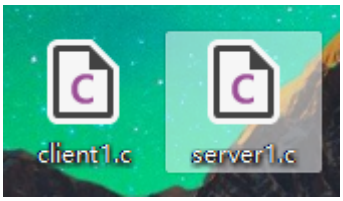
计算机网络编程第五次作业

姓名：冯冠玺 学号：15051415

一.实验内容

基于TCP的聊天室

二.实验文件



三.实验结果

服务器端运行截图：

```
guanxi@ubuntu:~$ gcc -o client client1.c
guanxi@ubuntu:~$ ./server
bing success!
listening......
accept from:-1208588000
listening......
 fgx:hello,I am FengGanxi (1:44:32)
```

客户端的截图

```
guanxi@ubuntu:~$ ./client 127.1.1.0 3490 fgx

|----------Welcome to the chat room! ----------|
hello,I am FengGanxi
fgx:hello,I am FengGanxi (1:44:33)
```

四.实验代码

Client:

```
#include<stdio.h>
#include<netinet/in.h> //定义数据结构sockaddr_in
```

```c
#include<sys/socket.h> //提供socket函数及数据结构
#include<sys/types.h> //数据类型定义
#include<string.h>
#include<stdlib.h>
#include<netdb.h>
#include<unistd.h>
#include<signal.h>
#include<time.h>
int main(int argc, char *argv[])
{
    struct sockaddr_in clientaddr;//定义地址结构
    pid_t pid;
    int clientfd,sendbytes,recvbytes;//定义客户端套接字
    struct hostent *host;
    char *buf,*buf_r;
    if(argc < 4)
    {
        printf("usage:\n");
        printf("%s host port name\n",argv[0]);
        exit(1);
    }
    host = gethostbyname(argv[1]);
    if((clientfd = socket(AF_INET,SOCK_STREAM,0)) == -1) //创建客户端套接字
    {
        perror("socket\n");
        exit(1);
    }
    //绑定客户端套接字
    clientaddr.sin_family = AF_INET;
    clientaddr.sin_port = htons((uint16_t)atoi(argv[2]));
    clientaddr.sin_addr = *((struct in_addr *)host->h_addr);
    bzero(&(clientaddr.sin_zero),0);
    if(connect(clientfd,(struct sockaddr *)&clientaddr,sizeof(struct sock
addr)) == -1) //连接服务端
    {
        perror("connect\n");
        exit(1);
    }
    buf=(char *)malloc(120);
    memset(buf,0,120);
    buf_r=(char *)malloc(100);

    if( recv(clientfd,buf,100,0) == -1)
    {
        perror("recv:");
        exit(1);
    }
    printf("\n%s\n",buf);

    pid = fork();//创建子进程
    while(1)
    {
```

```
        if(pid > 0){
            //父进程用于发送信息

            //get_cur_time(time_str);

            strcpy(buf,argv[3]);
            strcat(buf,":");
            memset(buf_r,0,100);
            //gets(buf_r);
            fgets(buf_r,100,stdin);
            strncat(buf,buf_r,strlen(buf_r)-1);
            //strcat(buf,time_str);
            //printf("---%s\n",buf);
            if((sendbytes = send(clientfd,buf,strlen(buf),0)) == -1)
            {
            perror("send\n");
            exit(1);
            }
        }
        else if(pid == 0)
        {
            //子进程用于接收信息
            memset(buf,0,100);
            if(recv(clientfd,buf,100,0) <= 0)
            {
                perror("recv:");
                close(clientfd);
                raise(SIGSTOP);
                exit(1);
            }
            printf("%s\n",buf);
        }
        else
            perror("fork");
    }
    close(clientfd);
    return 0;
}
```

Server

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h> //数据类型定义
#include<sys/stat.h>
#include<netinet/in.h> //定义数据结构sockaddr_in
#include<sys/socket.h> //提供socket函数及数据结构
#include<string.h>
#include<unistd.h>
```

```c
#include<signal.h>
#include<sys/ipc.h>
#include<errno.h>
#include<sys/shm.h>
#include<time.h>
#define PERM S_IRUSR|S_IWUSR
#define MYPORT 3490 //宏定义定义通信端口
#define BACKLOG 10 //宏定义，定义服务程序可以连接的最大客户数量
#define WELCOME "|----------Welcome to the chat room! ----------|" //宏定义，当客户端连接服务端时，想客户发送此欢迎字符串
//转换函数，将int类型转换成char *类型
void itoa(int i,char*string)
{
    int power,j;
        j=i;
    for(power=1;j>=10;j/=10)
        power*=10;
    for(;power>0;power/=10)
    {
        *string++='0'+i/power;
        i%=power;
    }
    *string='\0';
}

//得到当前系统时间
void get_cur_time(char * time_str)
{
    time_t timep;
    struct tm *p_curtime;
    char *time_tmp;
    time_tmp=(char *)malloc(2);
    memset(time_tmp,0,2);

    memset(time_str,0,20);
    time(&timep);
    p_curtime = localtime(&timep);
    strcat(time_str," (");
    itoa(p_curtime->tm_hour,time_tmp);
    strcat(time_str,time_tmp);
    strcat(time_str,":");
    itoa(p_curtime->tm_min,time_tmp);
    strcat(time_str,time_tmp);
    strcat(time_str,":");
    itoa(p_curtime->tm_sec,time_tmp);
    strcat(time_str,time_tmp);
    strcat(time_str,")");
    free(time_tmp);
}

//创建共享存储区，进程间通讯
key_t shm_create()
```

```
{
    key_t shmid;
    //shmid = shmget(IPC_PRIVATE,1024,PERM);
    if((shmid = shmget(IPC_PRIVATE,1024,PERM)) == -1)
    {
        fprintf(stderr,"Create Share Memory Error:%s\n\a",strerror(errn
o));
        exit(1);
    }
    return shmid;
}
//端口绑定函数,创建套接字，并绑定到指定端口
int bindPort(unsigned short int port)
{
    int sockfd;
    struct sockaddr_in my_addr;
    sockfd = socket(AF_INET,SOCK_STREAM,0);//创建基于流套接字
    my_addr.sin_family = AF_INET;//IPv4协议族
    my_addr.sin_port = htons(port);//端口转换
    my_addr.sin_addr.s_addr = INADDR_ANY;
    bzero(&(my_addr.sin_zero),0);//置空

    if(bind(sockfd,(struct sockaddr*)&my_addr,sizeof(struct sockaddr)) ==
 -1)//绑定本地IP
    {
        perror("bind");
        exit(1);
    }
    printf("bing success!\n");
    return sockfd;
}
int main(int argc, char *argv[])
{
    int sockfd,clientfd,sin_size,recvbytes; //定义监听套接字、客户套接字
    pid_t pid,ppid; //定义父子线程标记变量
    char *buf, *r_addr, *w_addr, *temp, *time_str;//="\0"; //定义临时存储区
    struct sockaddr_in their_addr; //定义地址结构
    key_t shmid;

    shmid = shm_create(); //创建共享存储区

    temp = (char *)malloc(255);
    time_str=(char *)malloc(20);
    sockfd = bindPort(MYPORT);//绑定端口
    while(1)
    {
        if(listen(sockfd,BACKLOG) == -1)//在指定端口上监听
        {
            perror("listen");
            exit(1);
        }
        printf("listening......\n");
```

```
        if((clientfd = accept(sockfd,(struct sockaddr*)&their_addr,&sin_s
ize)) == -1)//接收客户端连接
        {
            perror("accept");
            exit(1);
        }
        printf("accept from:%d\n",inet_ntoa(their_addr.sin_addr));
        send(clientfd,WELCOME,strlen(WELCOME),0);//发送问候信息
        buf = (char *)malloc(255);

        ppid = fork();//创建子进程
        if(ppid == 0)
        {
            //printf("ppid=0\n");
            pid = fork(); //创建子进程
            while(1)
            {
                if(pid > 0)
                {
                    //父进程用于接收信息
                    memset(buf,0,255);
                    //printf("recv\n");
                    //sleep(1);
                    if((recvbytes = recv(clientfd,buf,255,0)) <= 0)
                    {
                        perror("recv1");
                        close(clientfd);
                        raise(SIGKILL);
                        exit(1);
                    }
                    //write buf's data to share memory
                    w_addr = shmat(shmid, 0, 0);
                    memset(w_addr, '\0', 1024);
                    strncpy(w_addr, buf, 1024);
                    get_cur_time(time_str);
                    strcat(buf,time_str);
                    printf(" %s\n",buf);
                }
                else if(pid == 0)
                {
                    //子进程用于发送信息
                    //scanf("%s",buf);
                    sleep(1);
                    r_addr = shmat(shmid, 0, 0);
                    //printf("---%s\n",r_addr);
                    //printf("cmp:%d\n",strcmp(temp,r_addr));
                    if(strcmp(temp,r_addr) != 0)
                    {
                        strcpy(temp,r_addr);
                        get_cur_time(time_str);
                        strcat(r_addr,time_str);
                        //printf("discriptor:%d\n",clientfd);
```

```
                        //if(send(clientfd,buf,strlen(buf),0) == -1)
                        if(send(clientfd,r_addr,strlen(r_addr),0) == -1)
                        {
                            perror("send");
                        }
                        memset(r_addr, '\0', 1024);
                        strcpy(r_addr,temp);
                    }
                }
                else
                    perror("fork");
            }
        }
    }
    printf("------------------------------\n");
    free(buf);
    close(sockfd);
    close(clientfd);
    return 0;
}
```