



Lab Assignment- 2

CSN-361: Computer Networks Laboratory

August 1, 2019

Gopi Kishan

Enrolment No. : 17114035

B.Tech, 3rd Yr

Computer Science and Engineering (CSE)





Problem Statement 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.

- **Algorithms and data structures used in the implementation**

Creating Server Process :

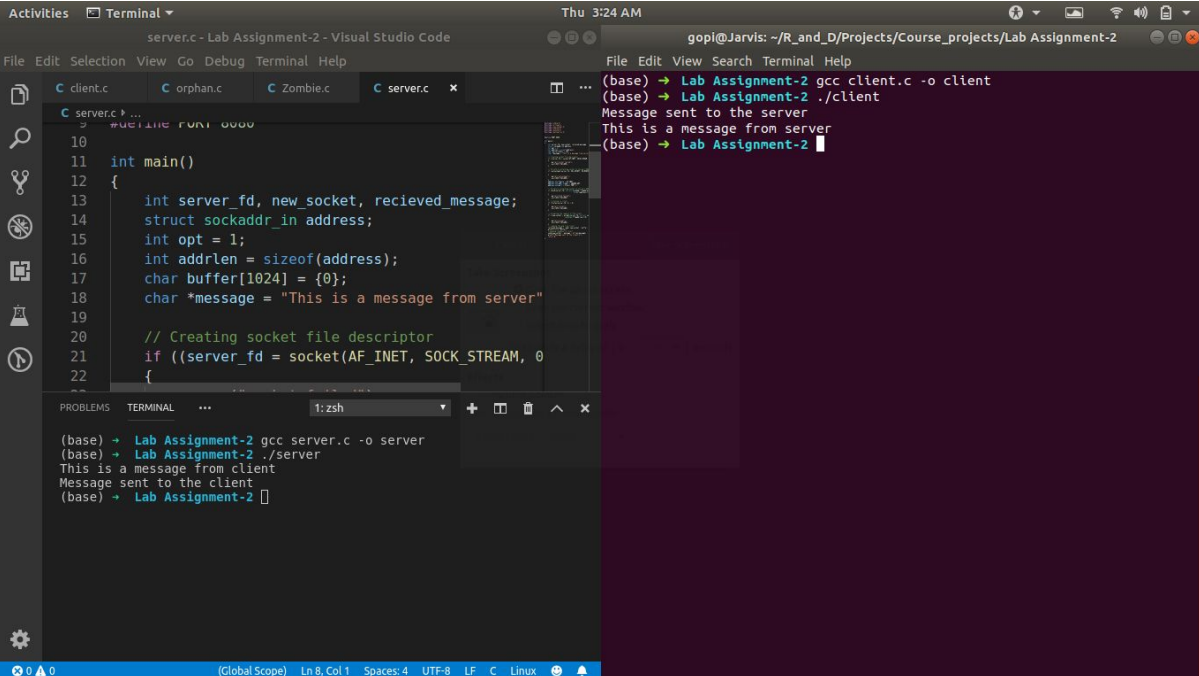
1. Create socket file descriptor using **socket(domain, type, protocol)**
2. Forcefully attach socket to the port 8080 using **bind(int sockfd, const struct sockaddr *addr, socklen_t, addrlen)**
3. Listen to client using **listen(int sockfd, int backlog)**
4. Accept the request of the client using **accept(int sockfd, struct sockaddr *addr, socklen_t, *addrlen)**
5. Read the message from client using **read(new_socket , buffer, 1024)**
6. Send response to client using **send(new_socket , message , strlen(message) , 0)**

Creating Client Process :

1. Create socket file descriptor using **socket(domain, type, protocol)**
2. Convert IPv4 and IPv6 addresses from text to binary form
3. Connect to the server using **connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen)**
4. Send request to server using **send(new_socket , message , strlen(message) , 0)**
5. Receive response from server using **read(new_socket , buffer, 1024).**

Data structures used :

- **struct sockaddr_in** : It is a structure containing an internet address.
 - **buffer** : character array of size 1024 to hold messages.
 - **message** : character array of strings.
-
- Snapshots of running the codes for each of the problems



```
server.c - Lab Assignment-2 - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
C client.c C orphan.c C Zombie.c C server.c x
C server.c
10 // #define PORT 8080
11 int main()
12 {
13     int server_fd, new_socket, recieved_message;
14     struct sockaddr_in address;
15     int opt = 1;
16     int addrlen = sizeof(address);
17     char buffer[1024] = {0};
18     char *message = "This is a message from server"
19
20     // Creating socket file descriptor
21     if ((server_fd = socket(AF_INET, SOCK_STREAM, 0
22
```

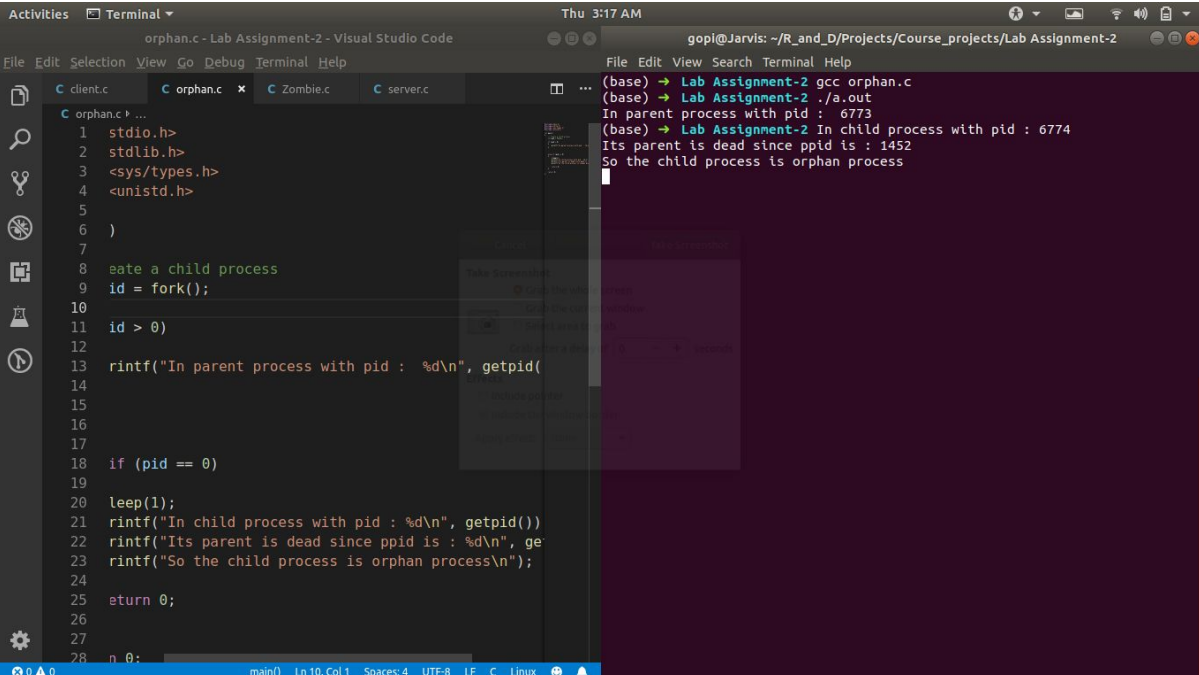
```
gopi@Jarvis: ~/R_and_D/Projects/Course_projects/Lab Assignment-2
File Edit View Search Terminal Help
(base) → Lab Assignment-2 gcc client.c -o client
(base) → Lab Assignment-2 ./client
Message sent to the server
This is a message from server
(base) → Lab Assignment-2

(base) → Lab Assignment-2 gcc server.c -o server
(base) → Lab Assignment-2 ./server
This is a message from client
Message sent to the client
(base) → Lab Assignment-2
```

Problem Statement 2

Write a C program to demonstrate both Zombie and Orphan process.

- Algorithms and data structures used in the implementation
No significant data structures used
- Snapshots of running the codes for each of the problems



The screenshot displays the Visual Studio Code editor with a C program named 'orphan.c' open. The code implements a parent process that forks a child process. The parent process prints its PID and then enters a loop. The child process prints its PID and its parent's PID (PPID). The terminal output shows the execution of the program, demonstrating the creation of an orphan process.

```
orphan.c - Lab Assignment-2 - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
C client.c C orphan.c x C Zombie.c C server.c
C orphan.c
1  stdio.h>
2  stdlib.h>
3  <sys/types.h>
4  <unistd.h>
5
6  )
7
8  eate a child process
9  id = fork();
10
11  id > 0)
12
13  rintf("In parent process with pid : %d\n", getpid(
14
15
16
17
18  if (pid == 0)
19
20  leep(1);
21  rintf("In child process with pid : %d\n", getpid())
22  rintf("Its parent is dead since ppid is : %d\n", ge
23  rintf("So the child process is orphan process\n");
24
25  eturn 0;
26
27
28  n 0;
```

```
gopi@Jarvis: ~/R_and_D/Projects/Course_projects/Lab Assignment-2
(base) → Lab Assignment-2 gcc orphan.c
(base) → Lab Assignment-2 ./a.out
In parent process with pid : 6773
(base) → Lab Assignment-2 In child process with pid : 6774
Its parent is dead since ppid is : 1452
So the child process is orphan process
```

The image shows a Visual Studio Code editor window with a terminal running a C program. The editor has four tabs: 'client.c', 'orphan.c', 'Zombie.c' (active), and 'server.c'. The 'Zombie.c' file contains the following code:

```
15 {
16     printf("Parent pid : %d\n",getpid());
17     printf("Parent is sleeping now\n");
18     sleep(3);
19 }
20
21 // Child process
22 else
23 {
24     printf("Child pid : %d\n", getpid());
25     printf("child process terminates now\n");
26     exit(0);
27 }
28
29 printf("pid of zombie child : %d\n", child_pid)
30
31 return 0;
32 }
33
```

The terminal window on the right shows the execution output:

```
(base) → Lab Assignment-2 gcc Zombie.c
(base) → Lab Assignment-2 ./a.out
Parent pid : 6938
Parent is sleeping now
Child pid : 6939
child process terminates now
pid of zombie child : 6939
(base) → Lab Assignment-2
```

The status bar at the bottom indicates the current file is 'main()' at line 14, column 24, with 4 spaces, UTF-8 encoding, LF line endings, and the C language on a Linux system.