Received, exiting server Z. Total requests handled 5, 4 invertible, 1 not.
```