**Data Components Laboratory**

**exercise 1**

**socket programming in c**

**1. Echo client server (one server and multiple clients) in C**

**Algorithm:**

**SERVER SIDE:**

* **Step 1:** Create a stream socket with internet domain using socket () system call which returns a socket descriptor.
* **Step 2:** Create a structure that holds the acceptable IP address, address family and port number to be connected. Assign the address, family and port to the structure.
* **Step 3:** Bind the address structure to the socket created using bind () system call.
* **Step 4:** Call the listen () function by passing the socket descriptor and length of the queue as arguments. The server starts listening for connection from client.
* **Step 5:** The accept () system call is used to accept the request from the client. It returns the socket descriptor of client.
* **Step 6:** The server responds with the same string that is given from the user.
* **Step 7:** There will be a single server ready to respond the client. Multiple clients can request to the server.

**CLIENT SIDE:**

* **Step 1**: Create a stream socket with internet domain using socket () system call which returns a socket descriptor.
* **Step 2:** Create a structure that holds the client’s IP address, address family and port number to be connected. Assign the address, family and port to the structure.
* **Step 3:** Create a connection with the server using connect () system call that takes the socket descriptor, address structure and size of the structure as arguments. The port address is got from the user.
* **Step 4:** Get the input from the user and send it to the server using write () system call.
* **Step 5:** Use read () system call to get the string from the server. Print the string to the user. The server responds whatever client sent and this is called as echo.

**Program:**

**SERVER SIDE:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#define PORT 4444

int main(){

    int sockfd, ret;

     struct sockaddr\_in serverAddr;

    int newSocket;

    struct sockaddr\_in newAddr;

    socklen\_t addr\_size;

    char buffer[1024];

    pid\_t childpid;

    sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

    if(sockfd < 0){

        printf("[-]Error in connection.\n");

        exit(1);

    }

    printf("[+]Server Socket is created.\n");

    memset(&serverAddr, '\0', sizeof(serverAddr));

    serverAddr.sin\_family = AF\_INET;

    serverAddr.sin\_port = htons(PORT);

    serverAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

    ret = bind(sockfd, (struct sockaddr\*)&serverAddr, sizeof(serverAddr));

    if(ret < 0){

        printf("[-]Error in binding.\n");

        exit(1);

    }

    printf("[+]Bind to port %d\n", 4444);

    if(listen(sockfd, 10) == 0){

        printf("[+]Listening....\n");

    }else{

        printf("[-]Error in binding.\n");

    }

    while(1){

        newSocket = accept(sockfd, (struct sockaddr\*)&newAddr, &addr\_size);

        if(newSocket < 0){

            exit(1);

        }

        printf("Connection accepted from %s:%d\n", inet\_ntoa(newAddr.sin\_addr), ntohs(newAddr.sin\_port));

        if((childpid = fork()) == 0){

            close(sockfd);

            while(1){

                recv(newSocket, buffer, 1024, 0);

                if(strcmp(buffer, ":exit") == 0){

                    printf("Disconnected from %s:%d\n", inet\_ntoa(newAddr.sin\_addr), ntohs(newAddr.sin\_port));

                    break;

                }else{

                    printf("Client: %s\n", buffer);

                    send(newSocket, buffer, strlen(buffer), 0);

                    bzero(buffer, sizeof(buffer));

                }

            }

        }

    }

    close(newSocket);

    return 0;

}

**CLIENT SIDE:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#define PORT 4444

int main(){

    int clientSocket, ret;

    struct sockaddr\_in serverAddr;

    char buffer[1024];

    clientSocket = socket(AF\_INET, SOCK\_STREAM, 0);

    if(clientSocket < 0){

        printf("[-]Error in connection.\n");

        exit(1);

    }

    printf("[+]Client Socket is created.\n");

    memset(&serverAddr, '\0', sizeof(serverAddr));

    serverAddr.sin\_family = AF\_INET;

    serverAddr.sin\_port = htons(PORT);

    serverAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

    ret = connect(clientSocket, (struct sockaddr\*)&serverAddr, sizeof(serverAddr));

    if(ret < 0){

        printf("[-]Error in connection.\n");

        exit(1);

    }

    printf("[+]Connected to Server.\n");

    while(1){

        printf("Client: \t");

        scanf("%s", &buffer[0]);

        send(clientSocket, buffer, strlen(buffer), 0);

        if(strcmp(buffer, ":exit") == 0){

            close(clientSocket);

            printf("[-]Disconnected from server.\n");

            exit(1);

        }

        if(recv(clientSocket, buffer, 1024, 0) < 0){

            printf("[-]Error in receiving data.\n");

        }else{

            printf("Server: \t%s\n", buffer);

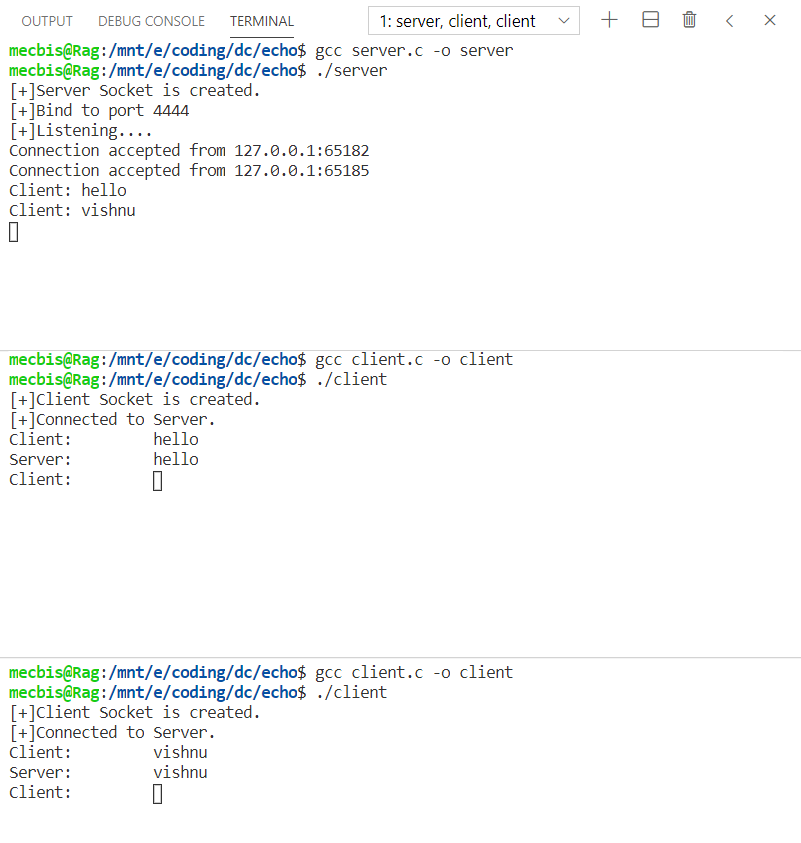
        }

    }

    return 0;

}

**Output:**

****

**2. Chat Application (multiple servers and multiple clients) in C**

**Algorithm:**

**SERVER SIDE:**

* **Step 1:** Create a stream socket with internet domain using socket () system call which returns a socket descriptor.
* **Step 2:** Create a structure that holds the acceptable IP address, address family and port number to be connected. Assign the address, family and port to the structure.
* **Step 3:** Bind the address structure to the socket created using bind () system call.
* **Step 4:** Call the listen () function by passing the socket descriptor and length of the queue as arguments. The server starts listening for connection from client.
* **Step 5:** The accept () system call is used to accept the request from the client. It returns the socket descriptor of client.
* **Step 6:** The server responds with the same string that is given from the user.
* **Step 7:** There will be multiple number of servers ready to respond the client. The client can request for a particular server.

**CLIENT SIDE:**

* **Step 1**: Create a stream socket with internet domain using socket () system call which returns a socket descriptor.
* **Step 2:** Create a structure that holds the client’s IP address, address family and port number to be connected. Assign the address, family and port to the structure.
* **Step 3:** Create a connection with the server using connect () system call that takes the socket descriptor, address structure and size of the structure as arguments. The port address is got from the user.
* **Step 4:** Get the input from the user and send it to the server using write () system call.
* **Step 5:** Use read () system call to get the string from the server. Print the string to the user.

**Program:**

**SERVER SIDE:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#define PORT 4444

int main(){

    int sockfd, ret;

     struct sockaddr\_in serverAddr;

    int newSocket;

    struct sockaddr\_in newAddr;

    socklen\_t addr\_size;

    char buffer[1024];

    pid\_t childpid;

    sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

    if(sockfd < 0){

        printf("[-]Error in connection.\n");

        exit(1);

    }

    printf("[+]Server Socket is created.\n");

    memset(&serverAddr, '\0', sizeof(serverAddr));

    serverAddr.sin\_family = AF\_INET;

    serverAddr.sin\_port = htons(PORT);

    serverAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

    ret = bind(sockfd, (struct sockaddr\*)&serverAddr, sizeof(serverAddr));

    if(ret < 0){

        printf("[-]Error in binding.\n");

        exit(1);

    }

    printf("[+]Bind to port %d\n", 4444);

    if(listen(sockfd, 10) == 0){

        printf("[+]Listening....\n");

    }else{

        printf("[-]Error in binding.\n");

    }

    while(1){

        newSocket = accept(sockfd, (struct sockaddr\*)&newAddr, &addr\_size);

        if(newSocket < 0){

            exit(1);

        }

        printf("Connection accepted from %s:%d\n", inet\_ntoa(newAddr.sin\_addr), ntohs(newAddr.sin\_port));

        if((childpid = fork()) == 0){

            close(sockfd);

            while(1){

                recv(newSocket, buffer, 1024, 0);

                if(strcmp(buffer, ":exit") == 0){

                    printf("Disconnected from %s:%d\n", inet\_ntoa(newAddr.sin\_addr), ntohs(newAddr.sin\_port));

                    break;

                }else{

                    printf("Client: %s\n", buffer);

                    send(newSocket, buffer, strlen(buffer), 0);

                    bzero(buffer, sizeof(buffer));

                }

            }

        }

    }

    close(newSocket);

    return 0;

}

**CLIENT SIDE:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

int PORT;

int main(){

    int clientSocket, ret;

    struct sockaddr\_in serverAddr;

    char buffer[1024];

    printf("Enter the port number to connect: ");

    scanf("%d",&PORT);

    clientSocket = socket(AF\_INET, SOCK\_STREAM, 0);

    if(clientSocket < 0){

        printf("[-]Error in connection.\n");

        exit(1);

    }

    printf("[+]Client Socket is created.\n");

    memset(&serverAddr, '\0', sizeof(serverAddr));

    serverAddr.sin\_family = AF\_INET;

    serverAddr.sin\_port = htons(PORT);

    serverAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

    ret = connect(clientSocket, (struct sockaddr\*)&serverAddr, sizeof(serverAddr));

    if(ret < 0){

        printf("[-]Error in connection.\n");

        exit(1);}

    printf("[+]Connected to Server.\n");

    while(1){

        printf("Client: \t");

        scanf("%s", &buffer[0]);

        send(clientSocket, buffer, strlen(buffer), 0);

        if(strcmp(buffer, ":exit") == 0){

            close(clientSocket);

            printf("[-]Disconnected from server.\n");

            exit(1);}

        if(recv(clientSocket, buffer, 1024, 0) < 0){

            printf("[-]Error in receiving data.\n");

        }else{

            printf("Server: \t%s\n", buffer);

         }

}

return 0;

}

**output:**

****

**3. Client Server Calculator**

**SERVER SIDE:**

* **Step 1:** Create a stream socket with internet domain using socket () system call which returns a socket descriptor.
* **Step 2:** Create a structure that holds the acceptable IP address, address family and port number to be connected. Assign the address, family and port to the structure.
* **Step 3:** Bind the address structure to the socket created using bind () system call.
* **Step 4:** Call the listen () function by passing the socket descriptor and length of the queue as arguments. The server starts listening for connection from client.
* **Step 5:** The accept () system call is used to accept the request from the client. It returns the socket descriptor of client. Using the descriptor read the operator and operands using read ().
* **Step 6:** Calculate the answer and store the result in a variable.
* **Step 7:** Send the result using write () system call.

**CLIENT SIDE:**

* **Step 1**: Create a stream socket with internet domain using socket () system call which returns a socket descriptor.
* **Step 2:** Create a structure that holds the client’s IP address, address family and port number to be connected. Assign the address, family and port to the structure.
* **Step 3:** Create a connection with the server using connect () system call that takes the socket descriptor, address structure and size of the structure as arguments.
* **Step 4:** Get the operands and operator from the user and send it to the server using write () system call.
* **Step 5:** Use read () system call to get the answer from the server. Print the result to the user.

**Program:**

**SERVER SIDE:**

#include<sys/types.h>

#include<sys/socket.h>

#include<stdio.h>

#include<netinet/in.h>

#include<unistd.h>

#include<string.h>

#include<arpa/inet.h>

void main()

{

        int sockfd,connfd,sin\_size;

        char operator;

        int op1,op2,result;

        if((sockfd=socket(AF\_INET,SOCK\_STREAM,0))>0)

        {

                printf("Socket created\n");

        }

        struct sockaddr\_in servaddr;

        struct sockaddr\_in clientaddr;

        servaddr.sin\_family=AF\_INET;

        servaddr.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

        servaddr.sin\_port=htons(6006);

        if((bind(sockfd, (struct sockaddr \*)&servaddr,sizeof(servaddr)))==0)

        {

                printf("Bind successful\n");

        }

        if((listen(sockfd,5))==0)

        {

                printf("Server listening\n");

        }

        sin\_size=sizeof(struct sockaddr\_in);

        if((connfd=accept(sockfd, (struct sockaddr \*)&clientaddr,&sin\_size))>0)

        {

                printf("Client accepted\n");

        }

        read(connfd,&operator,10);

        read(connfd,&op1,sizeof(op1));

        read(connfd,&op2,sizeof(op2));

        if(operator=='+')

        {

                result=op1 + op2;

        }

        else if(operator=='-')

        {

                result=op1 - op2;

        }

        else if(operator=='\*')

        {

                result=op1 \* op2;

        }

        else if(operator=='/')

        {

                result=op1 / op2;

        }

        else

        {

                printf("Invalid operator\n");

                result=-1;

        }

        write(connfd,&result,sizeof(result));

        close(sockfd);

}

**CLIENT SIDE:**

#include<sys/types.h>

#include<sys/socket.h>

#include<stdio.h>

#include<netinet/in.h>

#include<unistd.h>

#include<string.h>

#include<strings.h>

#include<arpa/inet.h>

void main()

{

        int sockfd,sin\_size,con;

        char operat;

        int op1,op2,result;

        if((sockfd=socket(AF\_INET,SOCK\_STREAM,0))>0)

        {

                printf("Socket created!\n");

        }

        struct sockaddr\_in servaddr;

        servaddr.sin\_family=AF\_INET;

        servaddr.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

        servaddr.sin\_port=htons(6006);

        sin\_size=sizeof(struct sockaddr\_in);

        if((con=connect(sockfd,(struct sockaddr \*) &servaddr, sin\_size))==0)

        {

                printf("Connected to server\n");

        }

        printf("Enter operands and operation to perform\n");

        scanf("%d %d %c",&op1,&op2,&operat);

        write(sockfd,&operat,10);

        write(sockfd,&op1,sizeof(op1));

        write(sockfd,&op2,sizeof(op2));

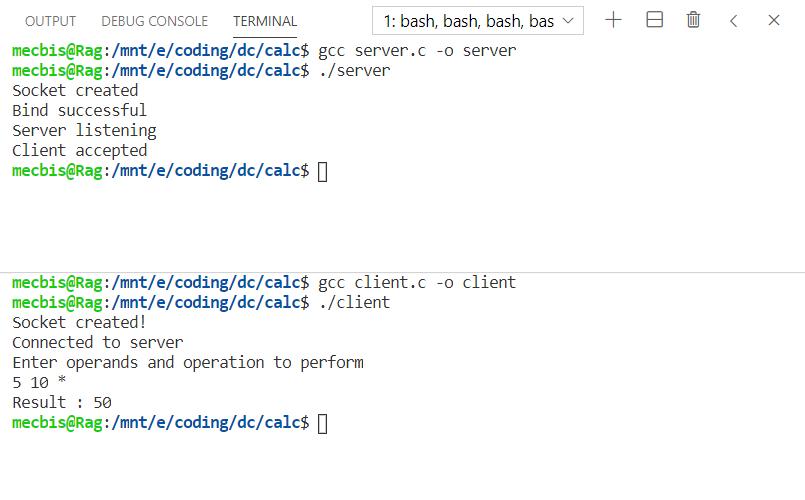
        read(sockfd,&result,sizeof(result));

        printf("Result : %d\n",result);

        close(sockfd);

}

**Output:**

****

**4. File Transfer**

**Algorithm:**

* **Step 1:** The server starts and waits for filename.
* **Step 2:** The client sends a filename.
* **Step 3:** The server receives filename. If file is present, server starts reading file

and continues to send a buffer filled with file contents encrypted until file-end is reached.

* **Step 4:** End is marked by EOF.
* **Step 5:** File is received as buffers until EOF is
* received. Then it is decrypted.
* **Step 6:** If Not present, a file not found is sent.

**Program:**

**SERVER SIDE:**

#include<unistd.h>

#include<stdio.h>

#include<sys/socket.h>

#include<stdlib.h>

#include<netinet/in.h>

#include<string.h>

#define PORT 4444

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 buf[1024]={0};

        if((server\_fd=socket(AF\_INET, SOCK\_STREAM, 0))==0)

        {

                perror("Socket failed");

                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);

        }

        FILE \*fp;

        read(new\_socket,buf,100);

        fp=fopen("servermessage.txt","w");

        fprintf(fp,"%s",buf);

        printf("File recieved\n");

        return 0;

}

**CLIENT SIDE:**

#include<stdio.h>

#include<sys/socket.h>

#include<arpa/inet.h>

#include<unistd.h>

#include<string.h>

#define PORT 4444

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

{

        int sock=0, valread;

        struct sockaddr\_in serv\_addr;

        char buf[1024]={0};

        if((sock=socket(AF\_INET, SOCK\_STREAM, 0))<0)

        {

                printf("\nSocket creation error\n");

                return -1;

        }

        serv\_addr.sin\_family = AF\_INET;

        serv\_addr.sin\_port = htons(PORT);

        if(inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr)<=0)

        {

                printf("\nInvalid address\n");

                return -1;

        }

        if(connect(sock, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr))<0)

        {

                printf("\nConnection failed\n");

                return -1;

        }

        FILE \*f;

        f=fopen("clientmessage.txt","r");

        fscanf(f,"%s",buf);

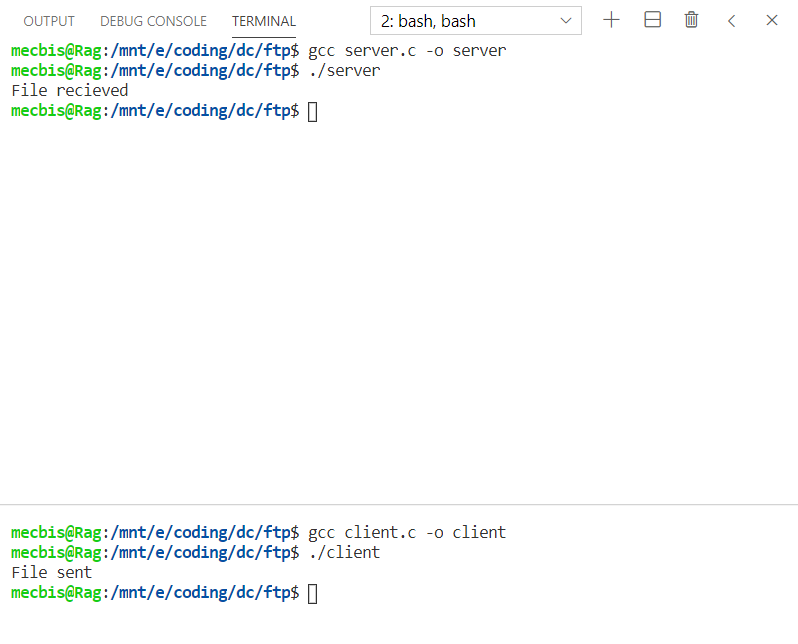
        write(sock,buf,100);

        printf("File sent\n");

        return 0;

}

**Output:**

****

**Result:**

The socket programming is done successfully using the C Programming language.