姓名：吴涵　　学号：202101000720　　成绩：

实验 33 套接字通信

1、消息传递通信

源程序：

sockserver.c：

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <semaphore.h>
#define LinkNum 5          // 连接数
int client_sockfd[LinkNum]; /*分别记录服务器端的套接字与连接的多个客户端的套接字*/
int server_sockfd = -1;     // 命名套接字
int curLink = 0;           // 当前连接数
sem_t mutex;               // 表示连接数的资源信号量
char stopmsg[100];          // 服务器端发送消息缓冲区
void quit()
{ // 客户服务通信结束处理函数
    int i;
    char *msg = "服务器将要关闭了!";
    while (1)
    {
        if (strcmp(stopmsg, "quit") == 0)
        { // 如果服务器端发送消息为"quit"，则提示服务器将关闭
            printf("服务器关闭!\n");
            for (i = 0; i < LinkNum; i++)
                if (client_sockfd[i] != -1)
                    write(client_sockfd[i], msg, sizeof(msg));
            /*依次向继续保持连接的客户端发出“服务器将关闭”的通知消息*/
            close(server_sockfd); // 关闭服务器监听套接字
            sem_destroy(&mutex);  // 销毁连接数资源信号量 mutex
            exit(0);
        }
    }
}
void rcv_snd(int n)
{ // 服务器与客户端的收发通信函数，n 为连接数组序号
    int i = 0;
    int retval;
    char recv_buf[1024]; // 接收消息缓冲区
    char send_buf[1024]; // 发送消息缓冲区
    int client_len = 0;
    int rcv_num;          // 从客户端接收到的消息长度
    pthread_t tid;        // 线程 id
    tid = pthread_self(); // 获取当前线程 id
```

```
    printf("-----------服务器线程 id=%u 使用套接字%d,n=%d 与客户机对话开始...\n", tid,
client_sockfd[n], n);
    do
    {                              // 服务器与客户端循环发送接收消息
        memset(recv_buf, 0, 1024); // 接收消息缓冲区清零
        printf("服务器线程 id=%u,套接字%d,n=%d 等待客户端回应...\n", tid, client_sockfd[n], n);
        rcv_num = read(client_sockfd[n], recv_buf, sizeof(recv_buf));
        printf("服务器线程 id=%u,套接字%d,n=%d 从客户端接受的消息长度=%d\n", tid,
client_sockfd[n], n, strlen(recv_buf));
        printf("3.服务器线程 id=%u,套接字%d,n=%d<---客户端,服务器从客户端接受的消息是:(%d) :%s\n",
tid, client_sockfd[n], n, rcv_num, recv_buf);
        if (rcv_num == 0)
            break;
        sleep(1);
        if (strncmp(recv_buf, "!q", 2) == 0)
            break; // 若接收到"!q"，则结束循环，通信结束
        printf("4.服务器线程 id=%u,套接字%d,n=%d--->客户端,请输入服务器要发送给客户机的消息：",
tid, client_sockfd[n], n);
        memset(send_buf, 0, 1024); // 发送消息缓冲区清零
        scanf("%s", send_buf);     // 服务器端键盘输入字符串消息，输入"!q"或"quit"，则通信结束
        strcpy(stopmsg, send_buf);
        write(client_sockfd[n], send_buf, sizeof(send_buf));
        if (strncmp(send_buf, "!q", 2) == 0)
            break; // 若服务器端发送"!q"，则结束循环，通信结束
        if (strncmp(send_buf, "quit", 4) == 0)
            break; // 若服务器端发送"quit"，则结束循环，通信结束
    } while (strncmp(recv_buf, "!q", 2) != 0);
    printf("-----------服务器线程 id=%u,套接字%d,n=%d 与客户机对话结束---------\n", tid,
client_sockfd[n], n);
    close(client_sockfd[n]);                                // 关闭连接套接字
    client_sockfd[n] = -1;                                  // 被关闭连接套接字数组项置为空闲
    curLink--;                                             // 当前连接数减 1
    printf("当前连接数为：%d(<=%d)\n", curLink, LinkNum); // 输出当前连接数和最大连接数
    sem_post(&mutex);                                       // 释放可用连接数资源信号量 mutex
    pthread_exit(&retval);                                  // 当前服务器线程结束
}
int main(void)
{
    char recv_buf[1024]; // 接收消息缓冲区
    char send_buf[1024]; // 发送消息缓冲区
    int client_len = 0;
    struct sockaddr_in server_addr; // 服务器端协议地址
    struct sockaddr_in client_addr; // 客户端协议地址
    int i = 0;                      // 连接套接字数组循环变量
    pthread_t thread;
    server_sockfd = socket(AF_INET, SOCK_STREAM, 0);
    server_addr.sin_family = AF_INET;                                          // 指定网络套接
字
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);                            // 接受所有 IP
地址的连接
    server_addr.sin_port = htons(9736);                                        // 绑定到 9736
端口
    bind(server_sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr)); // 协议套接
字命名为 server_sockfd
    printf("1、服务器开始 listen...\n");
    listen(server_sockfd, LinkNum);                          /*创建连接数最大为 LinkNum 的套接字队
列，监听命名套接字，listen 不会阻塞，它向内核报告套接字和最大连接数*/
    signal(SIGCHLD, SIG_IGN);                                // 忽略子进程停止或退出信号
```

```
    printf("输入!q，服务结束.\n");                          // 输入!q，服务结束
    pthread_create(&thread, NULL, (void *)(&quit), NULL); // 创建线程，执行函数 quit
    for (i = 0; i < LinkNum; i++)
        client_sockfd[i] = -1;    // 初始化连接队列
    sem_init(&mutex, 0, LinkNum); // 信号量 mutex 初始化为连接数
    while (1)
    {
        for (i = 0; i < LinkNum; i++) // 搜寻空闲连接
            if (client_sockfd[i] == -1)
                break;
        if (i == LinkNum)
        { // 如果达到最大连接数，则客户等待
            printf("已经达到最大连接数%d,请等待其它客户释放连接...\n", LinkNum);
            sem_wait(&mutex); // 阻塞等待空闲连接
            continue;         // 被唤醒后继续监测是否有空闲连接
        }
        client_len = sizeof(client_addr);
        printf("2、服务器开始 accept...i=%d\n", i);
        client_sockfd[i] = accept(server_sockfd, (struct sockaddr *)&client_addr,
&client_len);
        curLink++;        // 当前连接数增 1
        sem_wait(&mutex); // 可用连接数信号量 mutex 减 1
        printf("当前连接数为：%d(<=%d)\n", curLink, LinkNum);
        printf("连接来自:连接套接字号=%d,IP 地址=%s,端口号=%d\n", client_sockfd[i],
inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port)); // 输出客户端地址信息
        pthread_create(malloc(sizeof(pthread_t)), NULL, (void *)(&rcv_snd), (void *)i);
    }
}
```

sockclient.c：

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <errno.h>
#include <fcntl.h>
#include <string.h>
int main(void)
{
    int sockfd; // 客户端套接字描述符
    int len = 0;
    struct sockaddr_in address; // 套接字协议地址
    char snd_buf[1024];        // 发送消息缓冲区
    char rcv_buf[1024];        // 接收消息缓冲区
    int result;
    int rcv_num; // 接收消息长度
    pid_t cpid;  // 客户进程标识符
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
    {
        perror("客户端创建套接字失败! \n");
        return 1;
    }
    address.sin_family = AF_INET;                    // 使用网络套接字
```

```
    address.sin_addr.s_addr = inet_addr("127.0.0.1"); // 服务器地址
    address.sin_port = htons(9736);                  // 服务器所监听的端口
    if (inet_aton("127.0.0.1", &address.sin_addr) < 0)
    {
        printf("inet_aton error.\n");
        return -1;
    }
    len = sizeof(address);
    cpid = getpid(); // 获取客户进程标识符
    printf("1、客户机%ld 开始 connect 服务器...\n", cpid);
    result = connect(sockfd, (struct sockaddr *)&address, len);
    if (result == -1)
    {
        perror("客户机 connect 服务器失败!\n");
        exit(1);
    }
    printf("-----------客户机%ld 与服务器线程对话开始...\n", cpid);
    do
    { // 客户机与服务器循环发送接收消息
        printf("2.客户机%ld--->服务器:sockfd=%d,请输入客户机要发送给服务器的消息：", cpid,
sockfd);
        memset(snd_buf, 0, 1024);               // 发送缓冲区清零
        scanf("%s", snd_buf);                   // 键盘输入欲发送给服务器的消息字符串
        write(sockfd, snd_buf, sizeof(snd_buf)); // 将消息发送到套接字
        if (strncmp(snd_buf, "!q", 2) == 0)
            break;               // 若发送"!q"，则结束循环，通信结束
        memset(rcv_buf, 0, 1024); // 接收缓冲区清零
        printf("客户机%ld,sockfd=%d 等待服务器回应...\n", cpid, sockfd);
        rcv_num = read(sockfd, rcv_buf, sizeof(rcv_buf));
        printf("客户机%ld,sockfd=%d 从服务器接收的消息长度=%d\n", cpid, sockfd,
strlen(rcv_buf));
        printf("3.客户机%ld<---服务器:sockfd=%d,客户机从服务器接收到的消息是：（%d）:%s\n",
cpid, sockfd, rcv_num, rcv_buf); // 输出客户机从服务器接收的消息
        sleep(1);
        if (strncmp(rcv_buf, "quit", 4) == 0)
            break; // 如果收到"quit"，则结束循环，通信结束
    } while (strncmp(rcv_buf, "!q", 2) != 0); // 如果收到"!q"，则结束循环，通信结束
    printf("-----------客户机%ld,sockfd=%d 与服务器线程对话结束---------\n", cpid, sockfd);
    close(sockfd); // 关闭客户机套接字
}
```

编译链接命令：

gcc sockserver.c -o sockserver -lpthread

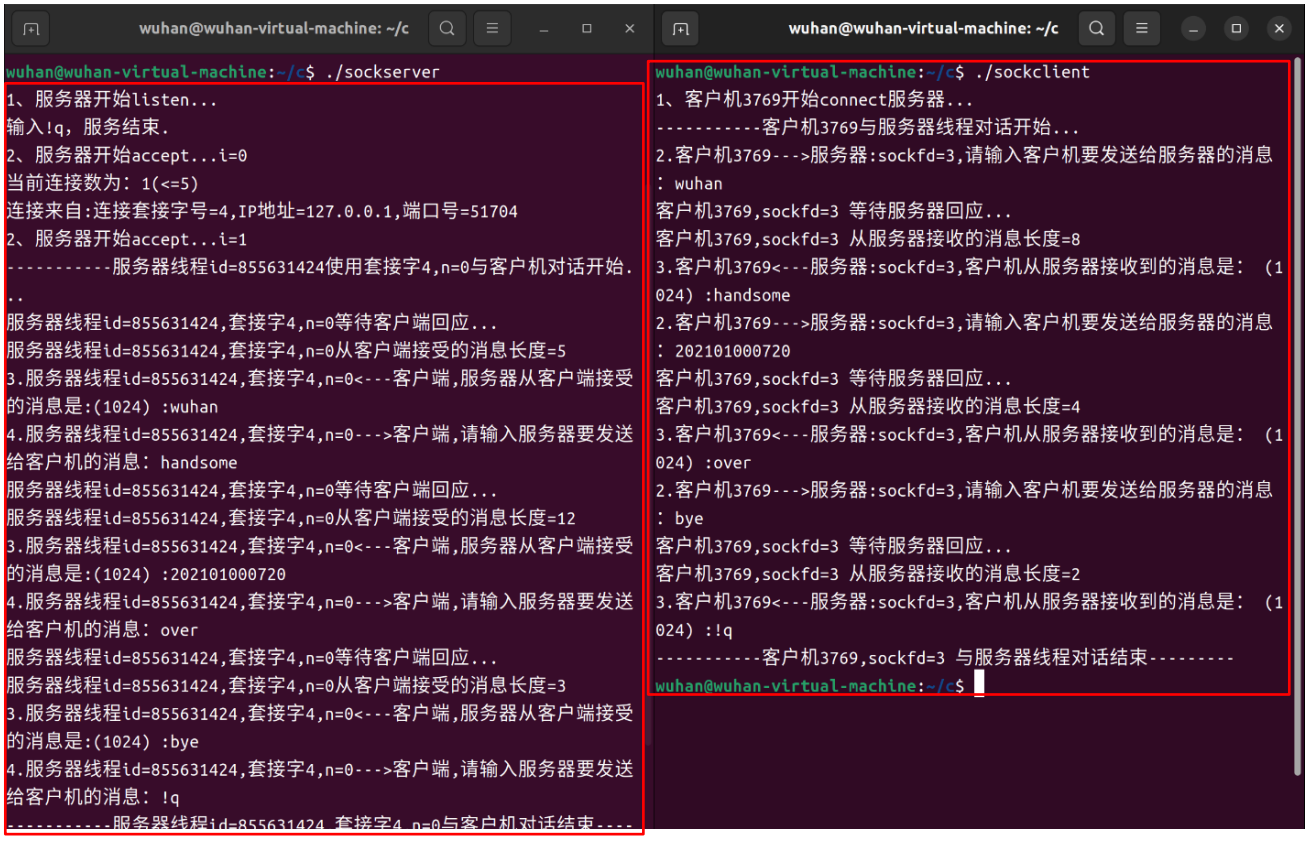
gcc sockclient.c -o sockclient

运行命令：

./ sockserver

./ sockclient

交互与结果：


wuhan@wuhan-virtual-machine: ~/c
wuhan@wuhan-virtual-machine:~/c$ ./sockserver
1、服务器开始listen...
输入!q，服务结束.
2、服务器开始accept...i=0
当前连接数为：1(<=5)
连接来自:连接套接字号=4,IP地址=127.0.0.1,端口号=51704
2、服务器开始accept...i=1
----------服务器线程id=855631424使用套接字4,n=0与客户机对话开始...
服务器线程id=855631424,套接字4,n=0等待客户端回应...
服务器线程id=855631424,套接字4,n=0从客户端接受的消息长度=5
3.服务器线程id=855631424,套接字4,n=0<---客户端,服务器从客户端接受的消息是:(1024) :wuhan
4.服务器线程id=855631424,套接字4,n=0--->客户端,请输入服务器要发送给客户机的消息：handsome
服务器线程id=855631424,套接字4,n=0等待客户端回应...
服务器线程id=855631424,套接字4,n=0从客户端接受的消息长度=12
3.服务器线程id=855631424,套接字4,n=0<---客户端,服务器从客户端接受的消息是:(1024) :202101000720
4.服务器线程id=855631424,套接字4,n=0--->客户端,请输入服务器要发送给客户机的消息：over
服务器线程id=855631424,套接字4,n=0等待客户端回应...
服务器线程id=855631424,套接字4,n=0从客户端接受的消息长度=3
3.服务器线程id=855631424,套接字4,n=0<---客户端,服务器从客户端接受的消息是:(1024) :bye
4.服务器线程id=855631424,套接字4,n=0--->客户端,请输入服务器要发送给客户机的消息：!q
----------服务器线程id=855631424,套接字4,n=0与客户机对话结束----
wuhan@wuhan-virtual-machine: ~/c
wuhan@wuhan-virtual-machine:~/c$ ./sockclient
1、客户机3769开始connect服务器...
----------客户机3769与服务器线程对话开始...
2.客户机3769--->服务器:sockfd=3,请输入客户机要发送给服务器的消息：wuhan
客户机3769,sockfd=3 等待服务器回应...
客户机3769,sockfd=3 从服务器接收的消息长度=8
3.客户机3769<---服务器:sockfd=3,客户机从服务器接收到的消息是：(1024) :handsome
2.客户机3769--->服务器:sockfd=3,请输入客户机要发送给服务器的消息：202101000720
客户机3769,sockfd=3 等待服务器回应...
客户机3769,sockfd=3 从服务器接收的消息长度=4
3.客户机3769<---服务器:sockfd=3,客户机从服务器接收到的消息是：(1024) :over
2.客户机3769--->服务器:sockfd=3,请输入客户机要发送给服务器的消息：bye
客户机3769,sockfd=3 等待服务器回应...
客户机3769,sockfd=3 从服务器接收的消息长度=2
3.客户机3769<---服务器:sockfd=3,客户机从服务器接收到的消息是：(1024) :!q
----------客户机3769,sockfd=3 与服务器线程对话结束---------
wuhan@wuhan-virtual-machine:~/c$