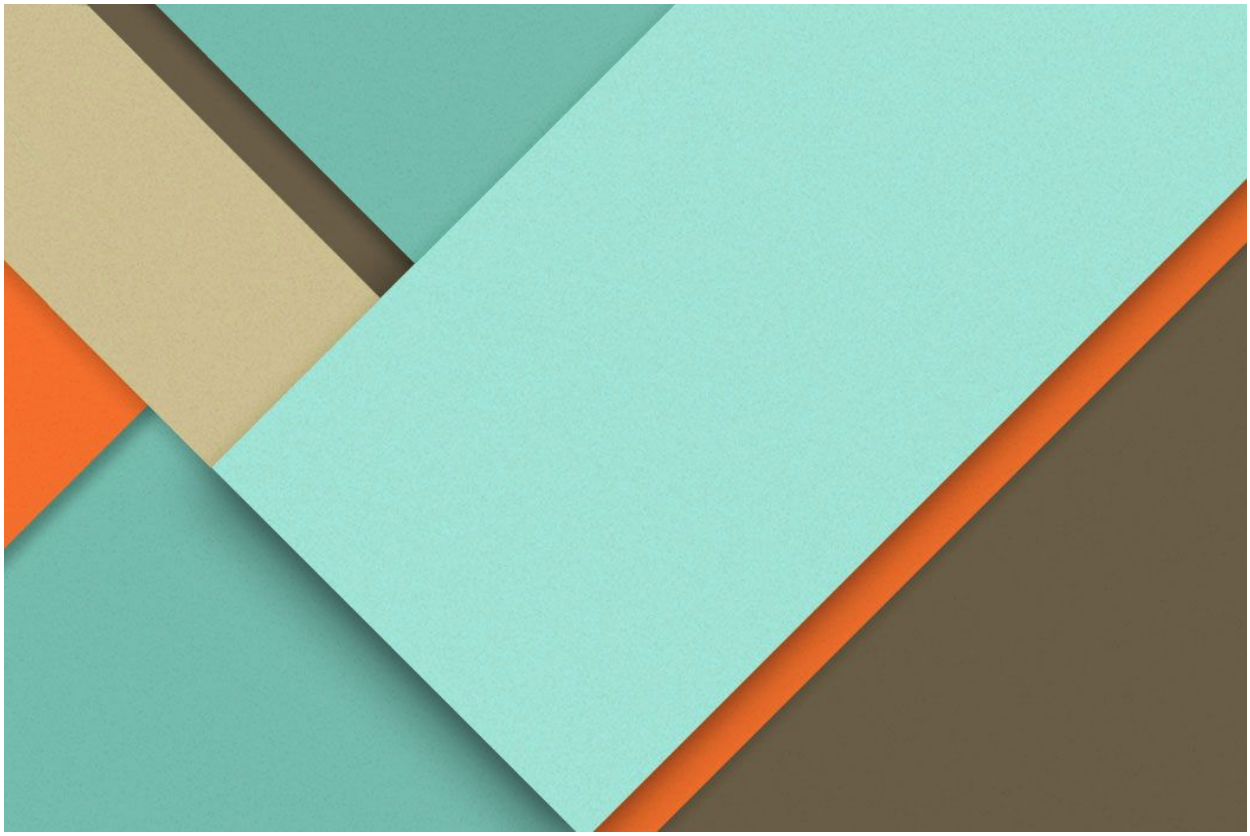


# CSN-361: Assignment 2

01.08.2019

Roodram Paneri
17114066
CSE, 3rd year

# Problem Statements

## I. 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.

## II. Problem Statement 2

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

# Algorithms

## Question 1

Socket Programming

Functions Used:

**1. Client side**

- Socket : creates a socket
- Inet_pton : convert internet address (IP) from numeric to binary form
- Connect : connects client to server through socket

**2. Server side**

- Socket : creates a socket
- Setsocketopt : sets up the attributes of the socket
- Bind : binds the socket to port
- Listen : opens the socket and server is ready to accept clients through the port.
- Accept : accepts a clients connection on the socket
- Read : accepts messages from the client.

## Question 2

- **Zombie process** = It has finished the execution but still has to report to its parent process, as the parent could not accept its return signal.
- **Orphan process =** Orphan process' parent process no more exists i.e. it has finished or terminated without waiting for the child to terminate.

What we do is that we create a child of the parent process and create a grandchild by forking the child as well. Now, we put the child on sleep and terminate the grandchild, so the child doesn't accept the grandchild's return signal and the grandchild becomes a zombie.

Also, we terminate the parent process before the child, this makes the child an orphan process as its parent has terminated before it.

# Implementation and ScreenShots

## 1. Question 1

### Ques1_client.c

```
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#define PORT 8080

int main(int argc, char const *argv[]) {

    int socket_fd = 0;
    char buffer[1024] = {0};

    if ((socket_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0){
        printf("Error while creating socket \n\n");
        return -1;
    }
    else{
        printf("Connected to socket \n\n");
    }

    struct sockaddr_in serverAddress;
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_port = htons(PORT);
```

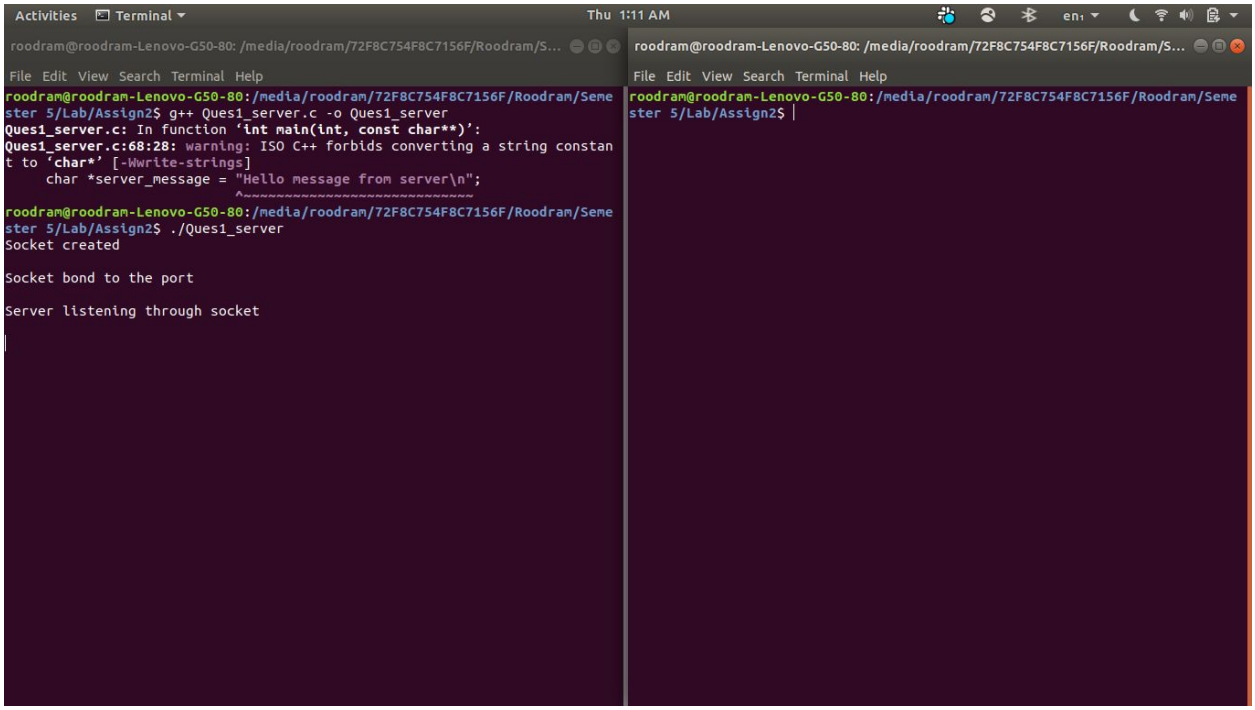
```
    if(inet_pton(AF_INET, "127.0.0.1",
&serverAddress.sin_addr)<=0)  {
        printf("Address is either not supported or invalid
\n\n");
        return -1;
    }
    else{
        printf("Valid IP address\n\n");
    }

    if (connect(socket_fd, (struct sockaddr *)&serverAddress,
sizeof(serverAddress)) < 0) {
        printf("Error while connecting to server through socket
\n\n");
        return -1;
    }
    else{
        printf("Server connected through socket\n\n");
    }

    char *client_message = "Hello message from client\n";
    send(socket_fd , client_message , strlen(client_message) , 0
);
    printf("Hello message sent to server\n\n");

    int readVal;
    readVal = read( socket_fd , buffer, 1024);
    printf("%s\n",buffer );
    return 0;
}
```

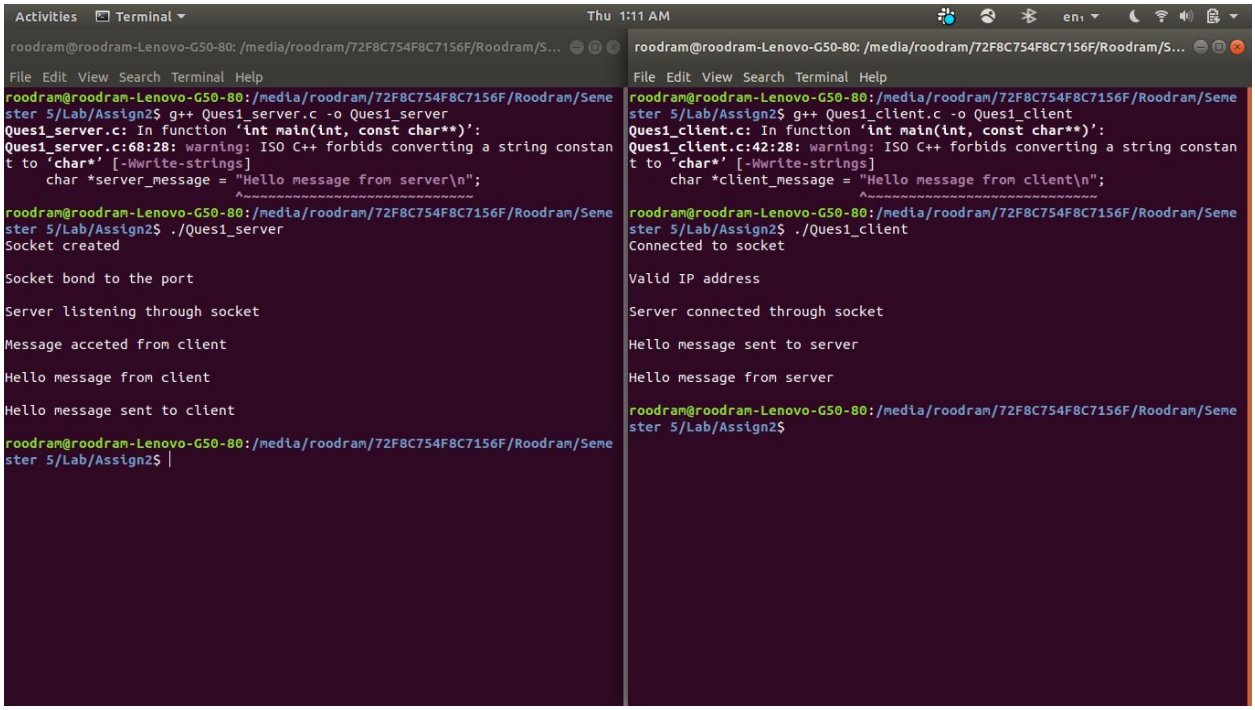
## Ques1_server.c

```
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#define PORT 8080

int main(int argc, char const *argv[]) {
    int fd_server;
    char buffer[1024] = {0};

    // Creating socket file descriptor
    if ((fd_server = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        perror("socket creation failed\n\n");
        exit(EXIT_FAILURE);
    }
    else{
        printf("Socket created\n\n");
    }

    int option = 1;
    if (setsockopt(fd_server, SOL_SOCKET, SO_REUSEADDR |
SO_REUSEPORT, &option, sizeof(option))) {
        perror("setsockopt error\n\n");
        exit(EXIT_FAILURE);
    }
```

```
    struct sockaddr_in socket_address;
    int addressLength = sizeof(socket_address);

    socket_address.sin_family = AF_INET;
    socket_address.sin_addr.s_addr = INADDR_ANY;
    socket_address.sin_port = htons( PORT );

    // Forcefully attaching socket to the port 8080
    if (bind(fd_server, (struct sockaddr *)&socket_address,
sizeof(socket_address))<0) {
        perror("binding of socket to port failed\n\n");
        exit(EXIT_FAILURE);
    }
    else{
        printf("Socket bond to the port\n\n");
    }


    if (listen(fd_server, 3) < 0) {
        perror("Listening failed\n\n");
        exit(EXIT_FAILURE);
    }
    else{
        printf("Server listening through socket\n\n");
    }

    int socket_server;
    if ((socket_server = accept(fd_server, (struct sockaddr
*)&socket_address,(socklen_t*)&addressLength))<0) {
        perror("acceptance failed\n\n");
        exit(EXIT_FAILURE);
```

```
    }
    else{
        printf("Message acceted from client\n\n");
    }

    int readVal;
    readVal = read( socket_server , buffer, 1024);
    printf("%s\n",buffer );

    char *server_message = "Hello message from server\n";
    send(socket_server , server_message , strlen(server_message)
, 0 );
    printf("Hello message sent to client\n\n");
    return 0;
}
```


Activities
Terminal
Thu 1:11 AM
roodram@roodram-Lenovo-G50-80: /media/roodram/72F8C754F8C7156F/Roodram/S...
File Edit View Search Terminal Help
roodram@roodram-Lenovo-G50-80:/media/roodram/72F8C754F8C7156F/Roodram/Semester 5/Lab/Assign2$ g++ Ques1_server.c -o Ques1_server
Ques1_server.c: In function 'int main(int, const char**)':
Ques1_server.c:68:28: warning: ISO C++ forbids converting a string constant to 'char*' [-Wwrite-strings]
char *server_message = "Hello message from server\n";
^~~~~~~~~~~~~~~~~~~~~~~~~~~~~
roodram@roodram-Lenovo-G50-80:/media/roodram/72F8C754F8C7156F/Roodram/Semester 5/Lab/Assign2$ ./Ques1_server
Socket created
Socket bond to the port
Server listening through socket
roodram@roodram-Lenovo-G50-80: /media/roodram/72F8C754F8C7156F/Roodram/S...
File Edit View Search Terminal Help
roodram@roodram-Lenovo-G50-80:/media/roodram/72F8C754F8C7156F/Roodram/Semester 5/Lab/Assign2$


Activities
Terminal
Thu 1:11 AM
roodram@roodram-Lenovo-G50-80: /media/roodram/72F8C754F8C7156F/Roodram/S...
File Edit View Search Terminal Help
roodram@roodram-Lenovo-G50-80:/media/roodram/72F8C754F8C7156F/Roodram/Semester 5/Lab/Assign2$ g++ Ques1_server.c -o Ques1_server
Ques1_server.c: In function 'int main(int, const char**)':
Ques1_server.c:68:28: warning: ISO C++ forbids converting a string constant to 'char*' [-Wwrite-strings]
char *server_message = "Hello message from server\n";
^~~~~~~~~~~~~~~~~~~~~~~~~~~~~
roodram@roodram-Lenovo-G50-80:/media/roodram/72F8C754F8C7156F/Roodram/Semester 5/Lab/Assign2$ ./Ques1_server
Socket created
Socket bond to the port
Server listening through socket
Message acceted from client
Hello message from client
Hello message sent to client
roodram@roodram-Lenovo-G50-80:/media/roodram/72F8C754F8C7156F/Roodram/Semester 5/Lab/Assign2$
roodram@roodram-Lenovo-G50-80: /media/roodram/72F8C754F8C7156F/Roodram/S...
File Edit View Search Terminal Help
roodram@roodram-Lenovo-G50-80:/media/roodram/72F8C754F8C7156F/Roodram/Semester 5/Lab/Assign2$ g++ Ques1_client.c -o Ques1_client
Ques1_client.c: In function 'int main(int, const char**)':
Ques1_client.c:42:28: warning: ISO C++ forbids converting a string constant to 'char*' [-Wwrite-strings]
char *client_message = "Hello message from client\n";
^~~~~~~~~~~~~~~~~~~~~~~~~~~~~
roodram@roodram-Lenovo-G50-80:/media/roodram/72F8C754F8C7156F/Roodram/Semester 5/Lab/Assign2$ ./Ques1_client
Connected to socket
Valid IP address
Server connected through socket
Hello message sent to server
Hello message from server
roodram@roodram-Lenovo-G50-80:/media/roodram/72F8C754F8C7156F/Roodram/Semester 5/Lab/Assign2$

## 2. Question 2

### Ques2_zombie_orphan.c

```
#include <bits/stdc++.h>
#include <unistd.h>
#include <stdio.h>
int main()
{
 printf("Parent process pid = %d \n", getpid());

    if (fork() == 0)
    {
        printf("Child process id = %d\nChild of parent process id
= %d \n", getpid(), getppid());

        printf("Creating a grandchild to demonstrate zombie
process\n\n");
        if (fork() > 0) {
            printf("(Grandchild) Parent process pid = %d \n",
getpid());
            printf("(Grandchild) Parent goes to sleep\n\n");
            sleep(8);
            printf("(Grandchild) Parent resumes but child
terminnated while it was in sleep\nSo GrandChild is a zombie\n");
        }
        else{
            sleep(3);
            printf("GrandChild process id = %d\nChild of parent
process id = %d \n", getpid(), getppid());
```

```
            printf("GrandChild exits\n\n");
            exit(0);
        }


        sleep(5);
        printf("Now child is an Orphan \n");
        printf("Child process id = %d\nChild of parent process id
= %d", getpid(), getppid());
    }
    sleep(8);
    return 0;


}
```