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：

