

中国科学技术大学计算机学院

网络系统实验报告

实验三

Socket 编程-简单聊天程序

学 号：PE20060014

姓 名：王晨

专 业：计算机科学与技术

指导老师：张信明

中国科学技术大学计算机学院

2021 年 4 月 17 日

一、 实验目的

- 掌握 Socket 编程的常用函数方法、理解 Socket 通信流程

二、 实验环境

- 系统：Mac OS based on UNIX
- 编程语言：C/C++

三、 实验过程

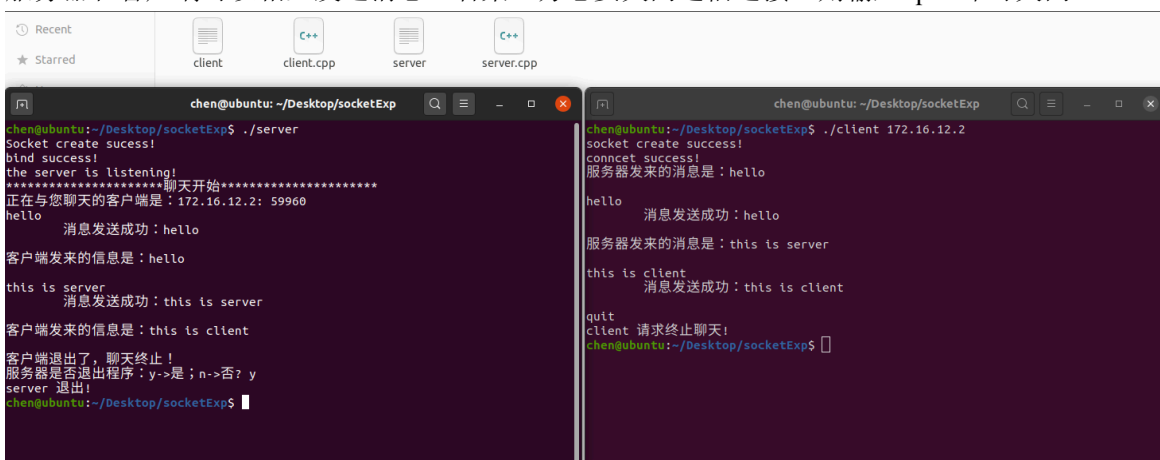
1. 实验结果展示

本实验要求是实现一个简单聊天程序，一个服务器，一个客户端，服务器可以与客户端互相发送聊天内容。

在 linux 系统上打开 server 端：./server 开启监听

之后在另一个用户终端输入：./client server 端 IP 地址 建立连接后即可开启通信。

服务器和客户端可以相互发送消息，若某一方想要关闭通信连接，则输入 quit 即可关闭。



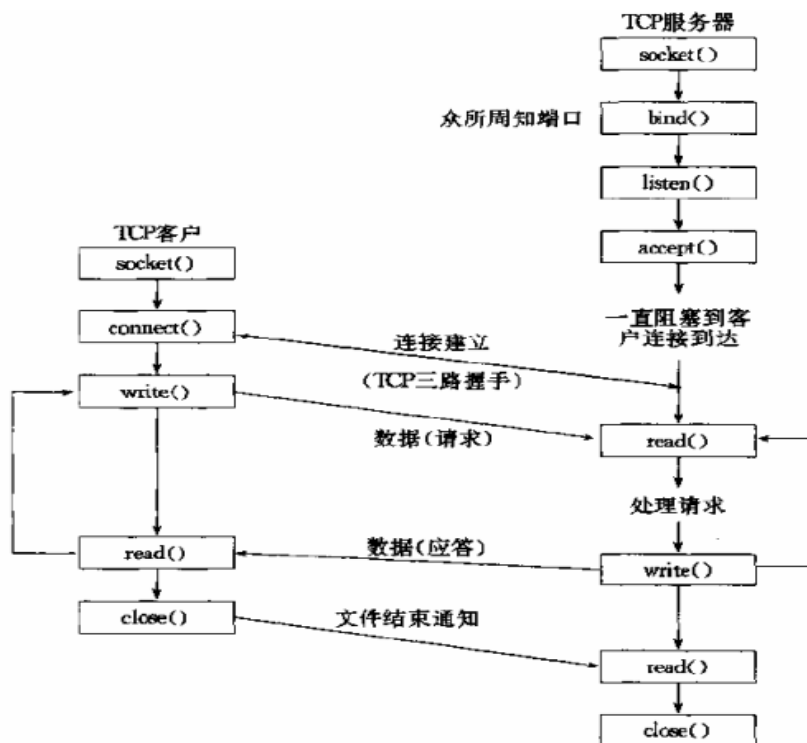
```
chen@ubuntu: ~/Desktop/socketExp
Recent
★ Starred
client client.cpp server server.cpp

chen@ubuntu:~/Desktop/socketExp$ ./server
Socket create success!
bind success!
the server is listening!
*****聊天开始*****
正在与您聊天的客户端是：172.16.12.2: 59960
hello
    消息发送成功: hello
客户端发来的信息是: hello
this is server
    消息发送成功: this is server
客户端发来的信息是: this is client
客户端退出了，聊天终止！
服务器是否退出程序：y->是；n->否？ y
server 退出！
chen@ubuntu:~/Desktop/socketExp$

chen@ubuntu:~/Desktop/socketExp$ ./client 172.16.12.2
socket create success!
connctet success!
服务器发来的消息是: hello
hello
    消息发送成功: hello
服务器发来的消息是: this is server
this is client
    消息发送成功: this is client
quit
client 请求终止聊天！
chen@ubuntu:~/Desktop/socketExp$
```

2. 源程序说明

两个 cpp 文件和可执行文件附在压缩包中，这里重点结合代码注释分析聊天程序的源代码。程序流程图和上课时 ppt 所给基本 TCP 客户-服务器流程的是一致的，如下图所示：



我们首先说明 server.cpp 源代码，以下是一些函数中需要调用的全局变量，相关含义已经注释：

```

int sockfd, newfd;           //socket handler
struct sockaddr_in s_addr, c_addr; //for server and client address
char buf[BUFLen];           //聊天数据流
socklen_t len;               //accept()调用参数，socklen_t长度和int相同
unsigned int port, listnum; //端口号，连接请求暂存队列长度
fd_set rfd;                  //文件描述字结构体，与文件句柄(socket、文件、管道、设备等)建立联系
struct timeval tv;           //超时时间
int retval, maxfd;
  
```

首先建立 socket，并指定协议，获取 socket 句柄：

```

//establish socket and get a handler
//check socket() for more info about arguments
if((sockfd=socket(PF_INET, SOCK_STREAM, 0))==-1){ //tcp/ip protocol, 顺序、可靠、双向数据流, TCP 连接。
    perror("socket");
    exit(errno);
}else
    printf("Socket create success!\n");
memset(&s_addr, 0, sizeof(s_addr));
s_addr.sin_family=AF_INET;
s_addr.sin_port=htons(PORT);
s_addr.sin_addr.s_addr=htons(INADDR_ANY);
  
```

其中 struct sockaddr_in 是之后 bind 需要使用的结构体，我们将相关协议、端口号、IP 地址填入，它的具体定义如下：

```

struct sockaddr_in
{
    sa_family_t      sin_family;      //地址族 (Address Family)
    uint16_t         sin_port;        //16 位 TCP/UDP 端口号
    struct in_addr    sin_addr;       //32 位 IP 地址
    char             sin_zero[8];     //不使用
};

```

完成 bind 后调用 listen 监听本地端口，调用 listen() 之后，系统将给此 Socket 配备一个连接请求的队列，暂存系统接收到的、申请向此 Socket 建立连接的请求，等待用户程序用 accept() 正式接受该请求。

```

//bind addr and port procedure
if((bind(sockfd,(struct sockaddr*)&s_addr,sizeof(struct sockaddr))!=-1)){
    perror("bind");
    exit(errno);
}else
    printf("bind success!\n");
/*侦听本地端口*/
if(listen(sockfd,listnum) == -1){
    perror("listen");
    exit(errno);
}else
    printf("the server is listening!\n");

```

接受请求，建立连接，其中 inet_ntoa() 功能是将网络地址转换成点分十进制，ntohs() 是将一个 16 位数由网络字节顺序转换为主机字节顺序。accept() 会从该暂存队列中取出一个连接请求，用该 Socket 的数据，创建一个新的 Socket：newfd，并为它分配一个文件描述符。newfd 即标识了此次建立的连接，可被己方用来向连接的另一方发送和接收数据（write/read，send/recv）。同时，原 Socket 仍然保持打开状态不变，继续用于等待网络连接请求。

```

//Communication part
while (1){
    printf("*****聊天开始*****\n");
    len =sizeof (struct sockaddr);
    if((newfd=accept(sockfd,(struct sockaddr*)&c_addr,&len))!=-1 ){
        perror("accept");
        exit(errno);
    }else
        printf("正在与您聊天的客户端是: %s: %d\n",inet_ntoa(c_addr.sin_addr),ntohs( c_addr.sin_port));

```

在完成通信连接的建立后，就可以处理聊天的功能了，这里我们没有用 ppt 中的 read/recv 函数，这是因为，而是用 select 以及 fd 的操作来完成异步的网络消息处理。这两者的区别是 read，recv，recvfrom 函数都是阻塞函数，当函数不能成功执行的时候，程序就会一直阻塞在这里，无法执行下面的代码，有点不太好，这时选用 select 函数就可以实现非阻塞编程。

select 函数是一个轮循函数，循环询问文件节点，看是否可读可写，可设置超时时间，超时时间到了就跳过代码继续往下执行。在执行 select 前需要确定最大的文件描述符，这是 select 参数中所要求的（maxfd 为集合中所有文件描述符的范围）。tv 为超时时间。Select（）第二个参数表述，如果 rfd

这个集合中有一个文件可读，`select` 就会返回一个大于 0 的值，表示有文件可写，如果没有可写的文件，则根据 `tv` 再判断是否超时，若超出时间，`select` 返回 0，若发生错误返回负值。

```
FD_ZERO( p: &rfd);
FD_SET( n: 0, p: &rfd);
maxfd=0;
FD_SET(newfd, p: &rfd);
//找出文件描述符集合中最大的文件描述符
if(maxfd<newfd)
    maxfd=newfd;
//设置超时时间
tv.tv_sec=6;
tv.tv_usec=0;
//等待聊天
retval=select(maxfd+1,&rfd,NULL,NULL,&tv);
if(retval== -1){
    printf("select出错，与该客户端连接的程序将退出\n");
    break;
} else if(retval==0) {
    printf("waiting.....\n");
    continue;
} else{
```

接下来是服务器端的消息处理，到这里就比较简单了，如果输入 `quit` 就退出循环，否则就用 `send` 进行发送，返回值为实际成功发送的字节数，-1 表示出错。

```
} else{
    //用户输入信息
    if(FD_ISSET( n: 0, p: &rfd)) {
        //发送消息
        memset(buf, c: 0, sizeof(buf));
        //fgets函数：从流中读取BUFLen-1个字符
        fgets(buf, BUFLen, stdin);
        //打印发送的消息
        if(!strncasecmp(buf, "quit", 4)){
            printf("server请求终止聊天!\n");
            break;
        } else
            len = send(newfd, buf, strlen(buf), 0);
        if(len>0)
            printf("\t消息发送成功: %s\n", buf);
        else{
            printf("消息发送失败! \n");
            break;
        }
    }
}
```

如果读到有客户端发来的消息，同样地，就用 `recv()` 进行接收，如果通信断开了（循环退出），关闭 socket 即可。

```

//客户端发来消息
if(FD_ISSET(newfd, p: &rfdsets)){
    //接收消息
    memset(buf, 0, sizeof(buf));
    //fgets函数: 从流中读取BUFLen-1个字符
    //fgets(buf, BUFLen, stdin);
    len=recv(newfd, buf, BUFLen, 0);
    if(len > 0)
        printf("客户端发来的信息是: %s\n", buf);
    else {
        if (len < 0)
            printf("接受消息失败! \n");
        else
            printf("客户端退出了, 聊天终止! \n");
        break;
    }
}
}
//关闭聊天的套接字
close(newfd);
//是否推出服务器
printf("服务器是否退出程序: y->是; n->否? ");
bzero(buf, BUFLen);
fgets(buf, BUFLen, stdin);
if(!strncasecmp(buf, "y", 1)){
    printf("server 退出!\n");
    break;
}
}

```

以上是 server 端的代码，理解了 server 端的代码后，client 就很简单了，这里就只简单说明一下。同样的，开始也是先建立一个 socket 并获取句柄，client 运行时需要指定 server 端的 ip 地址作为参数，把这个 ip 地址用 inet_aton 写入到 s_addr 中。

```

/*建立socket*/
if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1){
    perror("socket");
    exit(errno);
}else
    printf("socket create success!\n");

/*设置服务器ip*/
memset(&s_addr, 0, sizeof(s_addr));
s_addr.sin_family = AF_INET;
s_addr.sin_port = htons(PORT);
if (inet_aton(argv[1], (struct in_addr *)&s_addr.sin_addr.s_addr) == 0) {
    perror(argv[1]);
    exit(errno);
}

/*开始连接服务器*/
if(connect(sockfd, (struct sockaddr *)&s_addr, sizeof(struct sockaddr)) == -1){
    perror("connect");
    exit(errno);
}else
    printf("connect success!\n");

```

connect() 函数，则是主动地向对方建立连接时使用的。connect() 使用一个事先打开的 Socket，和目的方（即通信对方，或称服务器一方）地址信息，向对方发出连接建立请求。

用 `connect()` 建立连接后，类似地就开始处理消息了，这里的代码和 `server` 端基本时一样的，就不再赘述了：

```
FD_ZERO(&p; &rfd);
FD_SET(n; 0, &p; &rfd);
maxfd = 0;
FD_SET(sockfd, &p; &rfd);
if(maxfd < sockfd)
    maxfd = sockfd;
tv.tv_sec = 6;
tv.tv_usec = 0;
retval = select(maxfd+1, &rfd, NULL, NULL, &tv);
if(retval == -1){
    printf("select出错, 客户端程序退出\n");
    break;
}else if(retval == 0){
    printf("waiting...\n");
    continue;
}else{
    /*服务器发来了消息*/
    if(FD_ISSET(sockfd, &rfd)){
        /******接收消息******/
        bzero(buf, BUFLen);
        len = recv(sockfd, buf, BUFLen, 0);
        if(len > 0)
            printf("服务器发来的消息是: %s\n", buf);
        else{
            if(len < 0)
                printf("接受消息失败! \n");
            else
                printf("服务器退出了, 聊天终止! \n");
            break;
        }
    }
}
```

```
/*用户输入信息了, 开始处理信息并发送*/
if(FD_ISSET(n; 0, &p; &rfd)){
    /******发送消息******/
    bzero(buf, BUFLen);
    fgets(buf, BUFLen, stdin);

    if(!strncasecmp(buf, "quit", 4)){
        printf("client 请求终止聊天!\n");
        break;
    }
    len = send(sockfd, buf, strlen(buf), 0);
    if(len > 0)
        printf("\t消息发送成功: %s\n", buf);
    else{
        printf("消息发送失败!\n");
        break;
    }
}
}

/*关闭连接*/
close(sockfd);

return 0;
```

以上就是 socket 聊天程序的全部代码了，源文件附在压缩包中，如果无法编写或运行，请与我联系。