

Computer Network Laboratory

Assignment 2

Name: Hemant Singh

Enrollment Number: 17114038

Class: 3rd year, B.Tech CSE

Course: CSN-361

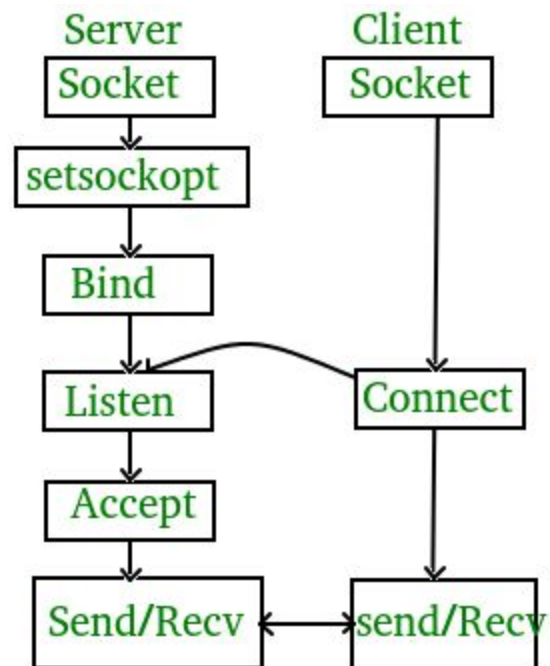
Problem Statements:

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);`

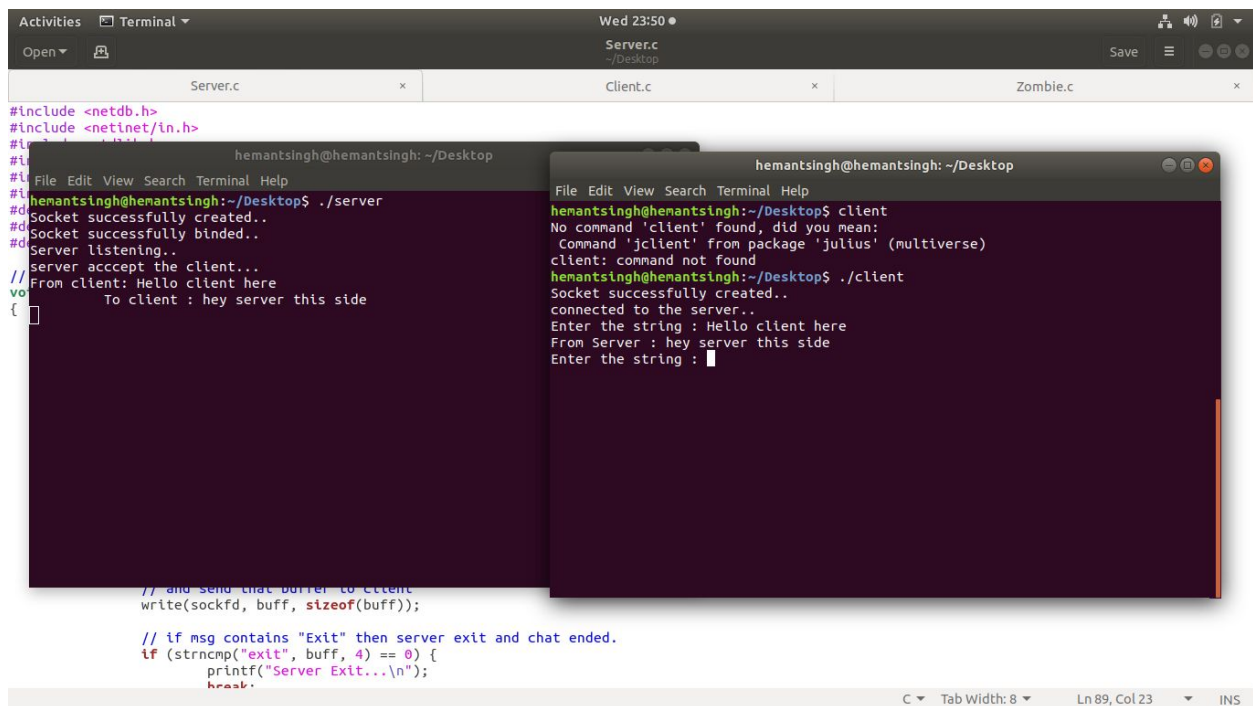
5. `int new_socket= accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);`

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>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>

int main()
{
    struct sockaddr_in serv_addr;
    int new_socket;
    int sockfd = 0;
    int n;
    char buf[1024];
    char *msg = "To client : hey server this side";

    // Create socket
    new_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (new_socket == -1)
    {
        printf("Error: socket creation\n");
        return -1;
    }

    // Bind socket to the port
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(8080);
    inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr);
    if (bind(new_socket, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
    {
        printf("Error: bind\n");
        return -1;
    }
    printf("Socket successfully created..\n");
    printf("Server listening..\n");

    // Accept the client connection
    struct sockaddr_in client_addr;
    socklen_t clntlen = sizeof(client_addr);
    int new_client_socket = accept(new_socket, (struct sockaddr *)&client_addr, &clntlen);
    if (new_client_socket == -1)
    {
        printf("Error: accept\n");
        return -1;
    }
    printf("server accept the client...\n");

    // Read the message from client
    n = read(new_client_socket, buf, 1024);
    if (n < 0)
    {
        printf("Error: read\n");
        return -1;
    }
    printf("from client: %s\n", buf);

    // Send the message to client
    write(new_client_socket, msg, strlen(msg));
    printf("To client : hey server this side\n");

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

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

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 :
```

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 :

```
#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.\n");
    int x;
    x = fork();

    if (x > 0)
        printf("IN PARENT PROCESS\nMY PROCESS ID: 5028\n");

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

        if (x > 0) {
            printf("CHILD PROCESS ID : %d\nNEW PARENT PROCESS ID : 1208\n");
        }

        while(1)
            sleep(1);

        printf("IN CHILD PROCESS\nMY PARENT PROCESS ID : 5029\n");
    }

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

return 0;
```

```
hemantsingh@hemantsingh: ~/Desktop
File Edit View Search Terminal Help
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
```