

# **LAB ASSIGNMENT-2**

## **CSN-361 Computer Networks Laboratory**

**Submitted by - Prateek Mali  
Enrollment no. - 17114059**

### **Problem Statements**

1. Write a socket program in C to connect two nodes on a network to communicate with each other, where one socket listens on a particular port at an IP, while other socket reaches out to the other to form a connection.
2. Write a C program to demonstrate both Zombie and Orphan process.

## Implementations details

### 1

Buffer is used as a data structure.

Sockets are implemented for the communication between client and server.

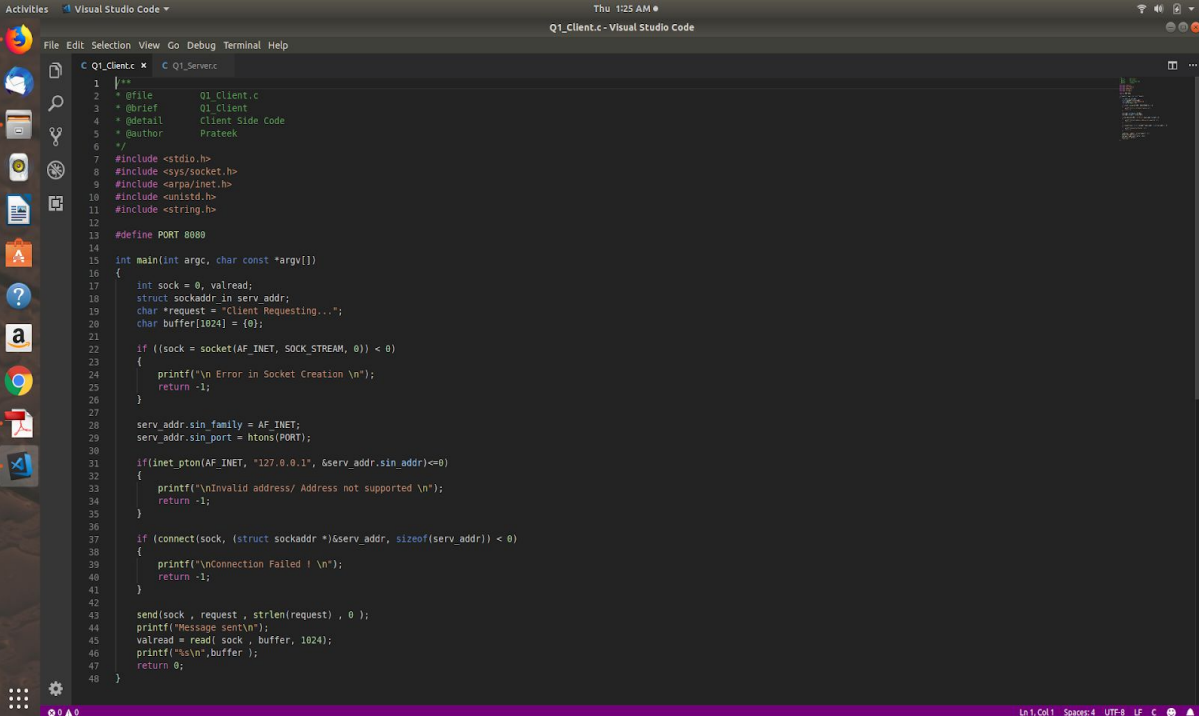
The solution has 2 programs:

Server : This program creates a node which has a socket to listen on the port 8080 (localhost).

Client : This program created a node which sends a request on port 8080 to be read by the server program.

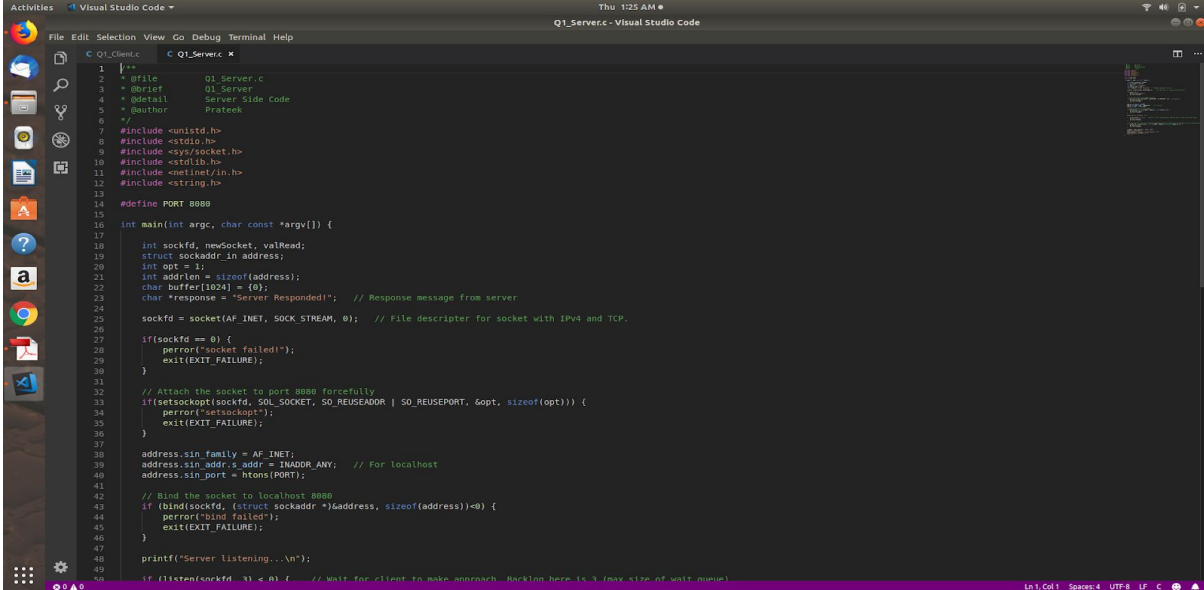
## Code Snippets

Client Side Code :-

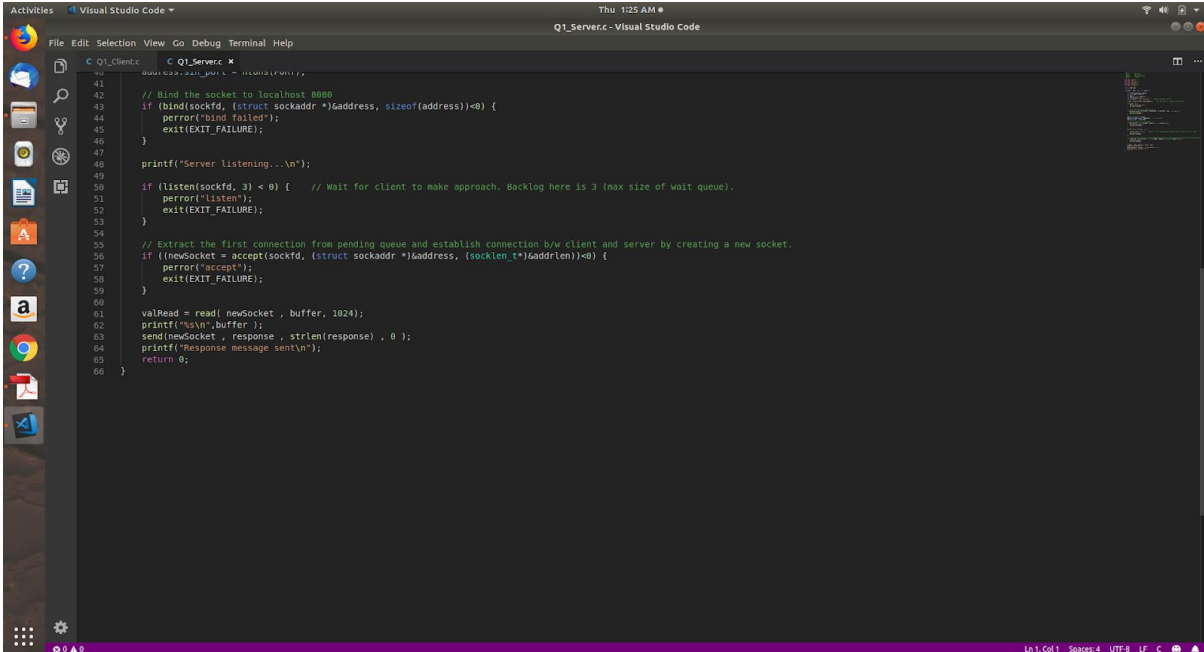


```
1  /**
2   * @file      Q1_Client.c
3   * @brief     Q1 Client
4   * @detail    Client Side Code
5   * @author    Prateek
6   */
7  #include <stdio.h>
8  #include <sys/socket.h>
9  #include <arpa/inet.h>
10 #include <unistd.h>
11 #include <string.h>
12
13 #define PORT 8080
14
15 int main(int argc, char const *argv[])
16 {
17     int sock = 0, valread;
18     struct sockaddr_in serv_addr;
19     char *request = "Client Requesting...";
20     char buffer[1024] = {0};
21
22     if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
23     {
24         printf("\n Error in Socket Creation \n");
25         return -1;
26     }
27
28     serv_addr.sin_family = AF_INET;
29     serv_addr.sin_port = htons(PORT);
30
31     if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0)
32     {
33         printf("\nInvalid address/ Address not supported \n");
34         return -1;
35     }
36
37     if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
38     {
39         printf("\nConnection Failed ! \n");
40         return -1;
41     }
42
43     send(sock, request, strlen(request), 0);
44     printf("Message sent\n");
45     valread = read(sock, buffer, 1024);
46     printf("%s\n", buffer);
47     return 0;
48 }
```

## Server Side Code :-



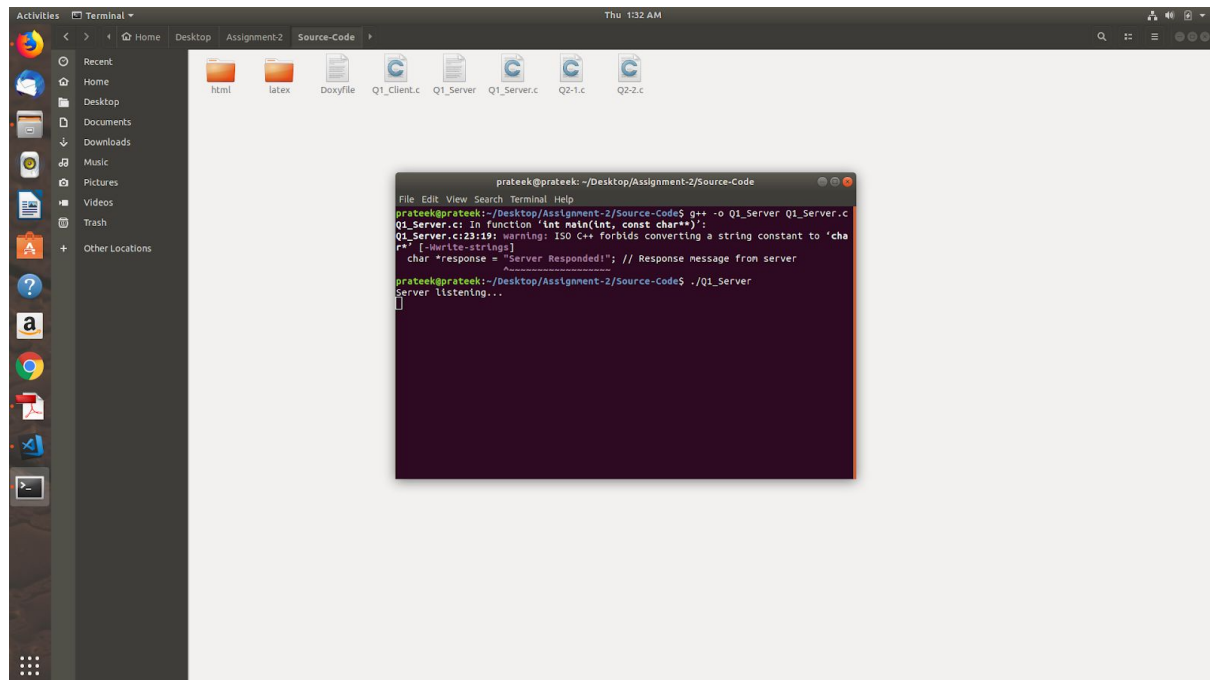
```
1 1
2 2 * @file Q1_Server.c
3 3 * @brief Q1 Server
4 4 * @detail Server Side Code
5 5 * @author Prateek
6 6 //
7 7 #include <unistd.h>
8 8 #include <stdio.h>
9 9 #include <sys/socket.h>
10 10 #include <stdlib.h>
11 11 #include <netinet/in.h>
12 12 #include <string.h>
13 13
14 14 #define PORT 8080
15 15
16 16 int main(int argc, char const *argv[]) {
17 17
18 18     int sockfd, newSocket, valRead;
19 19     struct sockaddr_in address;
20 20     int opt = 1;
21 21     int addrlen = sizeof(address);
22 22     char buffer[1024] = {0};
23 23     char *response = "Server Responded!"; // Response message from server
24 24
25 25     sockfd = socket(AF_INET, SOCK_STREAM, 0); // File descriptor for socket with IPv4 and TCP.
26 26
27 27     if(sockfd == 0) {
28 28         perror("socket failed");
29 29         exit(EXIT_FAILURE);
30 30     }
31 31
32 32     // Attach the socket to port 8080 forcefully
33 33     if(setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt, sizeof(opt))) {
34 34         perror("setsockopt");
35 35         exit(EXIT_FAILURE);
36 36     }
37 37
38 38     address.sin_family = AF_INET;
39 39     address.sin_addr.s_addr = INADDR_ANY; // For localhost
40 40     address.sin_port = htons(PORT);
41 41
42 42     // Bind the socket to localhost 8080
43 43     if (bind(sockfd, (struct sockaddr *)&address, sizeof(address))<0) {
44 44         perror("bind failed");
45 45         exit(EXIT_FAILURE);
46 46     }
47 47
48 48     printf("Server listening...\n");
49 49
50 50     if (listen(sockfd, 3) < 0) { // Wait for client to make approach. Backlog here is 3 (max size of wait queue)
51 51
52 52     }
```



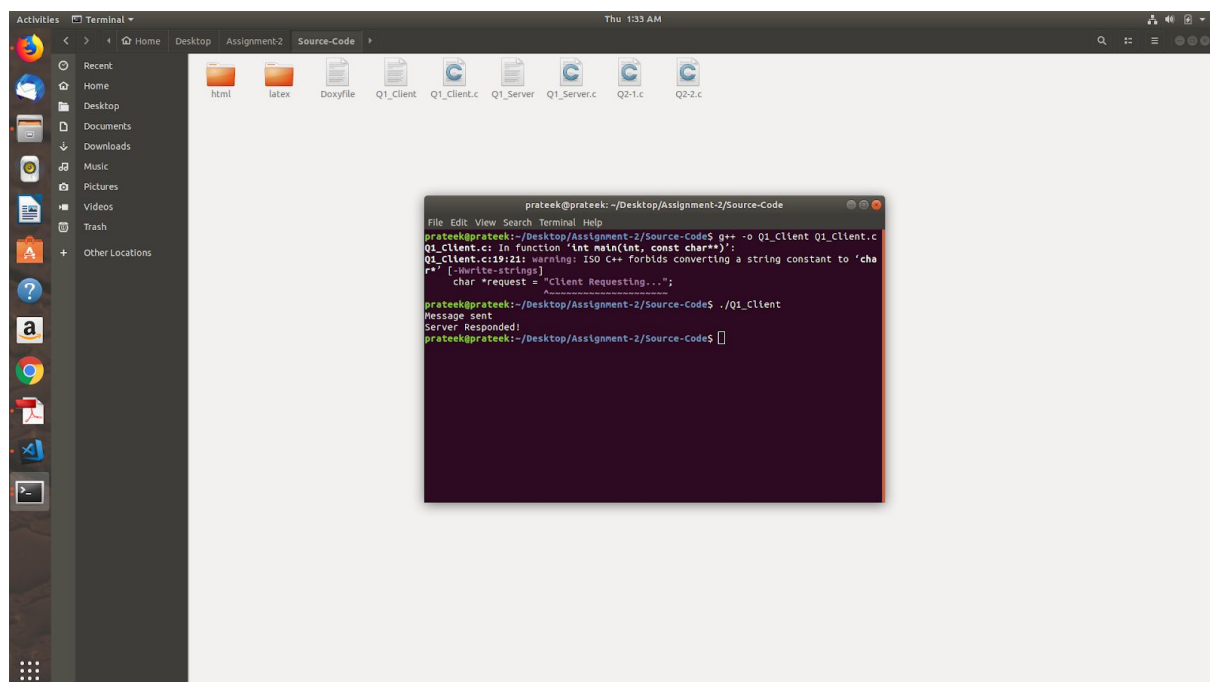
```
53 53     }
54 54
55 55     // Extract the first connection from pending queue and establish connection b/w client and server by creating a new socket.
56 56     if (newSocket = accept(sockfd, (struct sockaddr *)&address, (socklen_t *)&addrlen)<0) {
57 57         perror("accept");
58 58         exit(EXIT_FAILURE);
59 59     }
60 60
61 61     valRead = read(newSocket, buffer, 1024);
62 62     printf("%s\n", buffer);
63 63     send(newSocket, response, strlen(response), 0);
64 64     printf("Response message sent\n");
65 65     return 0;
66 66 }
```

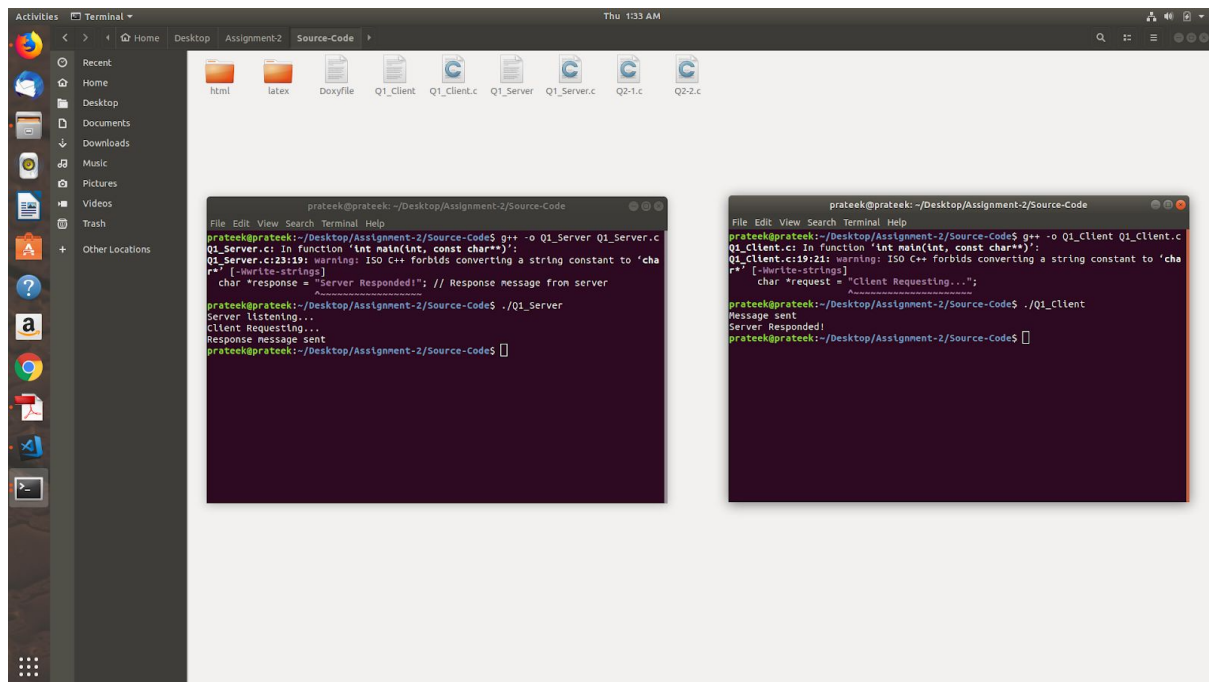
# Results

1. Making the server ready to listen for requests made by client.



2. Client making request and getting the response from the server.





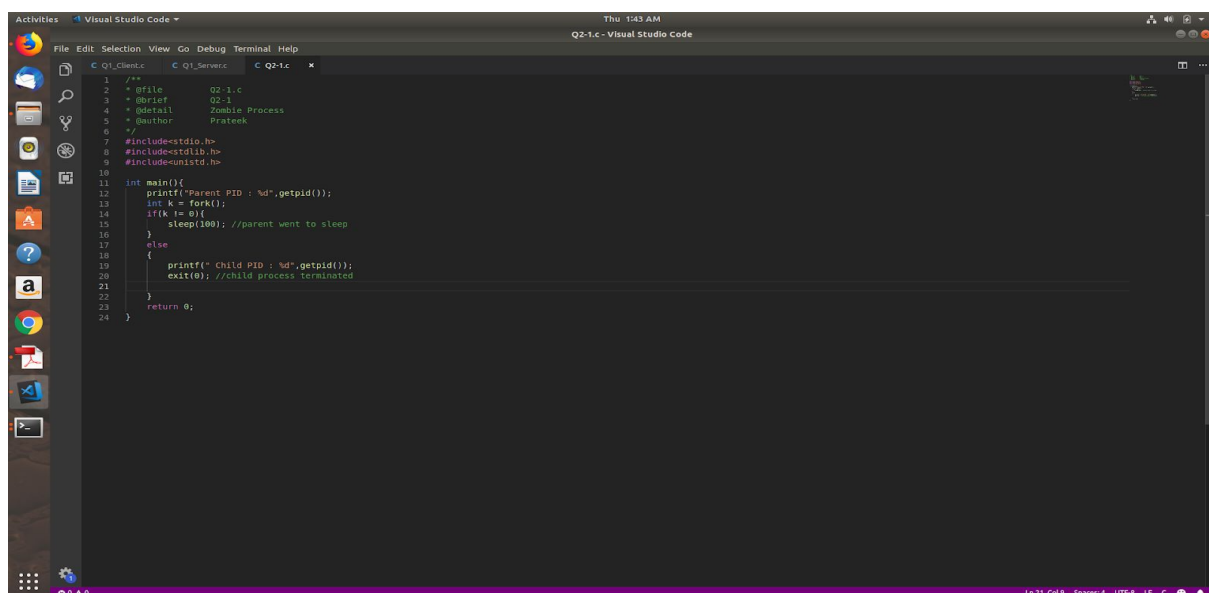
2

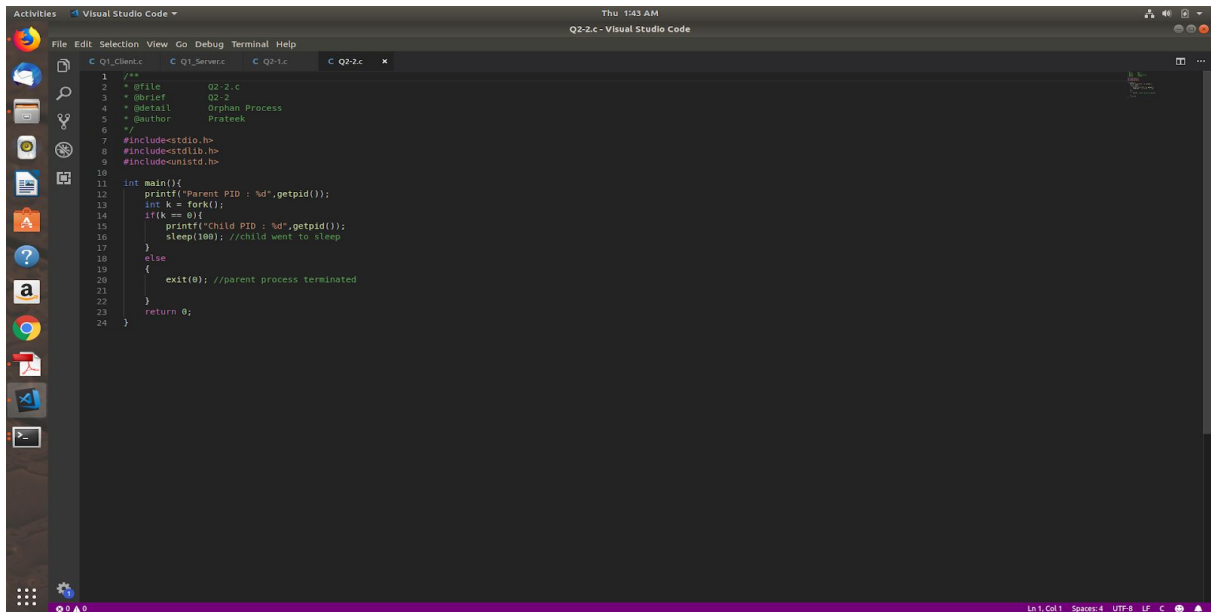
No major data structure is used.

The programs uses different System Calls to make zombie and orphan processes.

For ex:- fork(), exit(), sleep().

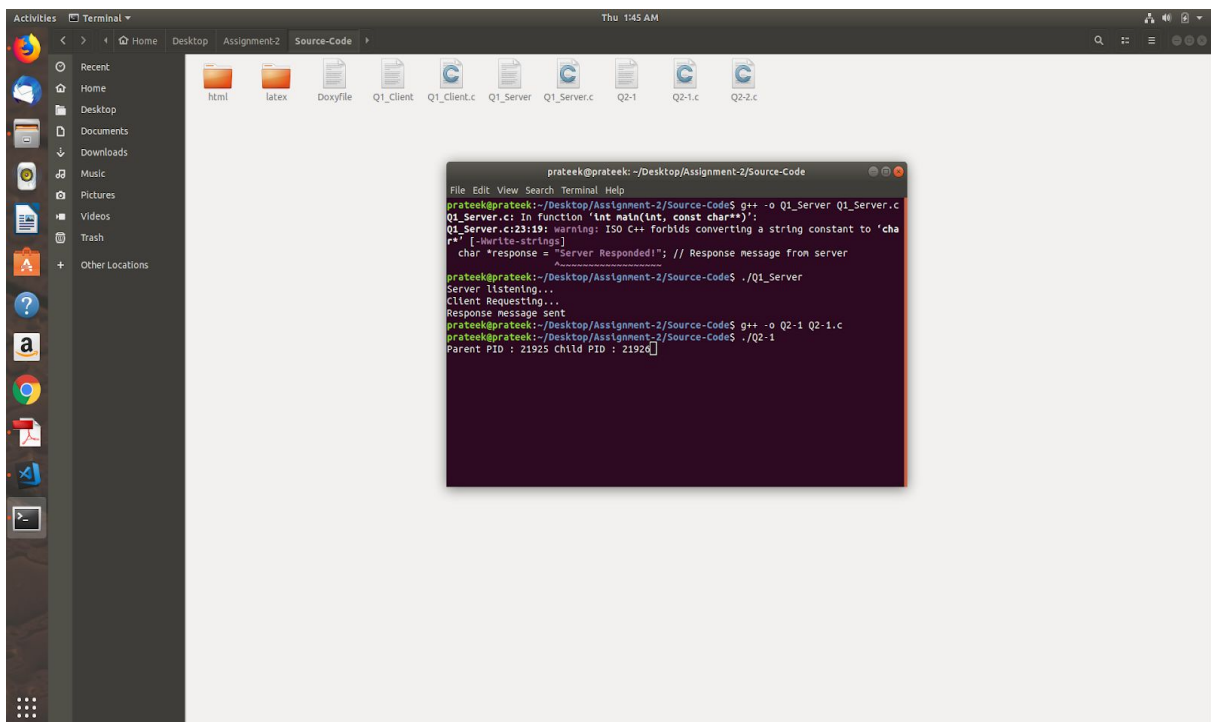
## Code Snippets (Zombie and Orphan Processes)





```
1  /**
2   * @file      Q2-2.c
3   * @brief     Q2-2
4   * @detail    Orphan Process
5   * @author    Prateek
6   */
7  #include<stdio.h>
8  #include<stdlib.h>
9  #include<unistd.h>
10
11 int main(){
12     printf("Parent PID : %d",getpid());
13     int k = fork();
14     if(k == 0){
15         printf("Child PID : %d",getpid());
16         sleep(100); //child went to sleep
17     }
18     else
19     {
20         exit(0); //parent process terminated
21     }
22
23     return 0;
24 }
```

## Results



```
prateek@prateek: ~/Desktop/Assignment-2/Source-Code
File Edit View Search Terminal Help
prateek@prateek:~/Desktop/Assignment-2/Source-Code$ g++ -o Q1_Server Q1_Server.c
Q1_Server.c: In function 'int main(int, const char**)':
Q1_Server.c:23:119: warning: ISO C++ forbids converting a string constant to 'char*' [-Wwrite-strings]
    char *response = "Server Responded!"; // Response message from server
                                                                    ^
prateek@prateek:~/Desktop/Assignment-2/Source-Code$ ./Q1_Server
Server listening...
Client Requesting...
Response message sent
prateek@prateek:~/Desktop/Assignment-2/Source-Code$ g++ -o Q2-1 Q2-1.c
prateek@prateek:~/Desktop/Assignment-2/Source-Code$ ./Q2-1
Parent PID : 21925 Child PID : 21926]
```

