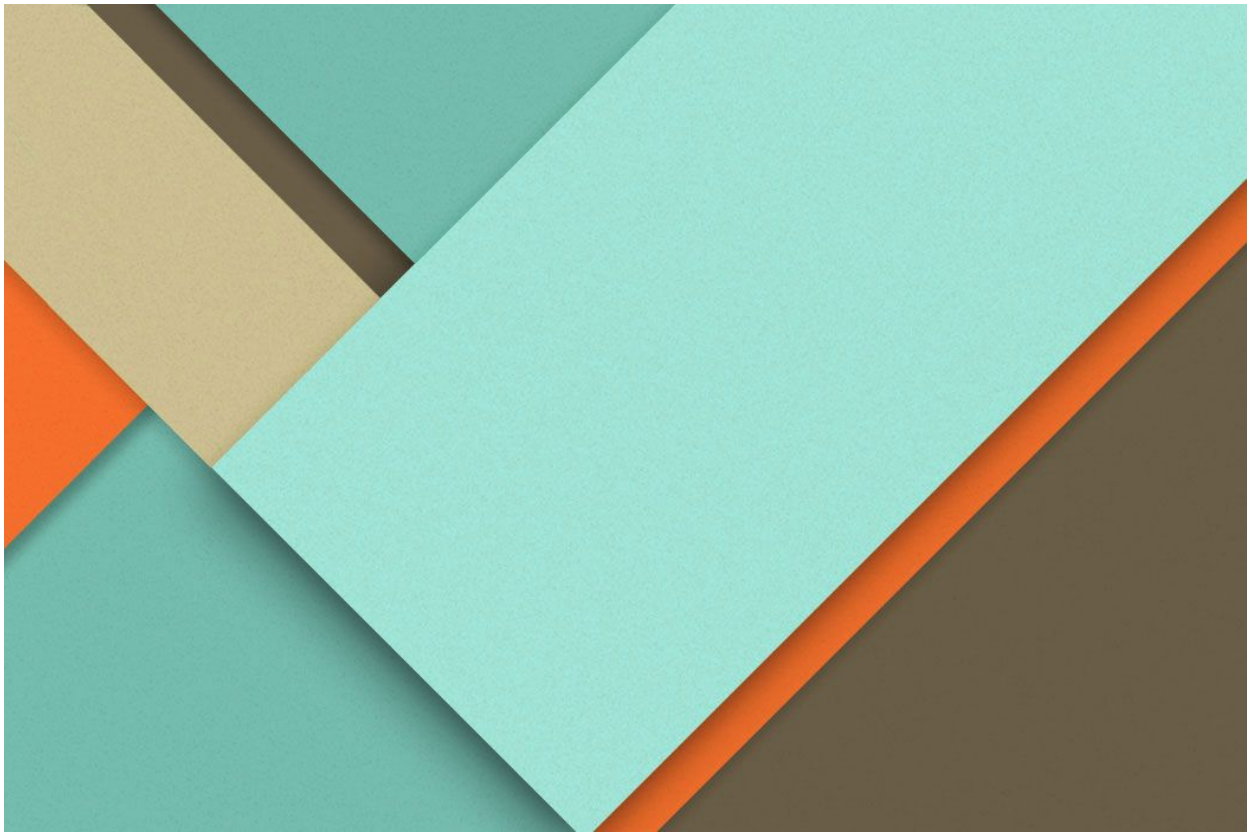


Computer Network Lab(CSN-361)

Assignment - 2

31.07.2019

Shivansh Bindal
17115088
Computer Science & Engineering
3rd yr

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.

Code

Server

```
#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 server_fd, new_socket, valread;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};
    char *hello = "Message from server to client";
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
    {
```

```
    perror("socket failed");
    exit(EXIT_FAILURE);
}

if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
                                                &opt, sizeof(opt)))
{
    perror("setsockopt");
    exit(EXIT_FAILURE);
}
address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons( PORT );

if (bind(server_fd, (struct sockaddr *)&address,
                             sizeof(address))<0)
{
    perror("bind failed");
    exit(EXIT_FAILURE);
}
if (listen(server_fd, 3) < 0)
{
    perror("listen");
    exit(EXIT_FAILURE);
}
if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                    (socklen_t*)&addrlen))<0)
{
    perror("accept");
```

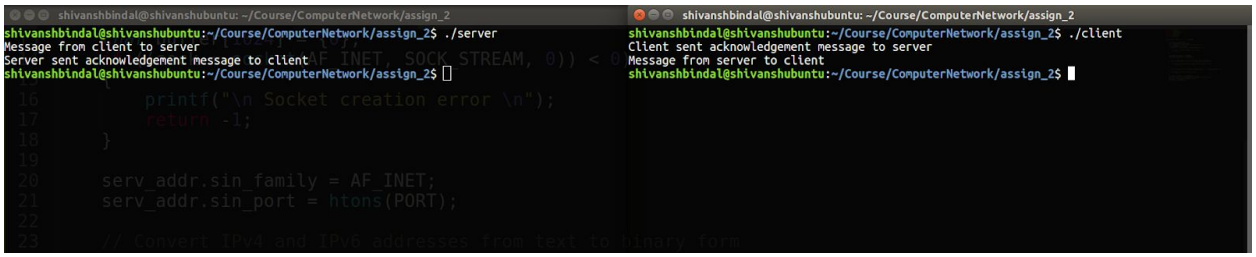
```
        exit(EXIT_FAILURE);
    }
    valread = read( new_socket , buffer, 1024);
    printf("%s\n",buffer );
    send(new_socket , hello , strlen(hello) , 0 );
    printf("Server sent acknowledgement message to client\n");
    return 0;
}
```

Client

```
#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 sock = 0, valread;
    struct sockaddr_in serv_addr;
    char *hello = "Message from client to server";
    char buffer[1024] = {0};
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("\n Socket creation error \n");
        return -1;
    }
```

```
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    // Convert IPv4 and IPv6 addresses from text to binary form
    if(inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr)<=0)
    {
        printf("\nInvalid address/ Address not supported \n");
        return -1;
    }

    if (connect(sock, (struct sockaddr *)&serv_addr,
sizeof(serv_addr)) < 0)
    {
        printf("\nConnection Failed \n");
        return -1;
    }
    send(sock , hello , strlen(hello) , 0 );
    printf("Client sent acknowledgement message to server\n");
    valread = read( sock , buffer, 1024);
    printf("%s\n",buffer );
    return 0;
}
```

Screenshot of running code



Problem Statement 2:

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

Code

Zombie Process

```
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <bits/stdc++.h>
using namespace std;
int main()
{
    // Zombie process
    pid_t child_pid = fork();

    // Parent process
    if (child_pid > 0)
    {
        sleep(1);
```

```
        printf("in parent process (Zombie Process)\n");
    }

    // Child process
    else
    {
        printf("in child process (Zombie Process)\n");
        exit(0);
    }
    return 0;
}
```

Orphan Process

```
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <bits/stdc++.h>
using namespace std;
int main()
{
// Orphan Process
    int pid = fork();

    if (pid > 0)
        printf("in parent process (Orphan Process)\n");

    else if (pid == 0)
    {
```

```
        sleep(1);
        printf("in child process (Orphan Process)\n");
    }

    sleep(2);
    return 0;
}
```

Screenshot of running code