LABORATORY 7 – OPERATING SYSTEMS

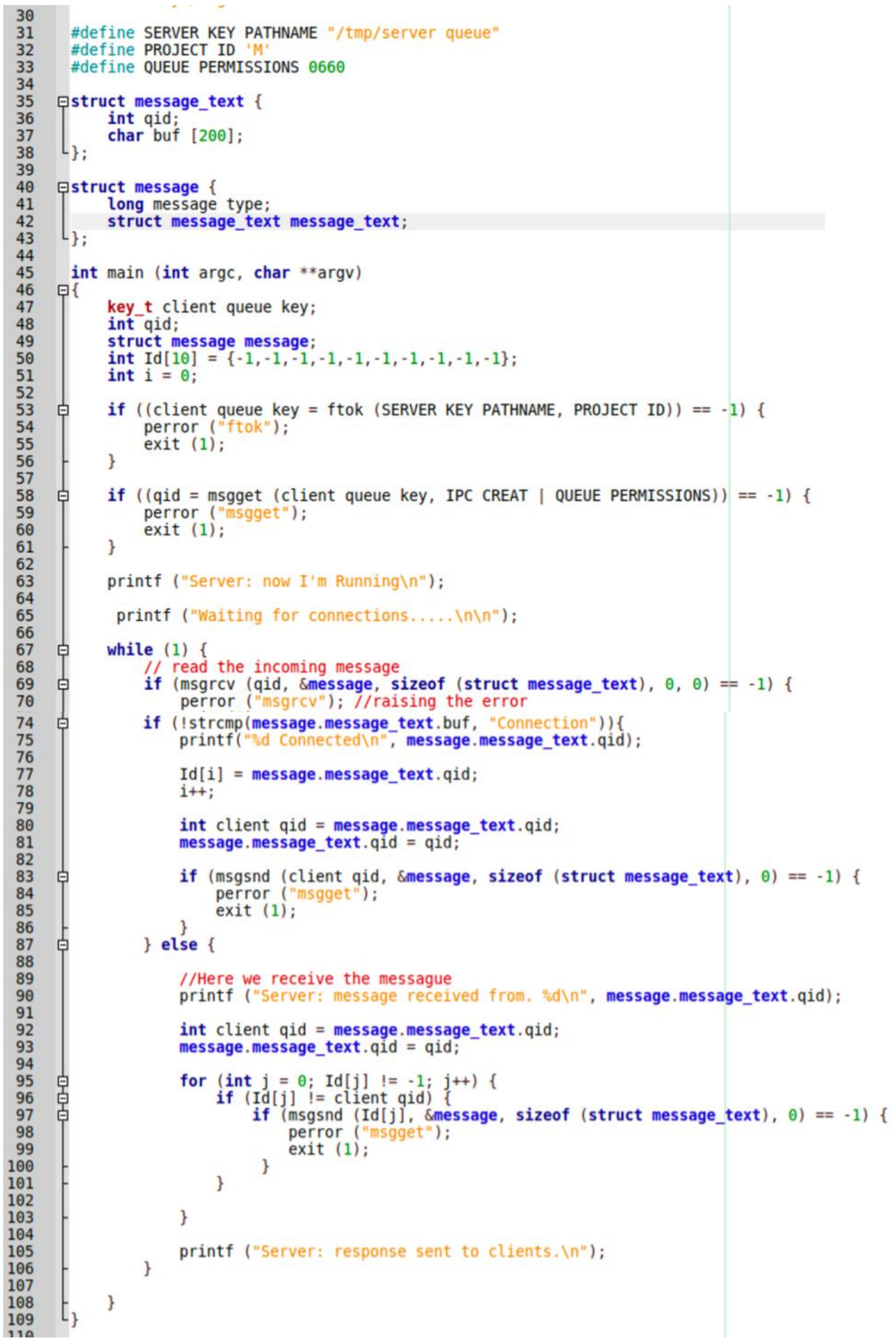
RUFINO GARCIA SANCHEZ

EXERCISE 7 -task 1

Write a simple chat application with server and several clients, which use IPC System V messages for the IPC communication.

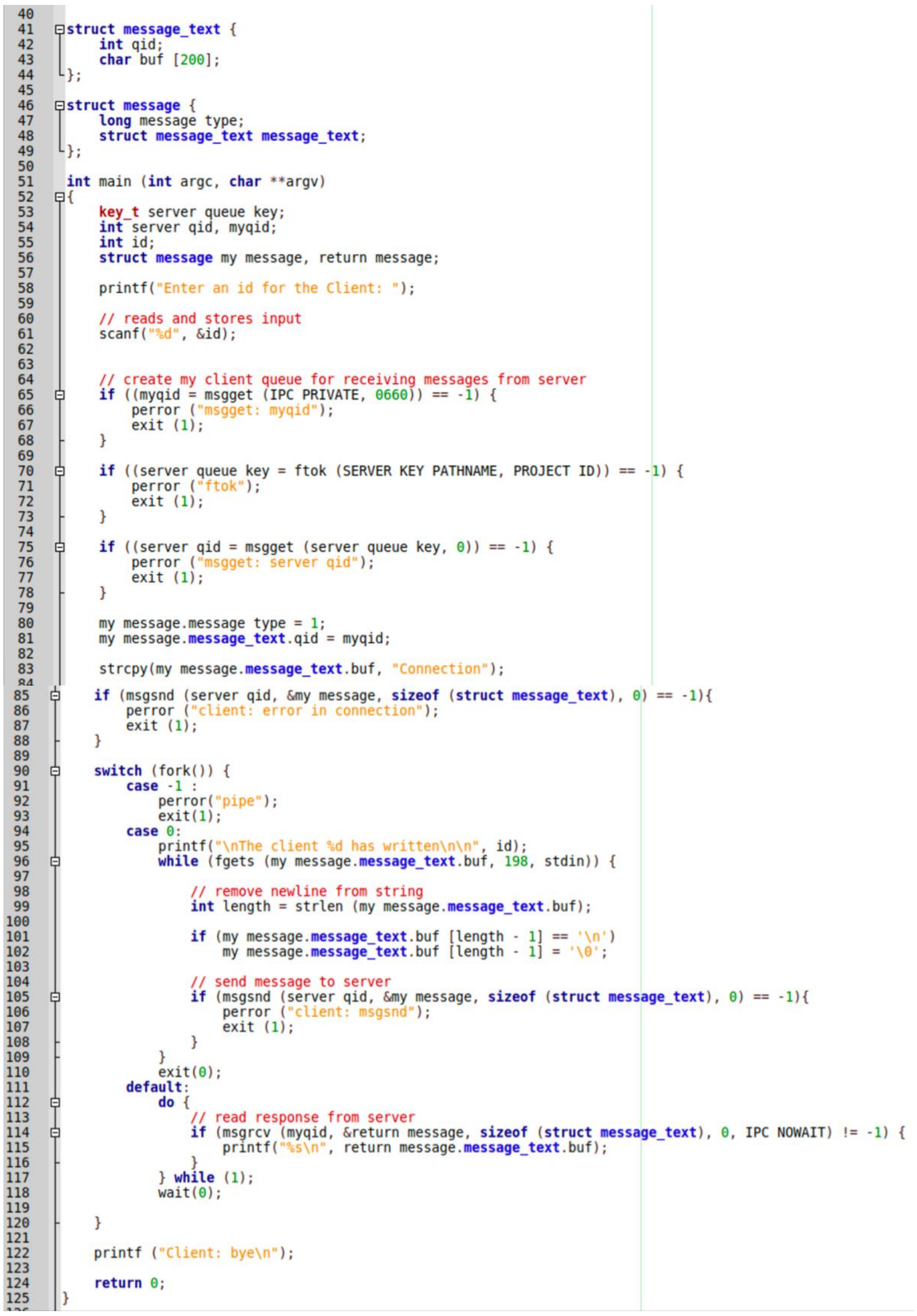
For this **first task we are going to use IPC Systems V.** The code of this program is the following:

From the **server.c:**



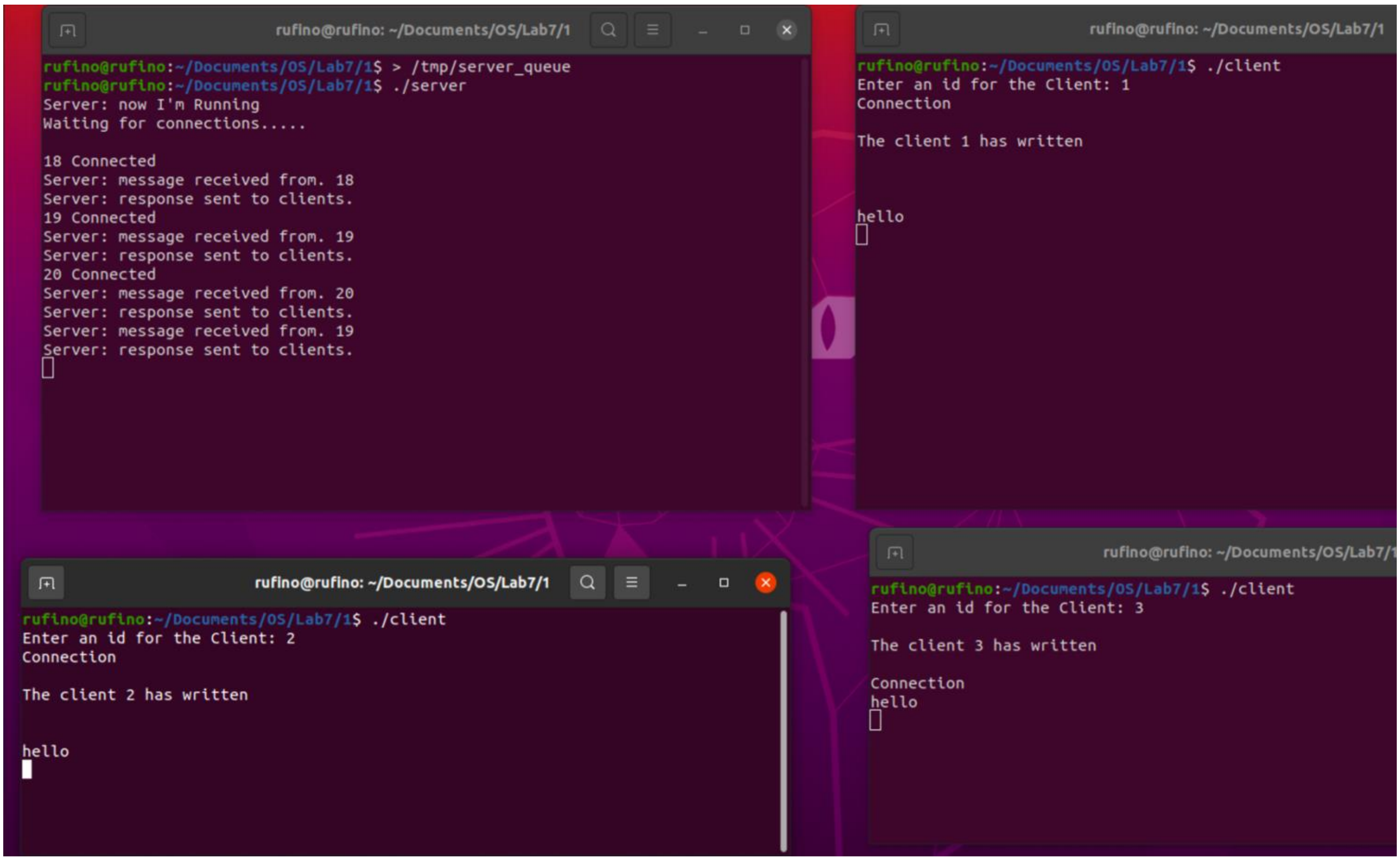
```

#define SERVER KEY PATHNAME "/tmp/server queue"
#define PROJECT ID 'M'
#define QUEUE PERMISSIONS 0660

struct message_text {
    int qid;
    char buf [200];
};

struct message {
    long message type;
    struct message_text message_text;
};

int main (int argc, char **argv)
{
    key_t client queue key;
    int qid;
    struct message message;
    int Id[10] = {-1,-1,-1,-1,-1,-1,-1,-1,-1,-1};
    int i = 0;

    if ((client queue key = ftok (SERVER KEY PATHNAME, PROJECT ID)) == -1) {
        perror ("ftok");
        exit (1);
    }

    if ((qid = msgget (client queue key, IPC CREAT | QUEUE PERMISSIONS)) == -1) {
        perror ("msgget");
        exit (1);
    }

    printf ("Server: now I'm Running\n");

     printf ("Waiting for connections.....\n\n");

    while (1) {
        // read the incoming message
        if (msgrcv (qid, &message, sizeof (struct message_text), 0, 0) == -1) {
            perror ("msgrcv"); //raising the error
        if (!strcmp(message.message_text.buf, "Connection")){
            printf("%d Connected\n", message.message_text.qid);

            Id[i] = message.message_text.qid;
            i++;

            int client qid = message.message_text.qid;
            message.message_text.qid = qid;

            if (msgsnd (client qid, &message, sizeof (struct message_text), 0) == -1) {
                perror ("msgget");
                exit (1);
            }
        } else {

            //Here we receive the messague
            printf ("Server: message received from. %d\n", message.message_text.qid);

            int client qid = message.message_text.qid;
            message.message_text.qid = qid;

            for (int j = 0; Id[j] != -1; j++) {
                if (Id[j] != client qid) {
                    if (msgsnd (Id[j], &message, sizeof (struct message_text), 0) == -1) {
                        perror ("msgget");
                        exit (1);
                    }
                }

            }

            printf ("Server: response sent to clients.\n");
        }

    }
}
```

Code from the **client.c:**



```c

struct message_text {
    int qid;
    char buf [200];
};

struct message {
    long message type;
    struct message_text message_text;
};

int main (int argc, char **argv)
{
    key_t server queue key;
    int server qid, myqid;
    int id;
    struct message my message, return message;

    printf("Enter an id for the Client: ");

    // reads and stores input
    scanf("%d", &id);


    // create my client queue for receiving messages from server
    if ((myqid = msgget (IPC PRIVATE, 0660)) == -1) {
        perror ("msgget: myqid");
        exit (1);
    }

    if ((server queue key = ftok (SERVER KEY PATHNAME, PROJECT ID)) == -1) {
        perror ("ftok");
        exit (1);
    }

    if ((server qid = msgget (server queue key, 0)) == -1) {
        perror ("msgget: server qid");
        exit (1);
    }

    my message.message type = 1;
    my message.message_text.qid = myqid;

    strcpy(my message.message_text.buf, "Connection");

    if (msgsnd (server qid, &my message, sizeof (struct message_text), 0) == -1){
        perror ("client: error in connection");
        exit (1);
    }

    switch (fork()) {
        case -1 :
            perror("pipe");
            exit(1);
        case 0:
            printf("\nThe client %d has written\n\n", id);
            while (fgets (my message.message_text.buf, 198, stdin)) {

                // remove newline from string
                int length = strlen (my message.message_text.buf);

                if (my message.message_text.buf [length - 1] == '\n')
                    my message.message_text.buf [length - 1] = '\0';

                // send message to server
                if (msgsnd (server qid, &my message, sizeof (struct message_text), 0) == -1){
                    perror ("client: msgsnd");
                    exit (1);
                }
            }
            exit(0);
        default:
            do {
                // read response from server
                if (msgrcv (myqid, &return message, sizeof (struct message_text), 0, IPC NOWAIT) != -1) {
                    printf("%s\n", return message.message_text.buf);
                }
            } while (1);
            wait(0);

    }

    printf ("Client: bye\n");

    return 0;
}
```

This is running the codes:



```
rufino@rufino:~/Documents/OS/Lab7/1$ > /tmp/server_queue
rufino@rufino:~/Documents/OS/Lab7/1$ ./server
Server: now I'm Running
Waiting for connections.....

18 Connected
Server: message received from. 18
Server: response sent to clients.
19 Connected
Server: message received from. 19
Server: response sent to clients.
20 Connected
Server: message received from. 20
Server: response sent to clients.
Server: message received from. 19
Server: response sent to clients.
```

```
rufino@rufino:~/Documents/OS/Lab7/1$ ./client
Enter an id for the Client: 1
Connection

The client 1 has written


hello
```

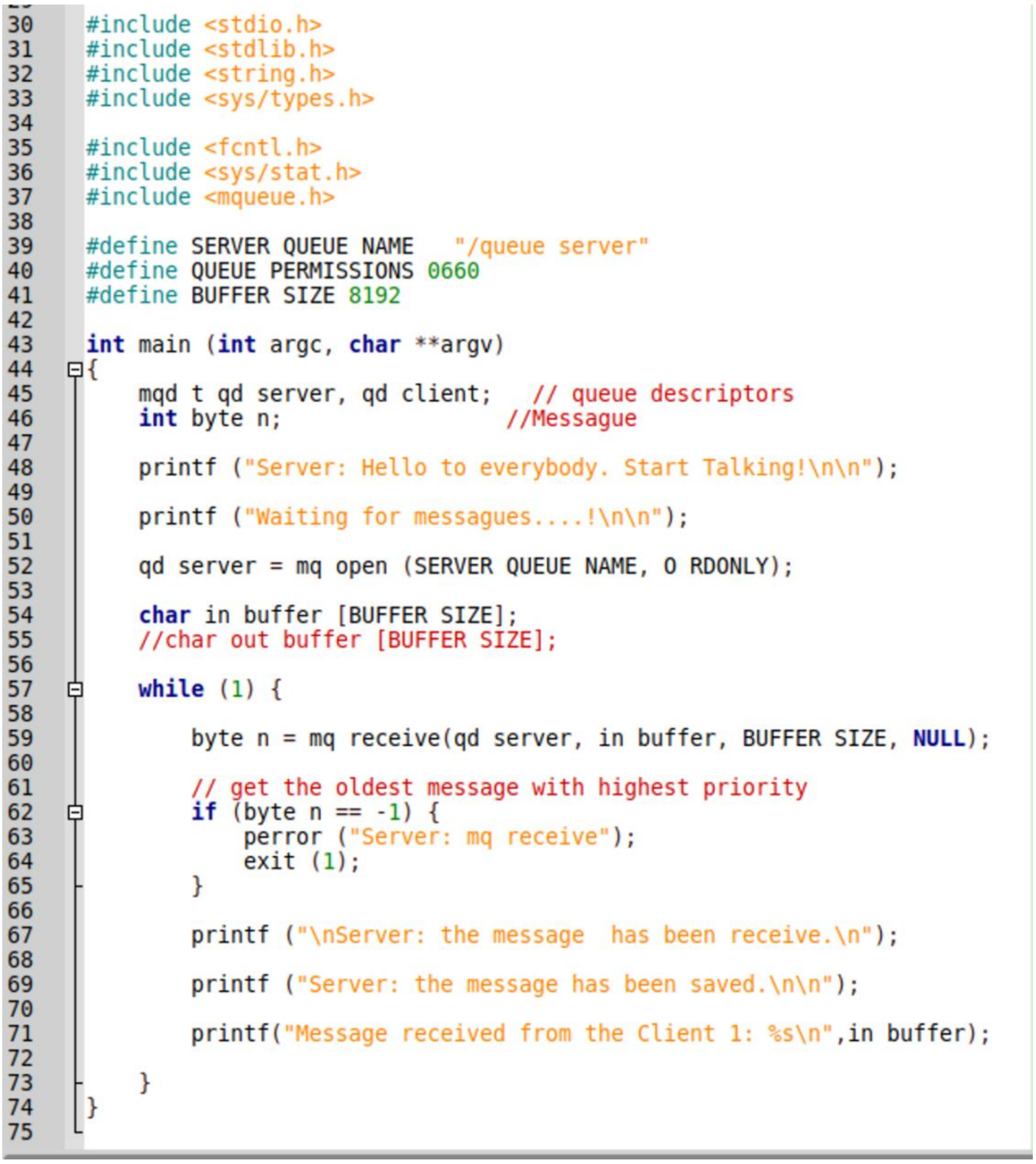
```
rufino@rufino:~/Documents/OS/Lab7/1$ ./client
Enter an id for the Client: 2
Connection

The client 2 has written


hello
```

```
rufino@rufino:~/Documents/OS/Lab7/1$ ./client
Enter an id for the Client: 3

The client 3 has written

Connection
hello
```

Task 2

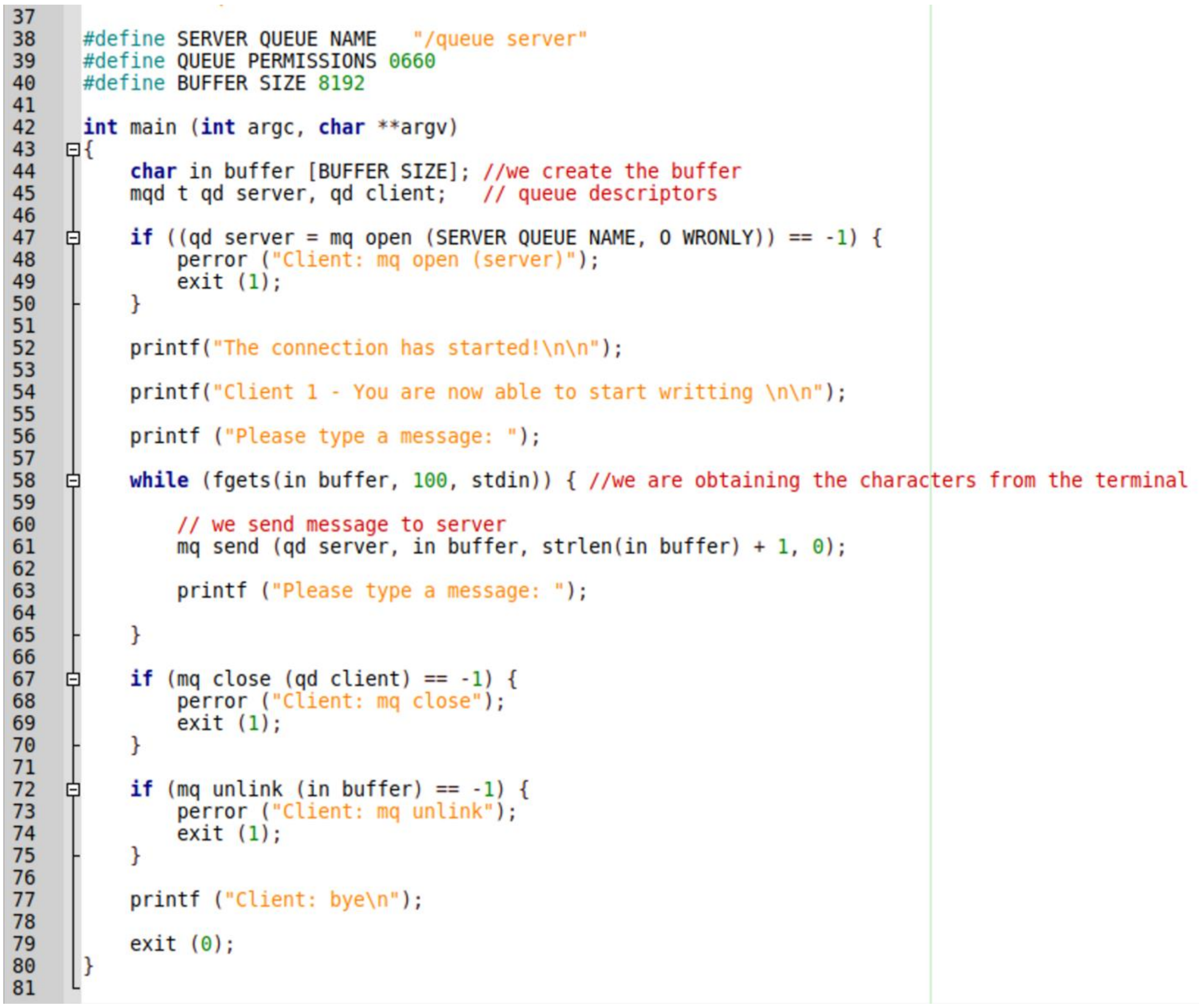
Implement similar functionality using IPC Posix package.

The code of the **server.c is the following:**



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>

#include <fcntl.h>
#include <sys/stat.h>
#include <mqueue.h>

#define SERVER QUEUE NAME   "/queue server"
#define QUEUE PERMISSIONS 0660
#define BUFFER SIZE 8192

int main (int argc, char **argv)
{
    mqd t qd server, qd client;   // queue descriptors
    int byte n;                  //Messague

    printf ("Server: Hello to everybody. Start Talking!\n\n");

    printf ("Waiting for messagues....!\n\n");

    qd server = mq open (SERVER QUEUE NAME, O RDONLY);

    char in buffer [BUFFER SIZE];
    //char out buffer [BUFFER SIZE];

    while (1) {

        byte n = mq receive(qd server, in buffer, BUFFER SIZE, NULL);

        // get the oldest message with highest priority
        if (byte n == -1) {
            perror ("Server: mq receive");
            exit (1);
        }

        printf ("\nServer: the message  has been receive.\n");

        printf ("Server: the message has been saved.\n\n");

        printf("Message received from the Client 1: %s\n",in buffer);

    }
}
```

And the code of the **client.c**:



```c

#define SERVER QUEUE NAME   "/queue server"
#define QUEUE PERMISSIONS 0660
#define BUFFER SIZE 8192

int main (int argc, char **argv)
{
    char in buffer [BUFFER SIZE]; //we create the buffer
    mqd t qd server, qd client;   // queue descriptors

    if ((qd server = mq open (SERVER QUEUE NAME, O WRONLY)) == -1) {
        perror ("Client: mq open (server)");
        exit (1);
    }

    printf("The connection has started!\n\n");

    printf("Client 1 - You are now able to start writting \n\n");

    printf ("Please type a message: ");

    while (fgets(in buffer, 100, stdin)) { //we are obtaining the characters from the terminal

        // we send message to server
        mq send (qd server, in buffer, strlen(in buffer) + 1, 0);

        printf ("Please type a message: ");

    }

    if (mq close (qd client) == -1) {
        perror ("Client: mq close");
        exit (1);
    }

    if (mq unlink (in buffer) == -1) {
        perror ("Client: mq unlink");
        exit (1);
    }

    printf ("Client: bye\n");

    exit (0);
}

```