

# TCP通信实验

## Server

```
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <unistd.h>
#include <string.h>
#include <signal.h>

double send_dg(int chat_fd);

int main(int argc, char *argv[]) {

    int server_fd = socket(AF_INET, SOCK_STREAM, 0); // create a socket file
descriptor: ipv4, TCP, ip
    if (server_fd < 0) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    struct sockaddr_in address = {
        .sin_family = AF_INET, // ipv4
        .sin_addr.s_addr = INADDR_ANY, // accept any incoming messages
        .sin_port = htons(9090), // port 9090
    }; // client addr

    int addrlen = sizeof(address);
    if (bind(server_fd, (struct sockaddr *) &address, addrlen) < 0) { // bind
connection
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

    if (listen(server_fd, 3) < 0) { // listen to connection
        perror("listen failed");
        exit(EXIT_FAILURE);
    }

    int stream_fd = accept(server_fd, (struct sockaddr *) &address, (socklen_t *) &addrlen); // accept connection and bind it to new file descriptor
    if (stream_fd < 0) {
        perror("accept failed");
        exit(EXIT_FAILURE);
    }

    send_dg(stream_fd);
```

```
close(server_fd);

return 0;
}

double send_dg(int chat_fd) {
    const int len = 1024;
    char buff[len];
    int n;
    while (1) {
        bzero(buff, len);
        read(chat_fd, buff, len);
        printf("%s\n", buff);

        bzero(buff, len);
        n = 0;
        while ((buff[n++] = getchar()) != '\n') { ;
        }

        write(chat_fd, buff, n);
        if (strcmp("exit", buff, 4) == 0) {
            printf("Server Exit\n");
            break;
        }
    }
}
```

## Client

```
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
double send_dg(int chat_fd);

int main(int argc, char *argv[]) {

    int client_fd = socket(AF_INET, SOCK_STREAM, 0); // create a socket file
descriptor: ipv4, TCP, ip
    if (client_fd < 0) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    struct sockaddr_in address={
        .sin_family = AF_INET, // ipv4
        .sin_port = htons(9090), // port 9090
}
```

```
}; // server addr

if (inet_pton(AF_INET, "127.0.0.1", &address.sin_addr) <= 0) { // check
    validity of ip addr
    perror("Invalid address");
    exit(EXIT_FAILURE);
}

if (connect(client_fd, (struct sockaddr *) &address, sizeof(address))) { // connect to server
    perror("Connection failed");
    exit(EXIT_FAILURE);
}

send_dg(client_fd);
close(client_fd);

return 0;
}

double send_dg(int chat_fd) {
    const int len = 1024;
    char buff[len];
    int n;
    while (1) {
        bzero(buff, len);
        n = 0;
        while ((buff[n++] = getchar()) != '\n') { ;
        }

        write(chat_fd, buff, n);

        bzero(buff, len);
        read(chat_fd, buff, len);
        printf("%s\n", buff);
        if (strcmp("exit", buff, 4) == 0) {
            printf("Server Exit\n");
            break;
        }
    }
}
```

# UDP通信实验

## Server

```
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
```

```
#include <netinet/in.h>
#include <unistd.h>
#include <string.h>
#include <signal.h>

int main(int argc, char *argv[]) {

    int server_fd = socket(AF_INET, SOCK_DGRAM, 0); // create a socket file
descriptor: ipv4, UDP, ip
    if (server_fd < 0) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    struct sockaddr_in address = {
        .sin_family = AF_INET, // ipv4
        .sin_addr.s_addr = INADDR_ANY, // accept any incoming messages
        .sin_port = htons(9090), // port 9090
    }; // client addr

    if (bind(server_fd, (struct sockaddr *) &address, sizeof(address)) < 0) { // bind connection
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

    int i;
    int n = 0;
    struct sockaddr_in cliaddr; int len;
    for(;;) {
        recvfrom(server_fd, &i, sizeof(i), MSG_WAITALL, (struct sockaddr *) &cliaddr, (socklen_t *) &len);
        if(i == 1000) {
            break;
        }
        n++;
    }
    printf("%d\n", n);

    close(server_fd);

    return 0;
}
```

## Client

```
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <unistd.h>
```

```
#include <string.h>
#include <arpa/inet.h>

int main(int argc, char *argv[]) {

    int client_fd = socket(AF_INET, SOCK_DGRAM, 0); // create a socket file
descriptor: ipv4, UDP, ip
    if (client_fd < 0) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    struct sockaddr_in address={
        .sin_family = AF_INET, // ipv4
        .sin_port = htons(9090), // port 9090
        .sin_addr.s_addr = INADDR_ANY,
    }; // server addr

    if (inet_pton(AF_INET, "127.0.0.1", &address.sin_addr) <= 0) { // check
validity of ip addr
        perror("Invalid address");
        exit(EXIT_FAILURE);
    }

    const int len = 1000;
    for (int i = 0; i < len + 1; i++) {
        sendto(client_fd, &i, sizeof(i), MSG_CONFIRM, (const struct sockaddr
*)&address, sizeof(address));
    }
    printf("1000 message sent");

    close(client_fd);

    return 0;
}
```

# 实验结果

## TCP通信

The image shows two terminal windows side-by-side. The left window is titled 'chjz@chjz-X6Ti: ~/Desktop/socket/cmake-build-debug' and contains the following text:  
File Edit View Search Terminal Help  
chjz@chjz-X6Ti:~/Desktop/socket/cmake-build-debug\$ ./server  
Hi  
Hello  
Let's chat  
ok  
Close tcp connection  
exit  
Server Exit  
chjz@chjz-X6Ti:~/Desktop/socket/cmake-build-debug\$ █

The right window is also titled 'chjz@chjz-X6Ti: ~/Desktop/socket/cmake-build-debug' and contains:  
File Edit View Search Terminal Help  
chjz@chjz-X6Ti:~/Desktop/socket/cmake-build-debug\$ ./client  
Hi  
Hello  
Let's chat  
ok  
Close tcp connection  
exit  
Server Exit  
chjz@chjz-X6Ti:~/Desktop/socket/cmake-build-debug\$ █

## UDP通信

### 接收端

The image shows a terminal window titled 'Terminal' with the following content:  
sanboot@A32:~\$ cd Desktop  
sanboot@A32:~/Desktop\$ gcc serverudp.c -std=c99 -o sudp.out  
sanboot@A32:~/Desktop\$ ./sudp.out  
944  
sanboot@A32:~/Desktop\$ ./sudp.out  
1000  
sanboot@A32:~/Desktop\$ ./sudp.out  
1000  
sanboot@A32:~/Desktop\$ █

On the right, there is a screenshot of a Linux desktop environment showing the terminal window above and the system tray at the top right.

Below the terminal window, the desktop environment shows a dock with various icons, including a browser, file manager, and system settings.

The terminal window continues with:  
sanboot@A32:~\$ ifconfig  
eth0 Link encap:Ethernet HWaddr 8c:89:a5:7b:66:bf  
inet addr:192.168.133.27 Bcast:192.168.133.255 Mask:255.255.255.0  
inet6 addr: fe80::8e89:a5ff:fe7b:66bf/64 Scope:Link  
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
RX packets:271411 errors:0 dropped:31 overruns:0 frame:0  
TX packets:55761 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:1000  
RX bytes:387510559 (387.5 MB) TX bytes:26632403 (26.6 MB)  
lo Link encap:Local Loopback  
inet addr:127.0.0.1 Mask:255.0.0.0  
inet6 addr: ::1/128 Scope:Host  
UP LOOPBACK RUNNING MTU:16436 Metric:1  
RX packets:48 errors:0 dropped:0 overruns:0 frame:0  
TX packets:48 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:0  
RX bytes:3664 (3.6 KB) TX bytes:3664 (3.6 KB)  
sanboot@A32:~\$ █

### 发送端

