

# EXPERIMENT-1

## SOCKET COMMUNICATION USING TCP

PRIYABRAT ROUTRAY

Reg:-RA1911027010078

### CLIENT code:-

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
void chat (int sock)
{
    char item[1000];
    int n;
    while (1)
    {
        bzero (item, 1000);
        printf ("Type your Message: ");
        n = 0;
        while ((item[n++] = getchar ()) != '\n');
        write (sock, item, sizeof (item));
        bzero (item, 1000);
        read (sock, item, sizeof (item));
        printf ("From Server : %s", item);
        if ((strcmp (item, "done", 4)) == 0)
        {
            printf ("Chat has completed\n");
            break;
        }
    }
}

int main ()
{
    int net_socket = socket (AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in server_address;
    server_address.sin_family = AF_INET;
    server_address.sin_port = htons (8080);
    server_address.sin_addr.s_addr = INADDR_ANY;
    int conn =
    connect (net_socket, (struct sockaddr *) &server_address,
    sizeof (server_address));
    if (conn == -1)
    printf ("Connect failed\n");
    else
    printf ("Connected\n");
    chat (net_socket);
    close (net_socket);
    return 0;
}
```

## SERVER code:-

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <string.h>
#include <unistd.h>
void chat (int sock)
{
    char item[1000];
    int n;
    while (1)
    {
        bzero (item, 1000);
        read (sock, item, sizeof (item));
        printf ("From client: %s \n To client : ", item);
        bzero (item, 1000);
        n = 0;
        while ((item[n++] = getchar ()) != '\n');
        write (sock, item, sizeof (item));
        if (strncmp ("done", item, 4) == 0)
        {
            printf ("Chat has been completed\n");
            break;
        }
    }
}

int main ()
{
    int net_socket = socket (AF_INET, SOCK_STREAM, 0);
    int client_socket;
    struct sockaddr_in server_address, cli;
    server_address.sin_family = AF_INET;
    server_address.sin_port = htons (8080);
    server_address.sin_addr.s_addr = htons (INADDR_ANY);
    int conn = bind (net_socket, (struct sockaddr *) &server_address,
    sizeof (server_address));
    if (conn == -1)
        printf ("Bind failed\n");
    else
        printf ("Bind\n");
    if ((listen (net_socket, 5)) != 0)
    {
        printf ("Listening has failed\n");
    }
    else
        printf ("Server is listening\n");
    int len = sizeof (cli);
    client_socket = accept (net_socket, (struct sockaddr *) &cli, &len);
    if (client_socket < 0)
    {
        printf ("Client did not connect\n");
    }
    else
        printf ("Client connected\n");
    chat (client_socket);
    close (net_socket);
}
```

```
return 0;
}
```

## OUTPUT:-

The screenshot displays the AWS Cloud9 IDE interface. The top toolbar includes icons for File, Edit, Find, View, Go, Run, Tools, Window, and Support. The left sidebar shows a file explorer with a project named '18CSC302J Bat' containing several subfolders and files, including 'client.c' and 'server.c'. The main editor area has multiple tabs open: 'bash - "ip-172-31-6-144"', 'server.c', 'client.c', and another 'bash - "ip-172-31-6-144"'. The 'client.c' tab is active, showing the following code:

```
Ameenapriyadarshini:~/environment/RA1911027010078/exp1 $ cc -o client client.c
Ameenapriyadarshini:~/environment/RA1911027010078/exp1 $ ./client
Connected
Type your Message: hii
From Server : hello
Type your Message: hello server i am your client
From Server : yes client i know you and am waiting for your message
Type your Message: done
From Server : done
Chat has completed
Ameenapriyadarshini:~/environment/RA1911027010078/exp1 $
```

Below the editor, there is a 'Run' button and a 'Command' field. The 'Command' field contains the path 'RA1911027010078/New\ Folder\one\_odd\_appearing.c'. The 'Runner' is set to 'C', and the 'CWD' is set to 'ENV'. The output of the command is shown in a terminal window below the command field, displaying '[Idle]'. The bottom status bar indicates 'AWS: (not connected)'.

**RESULT:-** The socket communication using TCP was done successfully.