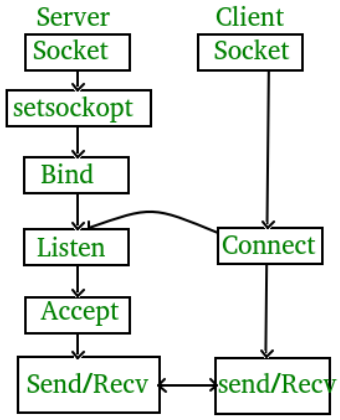
IT 694 - CN Date: 25/01/2022

Assignment : 0

List of Topics: Socket Programming

Socket Programming:

Socket programming is a way of connecting two nodes on a network to communicate with each other. Server forms the listener socket while the client reaches out to the server.For the TCP and UDP connection, socket binding process works differently.



Basic functions:

Stages for server

1. **Socket creation:** By using the below function, you are able to make the socket.

   ```
   int sockfd = socket(domain, type, protocol);
   ```

2. **Socket Binding:** This function helps to assign a unique address to the socket.

   ```
   int bind(int sockfd, const struct sockaddr *addr, socklen_t
   addrlen);
   ```

3. **Listen:** In this mode socket is ready to accept the client. In this mode the client approaches the server to make a connection.

```
int listen(int sockfd, int backlog);
```

4. **Accept:** Extract the first connection from the listening queue.Create new connected socket and return new file descriptor to the referred socket.

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

Stages for client

1. **Connect:** Helps to connect the specific socket to establish connection.

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

Reference:

https://www.geeksforgeeks.org/socket-in-computer-network/

https://www.geeksforgeeks.org/socket-programming-cc/

Example of socket programming

Client code

```
// Client side C/C++ program to demonstrate Socket programming
#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 *exit_msg = "exit", *msg;
    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;
    }

    while(1) {
        scanf("%s", msg);
        if(!strcmp(msg, exit_msg)){
            close(sock);
            return 0;
        }
        send(sock , msg , strlen(msg) , 0 );
        valread = read( sock , buffer, 1024);
        printf("%s\n",buffer );
    }
    return 0;
}
```

Server code

```
// Server side C/C++ program to demonstrate Socket programming
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
```

```c
#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];
    char *hello = "Hello from server";

    // Creating socket file descriptor
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
    {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    // Forcefully attaching socket to the port 8080 - For address
reuse
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR |
SO_REUSEPORT,
                                                &opt, sizeof(opt)))
```

```
    {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }
    address.sin_family = AF_INET; // match the socket() call
    address.sin_addr.s_addr = INADDR_ANY; // bind to any local
address
    address.sin_port = htons( PORT ); // specify port to listen on


    // Forcefully attaching socket to the port 8080
    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);
    }

    while(1) {
        valread = read( new_socket , buffer, 1024);
        printf("Client: %s\n",buffer );
        memset(buffer, 0, 1024);
        send(new_socket , hello , strlen(hello) , 0 );
        printf("Hello message sent\n");
    }

    return 0;
}
```

How to run?

1. Open a terminal
2. Type “gcc server.c -o server”, and press enter.
3. Type “./server”, and press enter.
4. Open 2nd terminal
5. Type “gcc client.c -o client”, and press enter.
6. Type “./client”, and press enter.
7. Both the server and client are running simultaneously.

8. Note: **Always** run the server first.

Exercise:

1. Create server and client using socket programming. Make them communicate with each other by sending packets between them.

Submission:

1. Submit assignment with the report consists of an input file and output file with proper explanation of each output of all the exercises in pdf format.
2. Add all the outputs and a brief description of the commands used in the given demo scripts in the report.
3. Submitted code in a report should be well commented.
4. Submit a zip file with your student id which will consist of the folder for each script & respective outputs. one common report in a parent directory.

   Eg:

   group-id-2021***.zip

   - report.pdf
   - PR1
     - Script
     - Its respective output
   - PR2
     - Script
     - Its respective output

5. Submission Deadline: 30/01/2022(Sunday), 23:59:00
6. Penalties will be imposed for the late submission.