Computer Network Laboratory

Assignment 2

Name: Hemant Singh

Enrollment Number: 17114038

Class: 3rd year, B.Tech CSE

Course: CSN-361

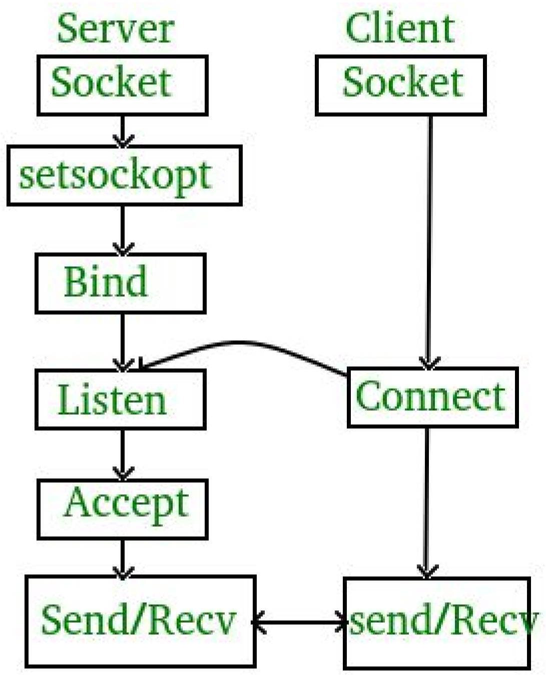
Problem Statements:

Problem 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 used :

● CLIENT SERVER MODEL ALGORITHM (for client and server as shown):



1. int sockfd = socket(domain, type, protocol)
2. int setsockopt(int sockfd, int level, int optname, const void *optval, socklen_t optlen);
3. int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
4. int listen(int sockfd, int backlog);

5. int new_socket= accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);

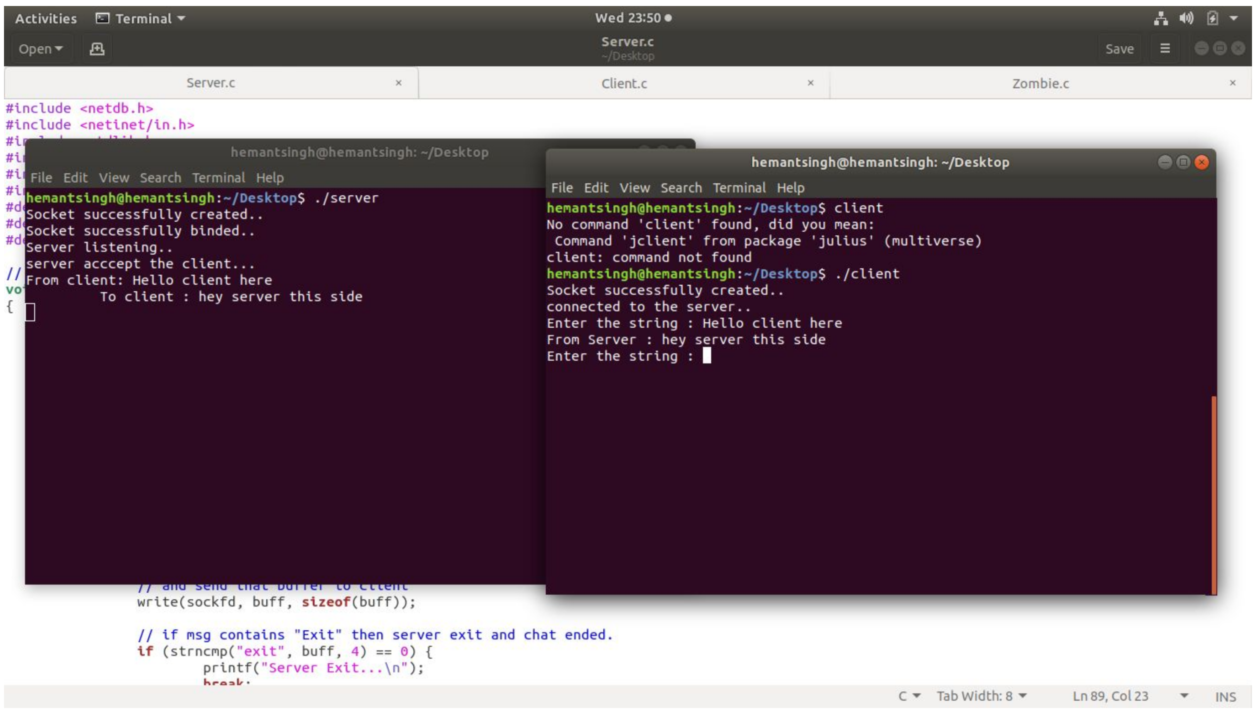
6. int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);

Data structures used :

● Int, char *, char []: To store the socket , strings, buffer

● struct sockaddr_in : for storing the port number and creating an instance of client and server

Screenshot :



```
#include <netdb.h>
#include <netinet/in.h>
```

```
hemantsingh@hemantsingh:~/Desktop$ ./server
Socket successfully created..
Socket successfully binded..
Server listening..
server acccept the client...
From client: Hello client here
         To client : hey server this side
```

```
hemantsingh@hemantsingh:~/Desktop$ client
No command 'client' found, did you mean:
 Command 'jclient' from package 'julius' (multiverse)
client: command not found
hemantsingh@hemantsingh:~/Desktop$ ./client
Socket successfully created..
connected to the server..
Enter the string : Hello client here
From Server : hey server this side
Enter the string :
```

```
                write(sockfd, buff, sizeof(buff));

                // if msg contains "Exit" then server exit and chat ended.
                if (strncmp("exit", buff, 4) == 0) {
                        printf("Server Exit...\n");
```

Problem 2 :

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

Algorithms used :

● fork() : To create new child

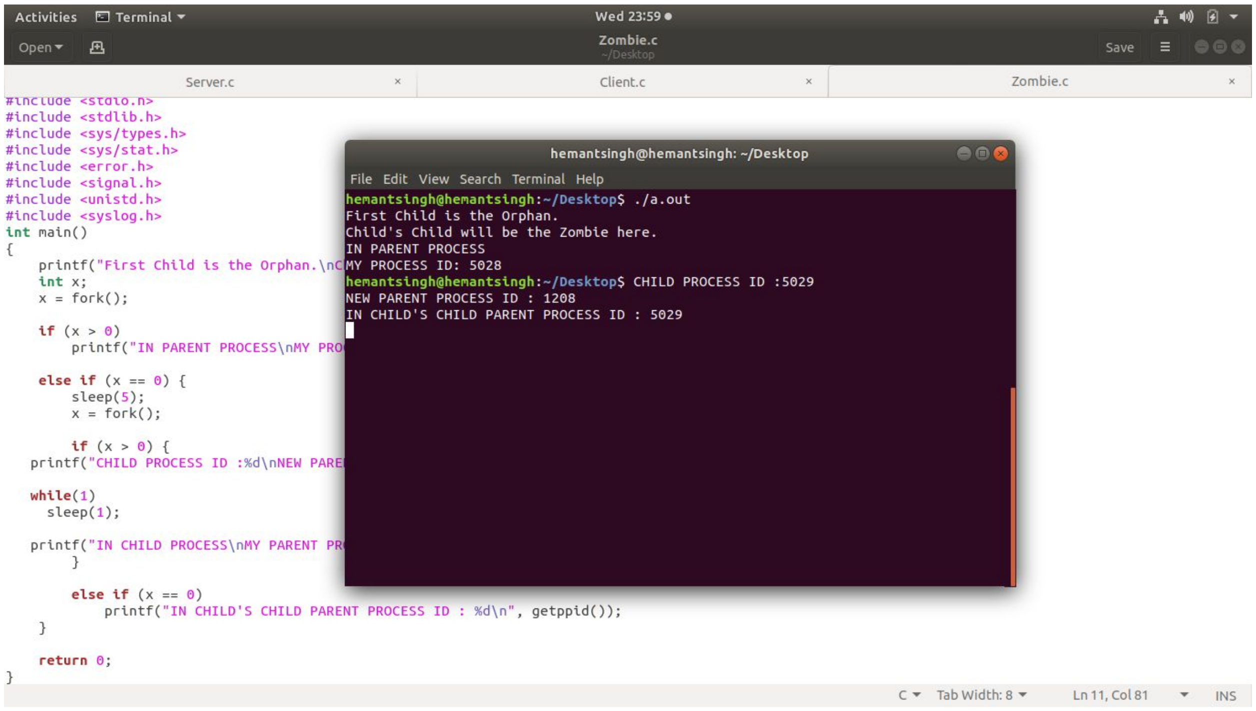
● sleep() : For proper functioning of the program

● Busy Waiting

Data Structures used :

● Int, To store the return value of fork() in it.

Screenshot :



```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <error.h>
#include <signal.h>
#include <unistd.h>
#include <syslog.h>
int main()
{
    printf("First Child is the Orphan.\nC
    int x;
    x = fork();

    if (x > 0)
        printf("IN PARENT PROCESS\nMY PRO

    else if (x == 0) {
        sleep(5);
        x = fork();

        if (x > 0) {
   printf("CHILD PROCESS ID :%d\nNEW PARE

   while(1)
     sleep(1);

   printf("IN CHILD PROCESS\nMY PARENT PR
        }

        else if (x == 0)
            printf("IN CHILD'S CHILD PARENT PROCESS ID : %d\n", getppid());
    }

    return 0;
}
```

```
hemantsingh@hemantsingh:~/Desktop$ ./a.out
First Child is the Orphan.
Child's Child will be the Zombie here.
IN PARENT PROCESS
MY PROCESS ID: 5028
hemantsingh@hemantsingh:~/Desktop$ CHILD PROCESS ID :5029
NEW PARENT PROCESS ID : 1208
IN CHILD'S CHILD PARENT PROCESS ID : 5029
```