实验序号： 4

《UNIX/LINUX 编程环境》

实验报告

实验名称：	Linux 网络编程之 UDP 编程
姓　　名：	李昊唐
学　　院：	工学院
专　　业：	物联网工程
班　　级：	1 班
学　　号：	1995131017
指导教师：	彭凯
实验地址：	数学学院 416
实验日期：	2021 年 12 月 14 日

实验 4 Linux 网络编程之 UDP 编程

一、实验目的

(1) 理解 Socket 概念

(2) 掌握 Linux 平台下 UDP 编程方法

二、实验环境

实验配置

本实验所需的软硬件配置如表 1 所示。

配置	2.6 GHz 六核 Intel Core i7, 16 GB 2400 MHz DDR4, Intel UHD Graphics 630 1536 MB
硬件	MacBook Pro (15-inch, 2019)
系统	macOS 12.0.1
应用软件	vi, sh

实验环境

本实验的环境为 Macintosh 机，如图 1 所示。

图 1 操作实验环境

三、实验原理

（1）程序进行网络通信时，是通过 IP 地址和套接字来访问一个主机的。

IP 地址：IP 地址的作用是标识计算机的网卡地址，每一台计算机都有一个 IP 地址。在程序中是通过 IP 地址来访问一台计算机的。IP 地址是 32 位长度的二进制数值，存储空间是 4 个字

节。例如 1100000010101000 00000001 00000110 是一台计算机的 IP 地址。IP 地址可以使用点分十进制来表示，192.168.1.1。

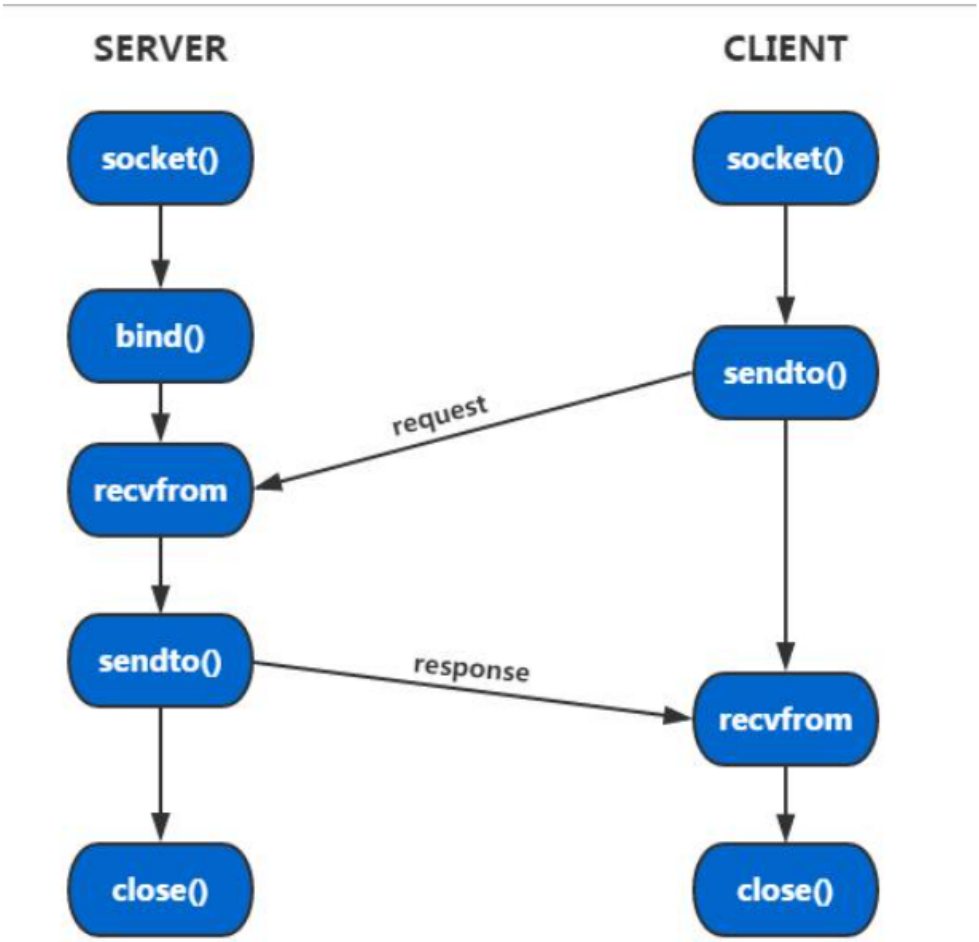
端口：所谓端口，是指计算机中为了标识在计算机中访问网络的不同程序而设的编号。端口号是一个 16 位的无符号整数，对应的十进制取值范围是 0~65535。

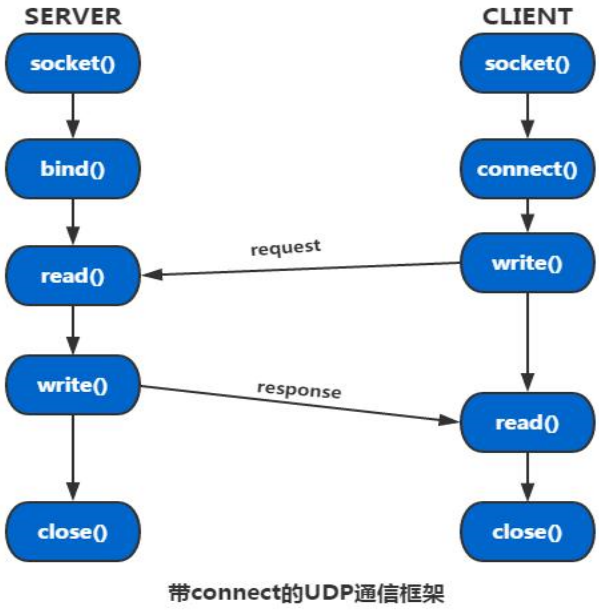
（2）UDP

UDP：UDP 是一种不面向连接的传输方式。对传输可靠性要求不高时，可以选择使用这种传输方式。

（3）套接字

UDP 编程的流程图





带connect的UDP通信框架

四、实验步骤和实验结果

程序要求实现功能：

客户端和服务器端通过 UDP 方式传输数据，客户端从本地读一个文件名为“IOT2019”的文件发送给服务器端，服务器收到后读出并显示文件内容，读完毕后关闭 socket 连接。

（请分别用带有 connect()和不带有 connect()函数方式实现）

IOT2019 内容

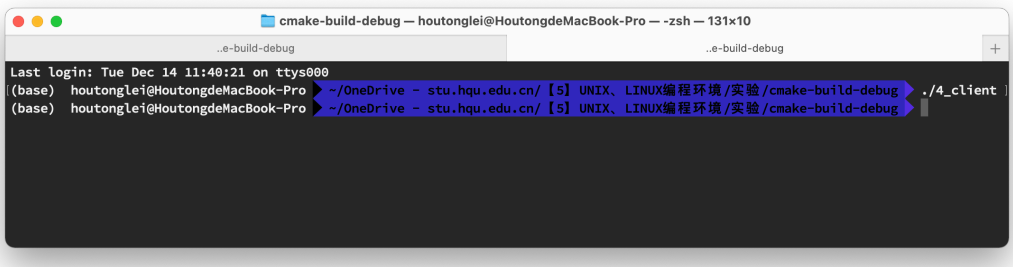
I am IOT2019er

I come from HQU

客户端（带 connect()）：

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

int main() {
    struct sockaddr_in server_addr;
    char buf[32];
    size_t length;
    int bufSize = 32;
    int sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    server_addr.sin_port = htons(9739);
    int server_len = sizeof(server_addr);
    char fileName[8] = "IOT2019";
    FILE *sfp = fopen("../实验 4/IOT2019", "rb");
    sendto(sockfd, fileName, strlen(fileName), 0, (struct sockaddr
*) &server_addr, server_len);
    while ((length = fread(buf, 1, bufSize, sfp)) > 0)
        sendto(sockfd, buf, length, 0, (struct sockaddr *)
&server_addr, server_len);
    fclose(sfp);
    sendto(sockfd, "\a", bufSize, 0, (struct sockaddr *)
&server_addr, server_len);
    close(sockfd);
    return 0;
}
```

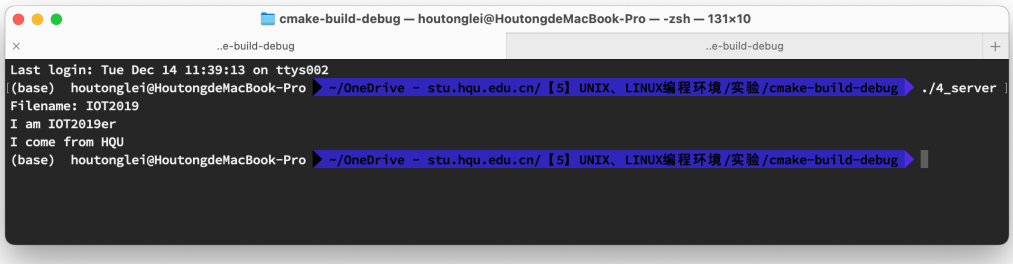


客户端（不带 connect()）：

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

int main() {
    struct sockaddr_in serverAddr;
    char buf[32];
    int bufSize = 32;
    int sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    serverAddr.sin_port = htons(9739);
    int serverLen = sizeof(serverAddr);
    size_t length;
    char fileName[8] = "IOT2019";
    FILE *sfp = fopen("../实验 4/IOT2019", "rb");
    if ((connect(sockfd, (struct sockaddr *) &serverAddr,
serverLen)) != 0) return 1;
    send(sockfd, fileName, strlen(fileName), 0);
    while ((length = fread(buf, 1, bufSize, sfp)) > 0)
        send(sockfd, buf, length, 0);
    fclose(sfp);
    send(sockfd, "\a", bufSize, 0);
    close(sockfd);
    return 0;
}
```

服务端：



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

int main() {
    socklen_t client_len = 0;
    char buf[32], fileName[8];
    int bufSize = 32, flag = 1;
    struct sockaddr_in server_addr;
    struct sockaddr_in client_addr;
    int sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    server_addr.sin_port = htons(9739);
    int server_len = sizeof(server_addr);
    if (bind(sockfd, (struct sockaddr *) &server_addr,
server_len) != 0) return 1;
    signal(SIGCHLD, SIG_IGN);
    while (flag) {
        recvfrom(sockfd, fileName, bufSize, 0, (struct sockaddr *)
&client_addr, &client_len);
        printf("Filename: %s \n", fileName);
        FILE *rfp = fopen(fileName, "wb");
        ssize_t length;
        while ((length = recvfrom(sockfd, buf, bufSize, 0, (struct
sockaddr *) &client_addr, &client_len))) {
            if (!strcmp(buf, "\a")) { flag = 0; break; }
            else fwrite(buf, 1, length, rfp);
        }
        fclose(rfp);
    }
    close(sockfd);
    FILE *rfp = fopen(fileName, "rb");
    while (fgets(buf, bufSize, rfp) != NULL) printf("%s", buf);
    fclose(rfp);
    return 0;
}
```

五、实验总结

在 UDP 编程中，需要将接收端设置成不断从连接获取消息，否则发送端在接收端离线时发送的数据不会保存在 buffer 中。同时，网络编程较难调试，一般 IDE 无法跟踪系统端口数据的传输，调试时一般只能凭感觉。