



# 计算机网络实验报告

## 实验二：Echo实验

数据科学与计算机学院 17大数据与人工智能

17341015 陈鸿峰

### 一、实验目的

掌握套接字的基本使用方法

### 二、实验说明

- 把源程序和可执行文件放在相应的上交源码目录中
- 截屏用按键(Ctrl+Alt+PrintScreen)截取当前窗口

### 三、参考资料

- 套接字, <https://www.cnblogs.com/hgwan/p/6074038.html>

### 四、实验环境

本机为Ubuntu 18.04 (LTS) + gcc 7.3.0

### 五、实验内容

先尝试运行文件夹“TCP”中的程序：先运行Server程序（打开TCPServer.sln，然后执行）再运行Client程序（打开TCPClient.sln，然后执行）。这两个程序的功能是客户端从服务器获取当前时间。

由于本实验在Linux环境下进行，故没有按照上面步骤进行，但是Linux程序可正常编译运行。

#### 1. TCP Echo程序

##### (i) 实验要求

服务器把客户端发送来的任何消息都返回给客户端，返回的消息前面要加上服务器的当前时间。

客户端把返回的消息显示出来。客户端每输入一条消息就建立TCP连接，并把消息发送给服务器，在收到服务器回应后关闭连接。

##### (ii) 运行结果

- 客户端（两次运行）

```

chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming$ ./client_tcp
输入要发送的信息: aaa
收到的信息:
Fri Mar 8 15:58:24 2019
aaa
按回车键继续...
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming$ ./client_tcp
输入要发送的信息: bbb
收到的信息:
Fri Mar 8 15:58:30 2019
bbb
按回车键继续...
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming$

```

## • 服务器

```

chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming$ ./server_tcp
服务器已启动!
收到消息: aaa
收到时间: Fri Mar 8 15:58:24 2019
收到消息: bbb
收到时间: Fri Mar 8 15:58:30 2019

```

## • 只运行客户端程序而不运行服务器程序会出现什么错误，截屏并说明原因。

```

chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming$ ./client_tcp
输入要发送的信息: aaa
Error!
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming$

```

客户端的消息会发不出去，进而报错Error!。因为TCP是可靠的有连接的全双工传输协议，需要两侧同时建立连接才可以进行正常消息传递。

## • 服务器如何可以退出循环？

任意键入（由kbhit()函数支持）或Ctrl+C

## (iii) 服务器源代码

注意kbhit()函数未附在下面程序中。

```

1 #include <sys/types.h>
2 #include <sys/socket.h>
3 #include <stdio.h>
4 #include <netinet/in.h>
5 #include <arpa/inet.h>
6 #include <unistd.h>
7 #include <stdlib.h>
8 #include <string.h>
9 #include <error.h>
10 #include <termios.h>
11 #include <unistd.h>

```

```
12 #include <fcntl.h>
13 #include <time.h>
14
15 #define BUF_LEN 2000
16
17 int kbhit(void);
18
19 void generateMsg(char* buffer){
20     printf("收到消息: %s\n", buffer);
21
22     time_t now; /* current time */
23     time(&now);
24     char* pts = (char *)ctime(&now);
25     printf("收到时间: %s\n", pts);
26
27     strcat(pts,buffer);
28     strcpy(buffer,pts);
29     strcat(buffer,"\n");
30 }
31
32 int main(int argc, char *argv[])
33 {
34     struct sockaddr_in fsin;          /* the from address of a client */
35     int msock, ssock;                 /* master & slave sockets */
36     char *service = "50500";
37     char buf[BUF_LEN+1];              /* buffer for one line of text */
38     struct sockaddr_in sin;           /* an Internet endpoint address */
39     int alen;                         /* from-address length */
40     char *pts;                        /* pointer to time string */
41
42     // 创建套接字, 参数: 因特网协议簇(family), 流套接字, TCP协议
43     // 返回: 要监听套接字的描述符或INVALID_SOCKET
44     msock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
45
46     memset(&sin, '\0', sizeof(sin)); // 从&sin开始的长度为sizeof(sin)的内存清0
47     sin.sin_family = AF_INET; // 因特网地址簇(INET-Internet)
48     sin.sin_addr.s_addr = INADDR_ANY; // 监听所有(接口的)IP地址
49     sin.sin_port = htons((u_short)atoi(service)); // 监听的端口号
50     bind(msock, (struct sockaddr *)&sin, sizeof(sin)); // 绑定监听的IP地址和端口号
51
52     // 建立长度为5的连接请求队列, 并把到来的连接请求加入队列等待处理
53     listen(msock, 5);
54
55     printf("服务器已启动! \n\n");
56 }
```

```

57 while (!kbhit()){ // 检测是否有按键,如果没有则进入循环体执行
58     alen = sizeof(struct sockaddr); // 取到地址结构的长度
59     // 如果在连接请求队列中有连接请求,则接受连接请求并建立连接,返回该连接的套接字
60     // 否则,本语句被阻塞直到队列非空。fsin包含客户端IP地址和端口号
61     ssock = accept(msock, (struct sockaddr *)&fsin, &alen);
62
63     // 第二个参数指向缓冲区,第三个参数为缓冲区大小(字节数),第四个参数一般设置为0
64     // 返回值:(>0)接收到的字节数,(=0)对方已关闭,<0)连接出错
65     int cc = recv(ssock, buf, BUF_LEN, 0);
66     if (cc <= 0)
67         printf("Error!\n"); // 出错或对方关闭(==0)。其后必须关闭套接字sock
68     else if (cc > 0) {
69         buf[cc] = '\0';
70
71         if (argc == 1){ // TCP Echo Enhancement
72             generateMsg(buf);
73         } else {
74             generateEnhancedMsg(buf, (unsigned char *) &(fsin.sin_addr), fsin.
75                 ↪ sin_port);
76         }
77
78         cc = send(ssock, buf, strlen(buf), 0);
79         if (cc <= 0)
80             printf("Server send message error!\n");
81     }
82     close(ssock);
83 }
84 close(msock);
85 return 0;
86 }

```

#### (iv) 客户端源代码

```

1  #include <sys/types.h>
2  #include <sys/socket.h>
3  #include <stdio.h>
4  #include <netinet/in.h>
5  #include <arpa/inet.h>
6  #include <unistd.h>
7  #include <stdlib.h>
8  #include <string.h>
9  #include <error.h>
10
11 #define BUF_LEN 2000

```

```
12
13 int main(int argc, char *argv[])
14 {
15     char    *host = "127.0.0.1";    /* server IP to connect    */
16     char    *service = "50500";    /* server port to connect */
17     struct sockaddr_in sin;    /* an Internet endpoint address */
18     char    buf[BUF_LEN+1];    /* buffer for one line of text */
19     int     sock;    /* socket descriptor    */
20     int     cc;    /* recv character count    */
21
22     // 创建套接字, 参数: 因特网协议簇(family), 流套接字, TCP协议
23     // 返回: 要监听套接字的描述符或INVALID_SOCKET
24     sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
25
26     memset(&sin, 0, sizeof(sin)); // 从&sin开始的长度为sizeof(sin)的内存清0
27     sin.sin_family = AF_INET;    // 因特网地址簇
28     sin.sin_addr.s_addr = inet_addr(host);    // 设置服务器IP地址(32位)
29     sin.sin_port = htons((u_short)atoi(service)); // 设置服务器端口号
30     // 连接到服务器, 第二个参数指向存放服务器地址的结构
31     // 第三个参数为该结构的大小, 返回值为0时表示无错误发生
32     // 返回SOCKET_ERROR表示出错, 应用程序可通过WSAGetLastError()获取相应错误代码
33     int ret = connect(sock, (struct sockaddr *)&sin, sizeof(sin));
34
35     printf("输入要发送的信息: ");
36     scanf("%s", buf);
37     // 第二个参数指向发送缓冲区, 第三个参数为要发送的字节数, 第四个参数一般置0
38     // 返回值为实际发送的字节数, 出错或对方关闭时返回SOCKET_ERROR
39     cc = send(sock, buf, strlen(buf), 0);
40     if (cc <= 0){
41         printf("Error!\n");
42         return 0;
43     }
44
45     printf("\n收到的信息: \n");
46     if (cc <= 0)
47         printf("Error!\n"); // 出错或对方关闭(==0)。其后必须关闭套接字sock
48     else if (cc > 0) {
49         cc = recv(sock, buf, BUF_LEN, 0);
50         buf[cc] = '\0';
51         printf("%s\n", buf);
52     }
53
54     close(sock); // 关闭连接套接字
55
56     printf("按回车键继续...\n");
```

```
57     getchar();
58     return 0;
59 }
```

## 2. TCP Echo增强程序

### (i) 实验要求

在上一实验的基础上，服务器在收到客户端的消息时显示服务器的当前时间、客户端的IP地址、客户端的端口号和客户端发来的信息，并把它们一并返回给客户端。

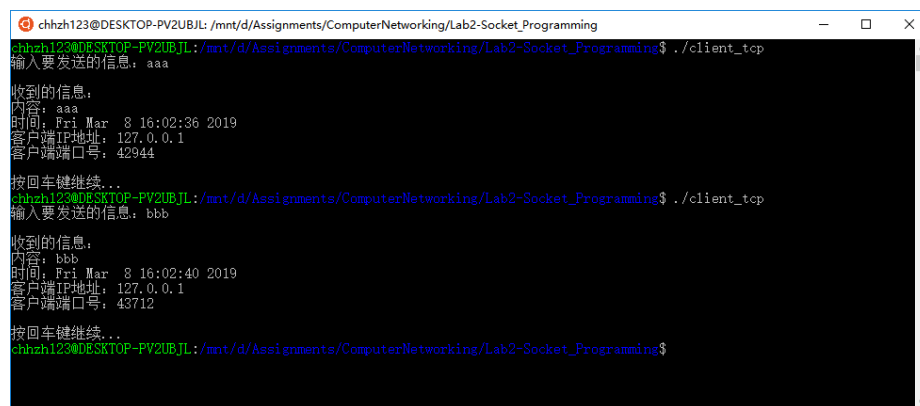
客户端在发送消息后把服务器发回给它的消息显示出来。（客户端程序与上一实验相同，不用修改）

要求服务器直接从`accept()`的参数`fsin`中得到客户端的IP地址和端口号。

注：服务器获取IP地址后要求直接使用`s_un_b`的四个分量得到IP地址，不能使用函数`inet_ntoa()`转换IP地址。

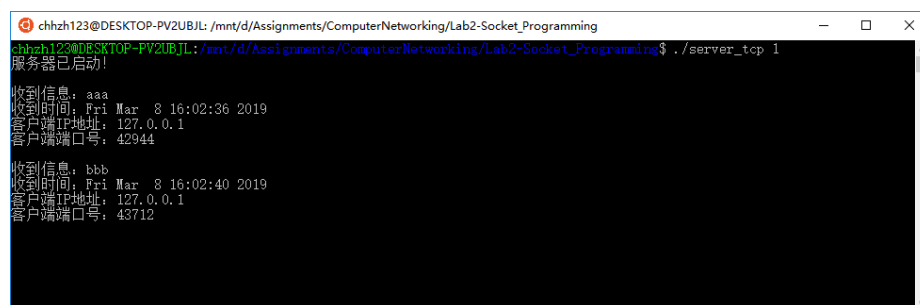
### (ii) 运行结果

#### • 客户端（两次运行）



```
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming$ ./client_tcp
输入要发送的信息: aaa
收到的信息:
内容: aaa
时间: Fri Mar 8 16:02:36 2019
客户端IP地址: 127.0.0.1
客户端端口号: 42944
按回车键继续...
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming$ ./client_tcp
输入要发送的信息: bbb
收到的信息:
内容: bbb
时间: Fri Mar 8 16:02:40 2019
客户端IP地址: 127.0.0.1
客户端端口号: 43712
按回车键继续...
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming$
```

#### • 服务器



```
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming$ ./server_tcp 1
服务器已启动!
收到信息: aaa
收到时间: Fri Mar 8 16:02:36 2019
客户端IP地址: 127.0.0.1
客户端端口号: 42944
收到信息: bbb
收到时间: Fri Mar 8 16:02:40 2019
客户端IP地址: 127.0.0.1
客户端端口号: 43712
```

### (iii) 服务器源代码

只增加了一个消息生成函数，代码如下。其他部分代码与原来的服务器相同，并且通过命

命令行读入参数，如果服务器程序命令行有参数读入，则执行增强版的程序。

```

1 void generateEnhancedMsg(char* buffer, unsigned char *bytes, u_short port)
2 {
3     char buf[BUF_LEN+1];
4     printf("收到信息: %s\n", buffer);
5     sprintf(buf, "内容: %s\n", buffer);
6
7     time_t now; /* current time */
8     time(&now);
9     char* pts = (char *)ctime(&now);
10    printf("收到时间: %s", pts);
11    sprintf(buffer, "时间: %s", pts);
12    strcat(buf, buffer);
13
14    // inet_ntoa
15    snprintf (buffer, sizeof (buf), "客户端IP地址: %d.%d.%d.%d\n",
16             bytes[0], bytes[1], bytes[2], bytes[3]);
17    printf("%s", buffer);
18    strcat(buf,buffer);
19
20    sprintf(buffer, "客户端端口号: %d\n", port);
21    printf("%s", buffer);
22    strcat(buf, buffer);
23
24    printf("\n");
25    strcpy(buffer,buf);
26 }

```

### 3. UDP Echo增强程序

#### (i) 实验要求

修改UDP例程，完成Echo功能，即当客户端发来消息时，服务器显示出服务器的当前时间、客户端的IP、客户端的端口号和客户发来的信息，并把它们一并发回给客户端，客户端然后把它们显示出来。

服务器可以直接从recvfrom()的参数from中得到客户端的IP地址和端口号，并且服务器用sendto()发回给客户端消息时可以直接用该参数from作为参数toAddr。可以使用inet\_ntoa()转换客户端IP地址。

客户端程序的recvfrom()可以直接使用原来sendto使用的sock。该sock已经绑定了客户端的IP地址和端口号，客户端可以直接用来接收数据。

#### (ii) 运行结果

- 客户端（两次运行）

```
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming
chhzh123@DESKTOP-PV2UBJL:/mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming$ ./client_udp
输入消息: aaa
客户端的消息: aaa
客户端IP地址: 127.0.0.1
客户端端口号: 31195
时间: Fri Mar 8 16:31:52 2019
按回车键继续...
chhzh123@DESKTOP-PV2UBJL:/mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming$ ./client_udp
输入消息: bbb
客户端的消息: bbb
客户端IP地址: 127.0.0.1
客户端端口号: 31451
时间: Fri Mar 8 16:31:56 2019
按回车键继续...
chhzh123@DESKTOP-PV2UBJL:/mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming$
```

## • 服务器

```
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming
chhzh123@DESKTOP-PV2UBJL:/mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming$ ./server_udp
服务器启动!
客户端的消息: aaa
客户端IP地址: 127.0.0.1
客户端端口号: 31195
时间: Fri Mar 8 16:31:52 2019
客户端的消息: bbb
客户端IP地址: 127.0.0.1
客户端端口号: 31451
时间: Fri Mar 8 16:31:56 2019
```

## • 只运行客户端程序而不运行服务器程序会出现什么错误，截屏并说明原因。

```
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming
chhzh123@DESKTOP-PV2UBJL:/mnt/d/Assignments/ComputerNetworking/Lab2-Socket_Programming$ ./client_udp
输入消息: aaa
```

客户端的消息可以正常发送，但发送完后会一直等待服务器返回信息。因为UDP是不可靠的无连接的全双工传输协议（数据报），不需两侧同时建立连接就可以消息传递，但导致的结果就是丢包。

## (iii) 服务器源代码

```
1 #include <sys/types.h>
2 #include <sys/socket.h>
3 #include <stdio.h>
4 #include <netinet/in.h>
5 #include <arpa/inet.h>
6 #include <unistd.h>
7 #include <stdlib.h>
8 #include <string.h>
```



```
9  #include <error.h>
10 #include <termios.h>
11 #include <unistd.h>
12 #include <fcntl.h>
13 #include <time.h>
14
15 #define BUFLen 2000
16
17 int kbhit(void);
18
19 void generateEnhancedMsg(char* buffer, unsigned char *bytes, u_short port)
20 {
21     char buf[BUFLen+1];
22     sprintf(buf, "客户端的消息: %s\n", buffer);
23     printf("%s", buf);
24
25     // inet_ntoa
26     snprintf (buffer, sizeof (buf), "客户端IP地址: %d.%d.%d.%d\n",
27             bytes[0], bytes[1], bytes[2], bytes[3]);
28     printf("%s", buffer);
29     strcat(buf, buffer);
30
31     sprintf(buffer, "客户端端口号: %d\n", port);
32     printf("%s", buffer);
33     strcat(buf, buffer);
34
35     time_t now; /* current time */
36     time(&now);
37     char* pts = (char *)ctime(&now);
38     printf("时间: %s\n", pts);
39     sprintf(buffer, "时间: %s", pts);
40     strcat(buf, buffer);
41
42     strcat(buf, "\n");
43     strcpy(buffer, buf);
44 }
45
46 int main(int argc, char *argv[])
47 {
48     char *host = "127.0.0.1"; /* server IP Address to connect */
49     char *service = "50500"; /* server port to connect */
50     struct sockaddr_in sin; /* an Internet endpoint address */
51     struct sockaddr_in from; /* sender address */
52     int fromsize = sizeof(from);
53     char buf[BUFLen+1]; /* buffer for one line of text */
```

```

54     int    sock;                                /* socket descriptor    */
55     int    cc;                                /* recv character count */
56
57     // 创建UDP套接字, 参数: 因特网协议簇(family), 数据报套接字, UDP协议号
58     // 返回: 要监听套接字的描述符或INVALID_SOCKET
59     sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP);
60     printf("服务器启动! \n\n");
61
62     memset(&sin, 0, sizeof(sin));
63     sin.sin_family = AF_INET;
64     sin.sin_addr.s_addr = INADDR_ANY;           // 绑定(监听)所有的接口
65     sin.sin_port = htons((u_short)atoi(service)); // 绑定指定接口
66     // htons -- 主机序(host)转化为网络序(network), 为short类型
67     // 绑定本地端口号 (和本地IP地址)
68     bind(sock, (struct sockaddr *)&sin, sizeof(sin));
69
70     while(!kbhit()){ // 检测是否有按键
71         // 接收客户数据。返回结果: cc为接收的字符数, from中将包含客户IP地址和端口号
72         cc = recvfrom(sock, buf, BUFLen, 0, (struct sockaddr *)&from, &fromsize);
73         if (cc < 0){
74             printf("recv() failed; %d\n", cc);
75             break;
76         } else {
77             buf[cc] = '\0';
78             generateEnhancedMsg(buf, (unsigned char *) &(from.sin_addr), from.
                ↪ sin_port);
79             cc = sendto(sock, buf, strlen(buf), 0, (struct sockaddr *)&from,
                ↪ fromsize);
80         }
81     }
82     close(sock);
83
84     printf("按回车键继续...\n");
85     getchar();
86 }

```

#### (iv) 客户端源代码

```

1  #include <sys/types.h>
2  #include <sys/socket.h>
3  #include <stdio.h>
4  #include <netinet/in.h>
5  #include <arpa/inet.h>
6  #include <unistd.h>
7  #include <stdlib.h>

```

```
8 #include <string.h>
9 #include <error.h>
10
11 #define BUFLen 2000
12
13 int main(int argc, char *argv[])
14 {
15     char *host = "127.0.0.1"; /* server IP to connect */
16     char *service = "50500"; /* server port to connect */
17     struct sockaddr_in toAddr; /* an Internet endpoint address */
18     int tosize = sizeof(toAddr);
19     char buf[BUFLen+1]; /* buffer for one line of text */
20     int sock; /* socket descriptor */
21     int cc; /* recv character count */
22     char *pts; /* pointer to time string */
23     time_t now; /* current time */
24
25     sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP);
26
27     memset(&toAddr, 0, sizeof(toAddr));
28     toAddr.sin_family = AF_INET;
29     toAddr.sin_port = htons((u_short)atoi(service));
30     // htons: 主机序(host)转化为网络序(network), s--short
31     toAddr.sin_addr.s_addr = inet_addr(host);
32     // 如果host为域名, 需要先用函数gethostbyname把域名转化为IP地址
33
34     printf("输入消息: ");
35     scanf("%s", buf);
36     printf("\n");
37
38     cc = sendto(sock, buf, BUFLen, 0, (struct sockaddr *)&toAddr, sizeof(toAddr));
39     if (cc <= 0) {
40         printf("发送失败, 错误号: %d\n", cc);
41     } else {
42         cc = recvfrom(sock, buf, BUFLen, 0, (struct sockaddr *)&toAddr, &tosize);
43         if (cc < 0){
44             printf("recv() failed; %d\n", cc);
45         } else {
46             buf[cc] = '\0';
47             printf("%s", buf);
48         }
49     }
50
51     close(sock);
52 }
```

```
53     printf("按回车键继续...\n");  
54     getchar();  
55 }
```

## 六、完成情况

是否完成以下步骤? (✓完成 ✗未做)

1. [✓]    2. [✓]    3. [✓]

## 七、实验体会

本实验相对比较简单，有了实验一C字符串的铺垫，本实验做起来就比较得心应手，基本没有遇到什么问题。本次代码编译在Linux环境下完成，会发现Linux环境下会方便很多，毕竟Linux内核就是用C写的。而且Linux的命令行UTF-8编码也可以完美支持中文。

总的来说，通过本实验我明白TCP和UDP协议具体是怎么实施的，对C的字符串操作更加熟练了，而且会觉得计算机网络非常有趣，希望之后还可以继续做到这么有趣的实验。