Exercise 4 – Correcting a run-time error in MT Server

Explain the problem with your own words.

The problem in this program is that the delegate socket variable (last argument of *pthread_create* function) is passed to a thread by reference in its creation. Because of that, several execution threads could access to the same delegate socket at the same time, producing unexpected behaviour of the program.

```
pthread_create(&threadForClient, (pthread_attr_t *) NULL, clientServerProtocol, &delegateSocket)
```

Figure 1: Function to create a thread

Create an environment for reproducing the problem.

I have created a Bash script to generate a lot of connections to the server and increased its Backlog to accept up to 100 connections. Here is the code of the script:

```
#!/bin/bash

# Run 30 clients in background
# and wait for them to finish
for i in {1..30} ; do
    ../bin/mt_client.o 127.0.0.1 60001 &
done
wait
```

Figure 2: Script for generating clients

Document the results that you have obtained.

After one execution of the source code obtained from paloalto.unileon.es (Multithreaded Server Links) with that script and the Backlog size change I obtained this result:

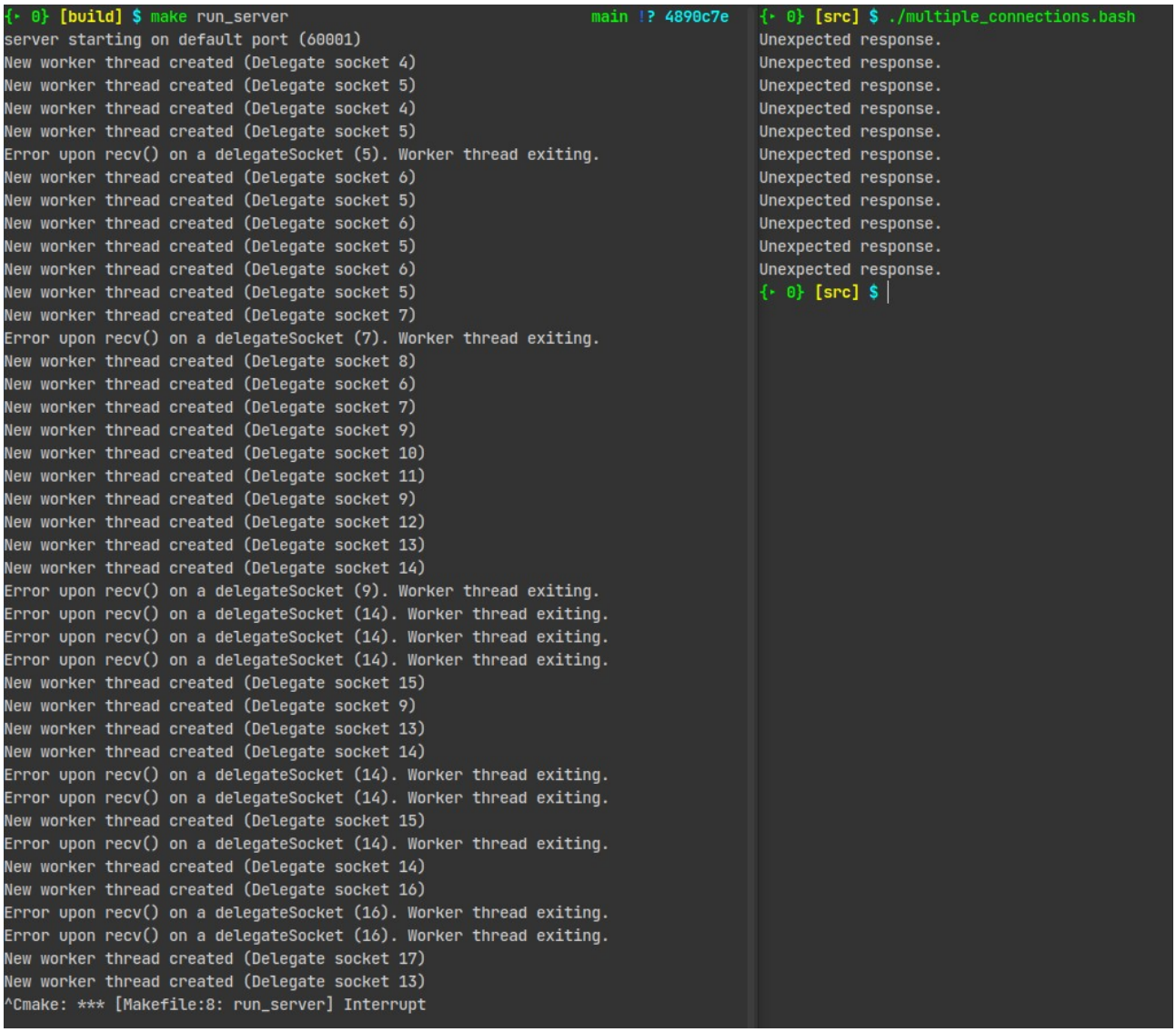

Figure 3: Execution of 30 threads

(Output is limited to the correct or bad execution of socket functions only for the purpose of these exercises).

Here we can see the unexpected behaviour mentioned in the first exercise. Several threads access to the delegate sockets with id 5, 7, 9, 14 and 16. When I stop the server, the number of “Unexpected response” messages at the clients is equal to the number of errors produced by the *recv* function on the server, because the same delegate socket is accessed by multiple threads.

Devise a solution to this problem, implement it and demonstrate that it works.

The solution that I provide is to **allocate** some **memory** on the heap to store the delegate socket, so that the created thread is the only thread that could access to that value and make operations with it. The C code for this is provided in the screenshot below.

```c
// Allocate memory on the heap for the delegate socket
int *del_sock = (int *) malloc(sizeof(int));
*del_sock = delegateSocket;
// Each thread has its own delegate socket

if (pthread_create(&threadForClient, (pthread_attr_t *) NULL, clientServerProtocol, del_sock) ≠ 0)
    perror("Thread creation error\n");
else
    printf("New worker thread created (Delegate socket %d)\n", delegateSocket);
```

Figure 4: Modification of the source code

Since the type of the delegate socket is an integer, the value must be stored into a pointer to an *int* in order not to waste memory.

The next image illustrates that now the server is responding to all the connections correctly.

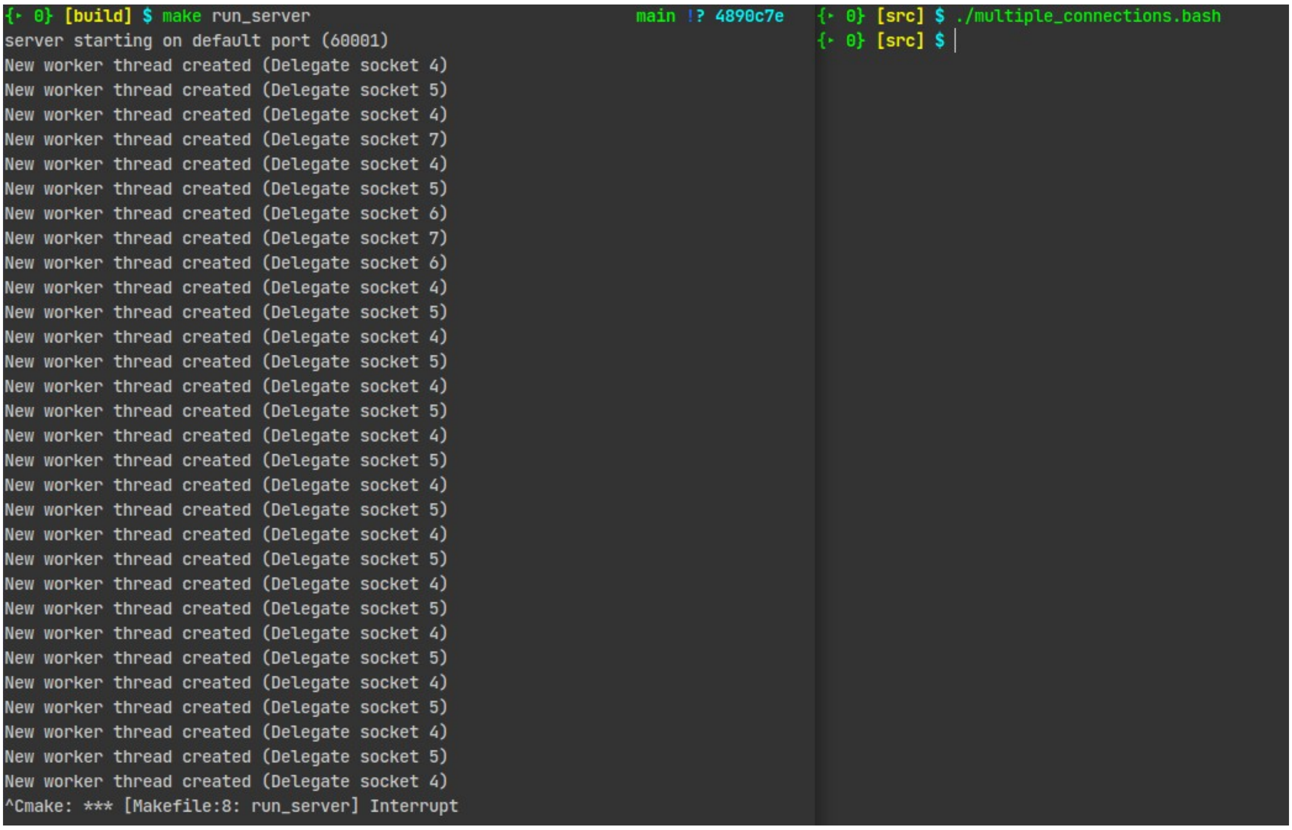

Figure 5: Successful execution of server and 30 clients

Now the right terminal does not show any “Unexpected response” error messages after terminating the server, which executes sucessfully.