

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>

#define MAX 80

#define PORT 8086

#define SA struct sockaddr


// Function designed for chat between client and server.
void func(int sockfd)
{
    char buff[MAX];
    int n;
    // infinite loop for chat
    for (;;) {
        bzero(buff, MAX);

        // read the message from client and copy it in buffer
        read(sockfd, buff, sizeof(buff));
        // print buffer which contains the client contents
        printf("From client: %s\t To client : ", buff);
        bzero(buff, MAX);
        n = 0;
        // copy server message in the buffer
        while ((buff[n++] = getchar()) != '\n')
            ;
    }
}
```

```

// and send that buffer to client
write(sockfd, buff, sizeof(buff));

// if msg contains "Exit" then server exit and chat ended.
if (strncmp("exit", buff, 4) == 0) {
    printf("Server Exit...\n");
    break;
}
}
}

```

// Driver function

```

int main()
{
    int sockfd, connfd, len;
    struct sockaddr_in servaddr, cli;

    // socket create and verification
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        printf("socket creation failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully created..\n");
    bzero(&servaddr, sizeof(servaddr));

    // assign IP, PORT
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(PORT);

```

```

// Binding newly created socket to given IP and verification
if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
    printf("socket bind failed...\n");
    exit(0);
}
else
    printf("Socket successfully binded..\n");

// Now server is ready to listen and verification
if ((listen(sockfd, 5)) != 0) {
    printf("Listen failed...\n");
    exit(0);
}
else
    printf("Server listening..\n");
len = sizeof(cli);

// Accept the data packet from client and verification
connfd = accept(sockfd, (SA*)&cli, &len);
if (connfd < 0) {
    printf("server acccept failed...\n");
    exit(0);
}
else
    printf("server acccept the client...\n");

// Function for chatting between client and server
func(connfd);

// After chatting close the socket

```

```
    close(sockfd);  
}
```

CLIENT CODE

```
#include <netdb.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/socket.h>  
#define MAX 80  
#define PORT 8086  
#define SA struct sockaddr  
void func(int sockfd)  
{  
    char buff[MAX];  
    int n;  
    for (;;) {  
        bzero(buff, sizeof(buff));  
        printf("Enter the string : ");  
        n = 0;  
        while ((buff[n++] = getchar()) != '\n')  
            ;  
        write(sockfd, buff, sizeof(buff));  
        bzero(buff, sizeof(buff));  
        read(sockfd, buff, sizeof(buff));  
        printf("From Server : %s", buff);  
        if ((strncmp(buff, "exit", 4)) == 0) {  
            printf("Client Exit...\n");  
            break;  
        }  
    }  
}
```

```
}
```

```
int main()
```

```
{
```

```
    int sockfd, connfd;
```

```
    struct sockaddr_in servaddr, cli;
```

```
    // socket create and varification
```

```
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
```

```
    if (sockfd == -1) {
```

```
        printf("socket creation failed...\n");
```

```
        exit(0);
```

```
    }
```

```
    else
```

```
        printf("Socket successfully created..\n");
```

```
    bzero(&servaddr, sizeof(servaddr));
```

```
    // assign IP, PORT
```

```
    servaddr.sin_family = AF_INET;
```

```
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
```

```
    servaddr.sin_port = htons(PORT);
```

```
    // connect the client socket to server socket
```

```
    if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0) {
```

```
        printf("connection with the server failed...\n");
```

```
        exit(0);
```

```
    }
```

```
    else
```

```
        printf("connected to the server..\n");
```

```
    // function for chat
```

```
func(sockfd);
```

```
// close the socket
```

```
close(sockfd);
```

```
}
```

OUTPUT

The screenshot displays a web-based IDE interface with a dark theme. The browser's address bar shows the URL: `us-west-1.console.aws.amazon.com/cloud9/ide/151c4768c25240cdadda0c7bedf1f13c?#`. The IDE's file explorer on the left shows a project structure with folders 380 through 391, and files `a.out`, `client.c`, `client.c.o`, `server.c`, `server.c.o`, `xyz`, and `xyz.c`. The main editor area contains two files: `server.c` and `client.c`. The `server.c` file (lines 19-29) includes `write`, `bzero`, `read`, and `printf` calls for handling a client connection. The `client.c` file (lines 30-39) includes `int main()`, `sockfd`, `connfd`, `struct sockaddr_in servaddr, cli;`, and `socket` creation logic. Below the editor, a terminal window shows the output of running `server/386/server.c`. The terminal output is:

```
Server listening..
server accept the client...
From client: hello
To client :
```

 The terminal also shows a 'Stop' button and a 'Command' field with `server/386/server.c`. The bottom status bar indicates 'AWS: (not connected)', '27°C Partly cloudy', and the date '12-08-2021'.

```
19 write(sockfd, buff, sizeof(buff));
20 bzero(buff, sizeof(buff));
21 read(sockfd, buff, sizeof(buff));
22 printf("From Server : %s", buff);
23 if ((strcmp(buff, "exit", 4)) == 0) {
24     printf("client Exit...\n");
25     break;
26 }
27 }
28 }
29 }
30 int main()
31 {
32     int sockfd, connfd;
33     struct sockaddr_in servaddr, cli;
34
35     // socket create and verification
36     sockfd = socket(AF_INET, SOCK_STREAM, 0);
37     if (sockfd == -1) {
38         printf("socket creation failed...\n");
39         exit(0);
```

```
server/386/SERVER/serv x server/386/server.c - Run server/386/client.c - Run
Stop Command: server/386/server.c Runner: C CWD ENV
Server listening..
server accept the client...
From client: hello
To client :
```