

南昌大学实验报告

姓名：林泓宇

学号：8000115196

邮箱地址：Littlered_Lynn@aliyun.com

专业班级：计科154班

实验日期：2018.5.26

课程名称：Linux程序设计

实验项目名称

Socket It Out

实验目的

The purpose of the lab

- Understanding the mechanism of socket
- Trying to learn some C
- Understanding the process of network programming

实验基础

CentOS 6.6, Oracle VirtualBox

实验步骤

Task 1

Socket it

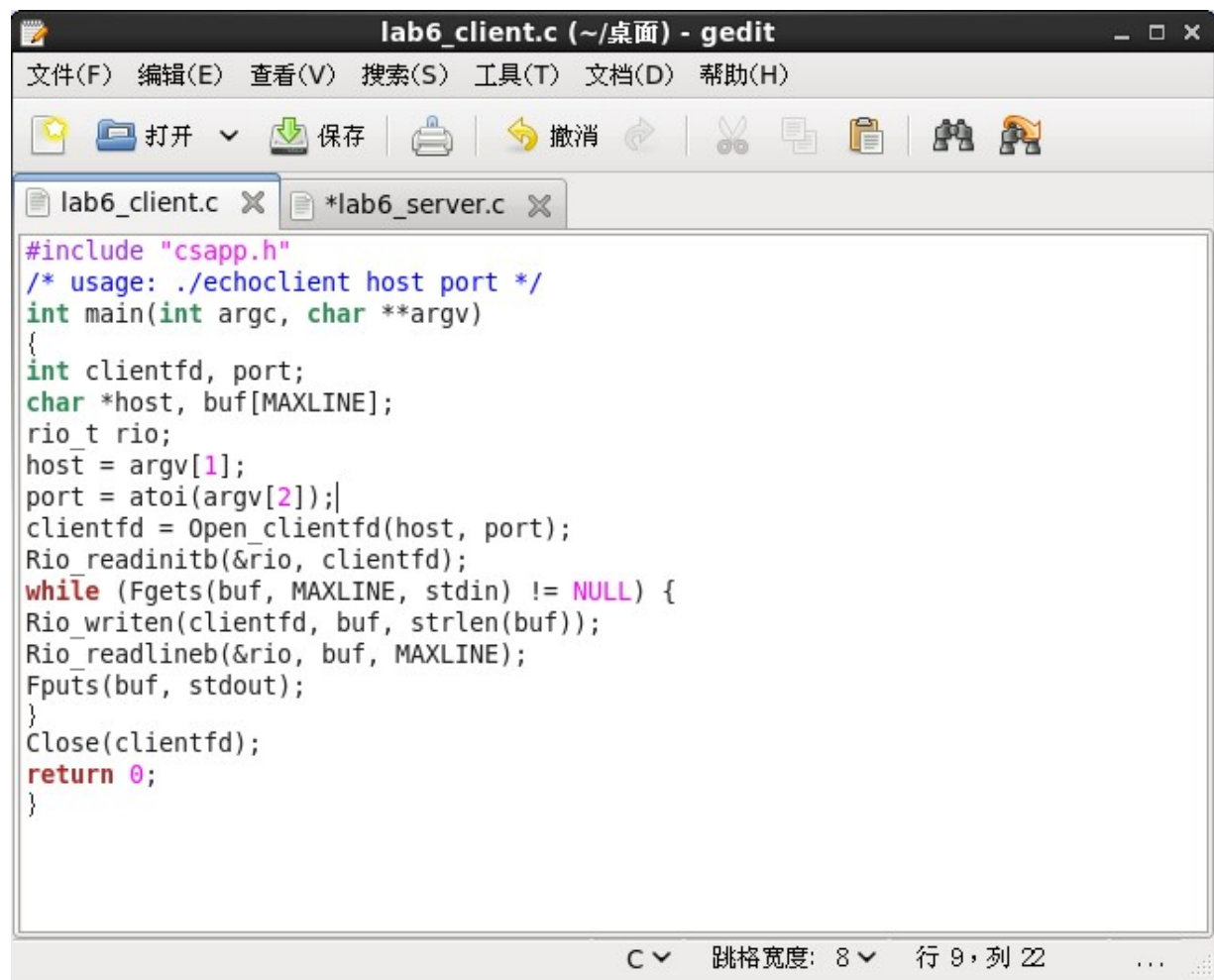
- Program a C/S communication in one host
- Echo the input from client at the server side
- Using socket interface to implement such process
- Make a real pair using Linux C

Task 2

Easier Job on the Way

- Repeat the task in the last slide, using Golang

实验数据或结果



```
lab6_client.c (~/.桌面) - gedit
文件(F) 编辑(E) 查看(V) 搜索(S) 工具(T) 文档(D) 帮助(H)

lab6_client.c x *lab6_server.c x

#include "csapp.h"
/* usage: ./echoclient host port */
int main(int argc, char **argv)
{
    int clientfd, port;
    char *host, buf[MAXLINE];
    rio_t rio;
    host = argv[1];
    port = atoi(argv[2]);
    clientfd = Open_clientfd(host, port);
    Rio_readinitb(&rio, clientfd);
    while (Fgets(buf, MAXLINE, stdin) != NULL) {
        Rio_writen(clientfd, buf, strlen(buf));
        Rio_readlineb(&rio, buf, MAXLINE);
        Fputs(buf, stdout);
    }
    Close(clientfd);
    return 0;
}

C 跳格宽度: 8 行 9, 列 22
```

lab6_server.c (~桌面) - gedit

文件(F) 编辑(E) 查看(V) 搜索(S) 工具(T) 文档(D) 帮助(H)

打开 保存 撤消

lab6_client.c lab6_server.c

```
int main(int argc, char **argv) {
int listenfd, connfd, port, clientlen;
struct sockaddr_in clientaddr;
struct hostent *hp;
char *haddrp;
port = atoi(argv[1]); /* the server listens on a port passed
on the command line */
listenfd = open_listenfd(port);
while (1) {
clientlen = sizeof(clientaddr);
connfd = Accept(listenfd, (SA *)&clientaddr, &clientlen);
hp = Gethostbyaddr((const char *)&clientaddr.sin_addr.s_addr,
sizeof(clientaddr.sin_addr.s_addr), AF_INET);
haddrp = inet_ntoa(clientaddr.sin_addr);
printf("server connected to %s (%s)
\n", hp
-
>h_name, haddrp);
echo(connfd);
Close(connfd);
}
```

C 跳格宽度: 8 行 1, 列 1

首先将老师给出的代码复制过来，写成C文件

```
littlered@localhost:~/桌面
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[littlered@localhost 桌面]$ gcc lab6_client.c -o client
/tmp/ccYy20U4.o: In function `Pthread_create':
lab6_client.c:(.text+0xba4): undefined reference to `pthread_create'
/tmp/ccYy20U4.o: In function `Pthread_cancel':
lab6_client.c:(.text+0xbdb): undefined reference to `pthread_cancel'
/tmp/ccYy20U4.o: In function `Pthread_join':
lab6_client.c:(.text+0xc18): undefined reference to `pthread_join'
/tmp/ccYy20U4.o: In function `Pthread_detach':
lab6_client.c:(.text+0xc4a): undefined reference to `pthread_detach'
/tmp/ccYy20U4.o: In function `Pthread_once':
lab6_client.c:(.text+0xcaa): undefined reference to `pthread_once'
/tmp/ccYy20U4.o: In function `Sem_init':
lab6_client.c:(.text+0xcd2): undefined reference to `sem_init'
/tmp/ccYy20U4.o: In function `P':
lab6_client.c:(.text+0xcfa): undefined reference to `sem_wait'
/tmp/ccYy20U4.o: In function `V':
lab6_client.c:(.text+0xd22): undefined reference to `sem_post'
collect2: ld 返回 1
[littlered@localhost 桌面]$
```

```
littlered@localhost:~/桌面
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
lab6_client.c:(.text+0xcaa): undefined reference to `pthread_once'
/tmp/ccYy20U4.o: In function `Sem_init':
lab6_client.c:(.text+0xcd2): undefined reference to `sem_init'
/tmp/ccYy20U4.o: In function `P':
lab6_client.c:(.text+0xcfa): undefined reference to `sem_wait'
/tmp/ccYy20U4.o: In function `V':
lab6_client.c:(.text+0xd22): undefined reference to `sem_post'
collect2: ld 返回 1
[littlered@localhost 桌面]$ gcc lab6_server.c -o server
lab6_server.c: 在函数 `main'中:
lab6_server.c:3: 错误: `clientaddr'的存储大小未知
lab6_server.c:11: 错误: `SA'未声明(在此函数内第一次使用)
lab6_server.c:11: 错误: (即使在一个函数内多次出现, 每个未声明的标识符在其
lab6_server.c:11: 错误: 所在的函数内也只报告一次。)
lab6_server.c:11: 错误: expected expression before `)' token
lab6_server.c:13: 错误: `AF_INET'未声明(在此函数内第一次使用)
lab6_server.c:15: 警告: 隐式声明与内建函数 `printf'不兼容
lab6_server.c:15:8: 警告: 缺少结尾的 `"' 字符
lab6_server.c:15: 错误: 缺少结尾的 `"' 字符
lab6_server.c:17:2: 警告: 缺少结尾的 `"' 字符
lab6_server.c:17: 错误: 缺少结尾的 `"' 字符
lab6_server.c:17: 错误: `h'未声明(在此函数内第一次使用)
lab6_server.c:19: 错误: expected expression before `>' token
[littlered@localhost 桌面]$
```

编译两个文件，会发现报出大量错误，这里首要问题就是没有"csapp.h"这个文件，并且老师的实验要求中也没有给出，所以只能自行到网上下载，下载之后解压，发现还有

一个捆绑的文件"csapp.c"，并且在这个"csapp.h"的头文件中，包含了老师在实验介绍中提到的一些函数，说明了这个头文件的正确性

```
lab6_client.c X lab6_server.c X csapp.h X
/* $begin csapp.h */
#ifndef __CSAPP_H__
#define __CSAPP_H__

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <ctype.h>
#include <setjmp.h>
#include <signal.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <errno.h>
#include <math.h>
#include <pthread.h>
#include <semaphore.h>
#include <sys/socket.h>
#include <netdb.h>
#include <netinet/in.h>
#include <arpa/inet.h>

/* Default file permissions are DEF_MODE & ~DEF_UMASK */
/* $begin createmasks */
#define DEF_MODE    S_IRUSR|S_IWUSR|S_IRGRP|S_IWGRP|S_IROTH|S_IWOTH
#define DEF_UMASK    S_IWGRP|S_IWOTH
/* $end createmasks */

/* Simplifies calls to bind(), connect(), and accept() */
/* $begin sockaddrdef */
typedef struct sockaddr SA;
/* $end sockaddrdef */

/* Persistent state for the robust I/O (Rio) package */
/* $begin rio_t */
#define RIO_BUFSIZE 8192
typedef struct {
    int rio_fd;                /* Descriptor for this internal buf */
    int rio_cnt;               /* Unread bytes in internal buf */
    char *rio_bufptr;          /* Next unread byte in internal buf */
    char rio_buf[RIO_BUFSIZE]; /* Internal buffer */
} rio_t;
/* $end rio_t */
```



```

int open_clientfd(char *hostname, int port)
{
    int clientfd;
    struct hostent *hp;
    struct sockaddr_in serveraddr;

    if ((clientfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
        return -1; /* check errno for cause of error */

    /* Fill in the server's IP address and port */
    if ((hp = gethostbyname(hostname)) == NULL)
        return -2; /* check h_errno for cause of error */
    bzero((char *) &serveraddr, sizeof(serveraddr));
    serveraddr.sin_family = AF_INET;
    bcopy((char *)hp->h_addr,
        (char *)&serveraddr.sin_addr.s_addr, hp->h_length);
    serveraddr.sin_port = htons(port);

    /* Establish a connection with the server */
    if (connect(clientfd, (SA *) &serveraddr, sizeof(serveraddr)) < 0)
        return -1;
    return clientfd;
}

```

This function opens a connection from the client to the server at hostname:port

```

/* Client/server helper functions */
int open_clientfd(char *hostname, int portno);
int open_listenfd(int portno);

/* Wrappers for client/server helper functions */
int Open_clientfd(char *hostname, int port);
int Open_listenfd(int port);
#include "csapp.c"
#endif /* __CSAPP_H__ */
/* $end csapp.h */

int open_clientfd(char *hostname, int port)
{
    int clientfd;
    struct hostent *hp;
    struct sockaddr_in serveraddr;

    if ((clientfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
        return -1; /* Check errno for cause of error */

    /* Fill in the server's IP address and port */
    if ((hp = gethostbyname(hostname)) == NULL)
        return -2; /* Check h_errno for cause of error */
    bzero((char *) &serveraddr, sizeof(serveraddr));
    serveraddr.sin_family = AF_INET;
    bcopy((char *)hp->h_addr_list[0],
        (char *)&serveraddr.sin_addr.s_addr, hp->h_length);
    serveraddr.sin_port = htons(port);

    /* Establish a connection with the server */
    if (connect(clientfd, (SA *) &serveraddr, sizeof(serveraddr)) < 0)
        return -1;
    return clientfd;
}

```

再次编译，还是报错，提示没有这个头文件，原来是我自己手贱把

```
#include "csapp.h"
```

改成了

```
#include
```

这样会导致一个什么样的情况了，就是这个程序会到系统的头文件库里去找这个"csapp.h"，这个库位于/usr/include之下，这里就直接改回来便可，程序会自动寻找当前目录下的对应头文件来进行引用

```
littlered@localhost:~/桌面
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[littlered@localhost 桌面]$ gcc lab6_client.c -o client
lab6_client.c:1:19: 错误: csapp.h: 没有那个文件或目录
lab6_client.c: 在函数 main'中:
lab6_client.c:6: 错误: MAXLINE'未声明(在此函数内第一次使用)
lab6_client.c:6: 错误: (即使在一个函数内多次出现, 每个未声明的标识符在其
lab6_client.c:6: 错误: 所在的函数内也只报告一次。)
lab6_client.c:7: 错误: 'rio_t'未声明(在此函数内第一次使用)
lab6_client.c:7: 错误: expected ';' before 'rio'
lab6_client.c:11: 错误: 'rio'未声明(在此函数内第一次使用)
lab6_client.c:12: 错误: 'stdin'未声明(在此函数内第一次使用)
lab6_client.c:12: 错误: NULL'未声明(在此函数内第一次使用)
lab6_client.c:13: 警告: 隐式声明与内建函数 'strlen'不兼容
lab6_client.c:15: 错误: 'stdout'未声明(在此函数内第一次使用)
[littlered@localhost 桌面]$
```



lab6_client.c (~桌面) - gedit

文件(F) 编辑(E) 查看(V) 搜索(S) 工具(T) 文档(D) 帮助(H)

打开 保存 撤消

lab6_client.c lab6_server.c csapp.h csapp.c

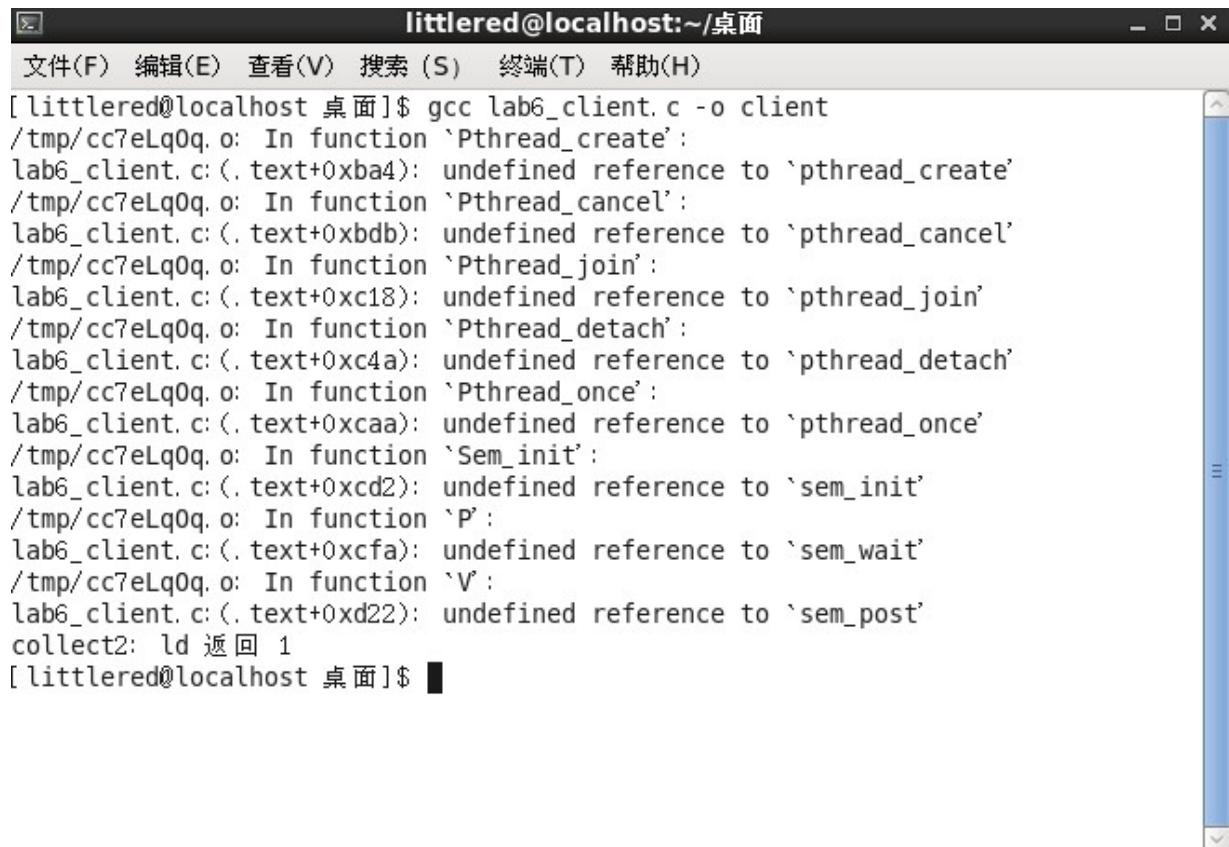
```
#include "csapp.h"
/* usage: ./echoclient host port */
int main(int argc, char **argv)
{
    int clientfd, port;
    char *host, buf[MAXLINE];
    rio_t rio;
    host = argv[1];
    port = atoi(argv[2]);
    clientfd = Open_clientfd(host, port);
    Rio_readinitb(&rio, clientfd);
    while (Fgets(buf, MAXLINE, stdin) != NULL) {
        Rio_writen(clientfd, buf, strlen(buf));
        Rio_readlineb(&rio, buf, MAXLINE);
        Fputs(buf, stdout);
    }
    Close(clientfd);
    return 0;
}
```

C 跳格宽度: 8 行 1, 列 19

然后进行编译，可以看到，头文件的问题已经没有了，但是又出现了新的问题，提示线程相关的错误，这个错误我到网上去查了一下，是因为本程序中含有线程相关的语句，在编译的时候需要在后面加上一条语句，成为

```
gcc lab6_client.c -o client -lpthread
```

这样的形式

A terminal window titled 'littlered@localhost:~/桌面' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command 'gcc lab6_client.c -o client' and a series of linker errors. The errors indicate that several pthread functions are undefined: pthread_create, pthread_cancel, pthread_join, pthread_detach, pthread_once, sem_init, sem_wait, and sem_post. The errors are listed with their locations in the object file. The terminal ends with 'collect2: ld 返回 1' and the prompt '[littlered@localhost 桌面]\$'.

再看服务器端的代码，也是报错，这里错误很愚蠢，是因为复制过来的代码出现了一些格式问题，自动换了行，编译器没能识别出来，退格还原就可以了

***lab6_server.c (~ /桌面) - gedit**

文件(F) 编辑(E) 查看(V) 搜索(S) 工具(T) 文档(D) 帮助(H)

打开 保存 撤消 剪切 粘贴 打印 窗口 帮助

lab6_client.c *lab6_server.c csapp.h csapp.c

```
#include "csapp.h"
int main(int argc, char **argv) {
    int listenfd, connfd, port, clientlen;
    struct sockaddr_in clientaddr;
    struct hostent *hp;
    char *haddrp;
    port = atoi(argv[1]); /* the server listens on a port passed
on the command line */
    listenfd = open_listenfd(port);
    while (1) {
        clientlen = sizeof(clientaddr);
        connfd = Accept(listenfd, (SA *)&clientaddr, &clientlen);
        hp = Gethostbyaddr((const char *)&clientaddr.sin_addr.s_addr,
sizeof(clientaddr.sin_addr.s_addr), AF_INET);
        haddrp = inet_ntoa(clientaddr.sin_addr);
        printf("server connected to %s (%s)
\n", hp
-
>h_name, haddrp);
        echo(connfd);
        Close(connfd);
    }
}
```

C 跳格宽度: 8 行 1, 列 19

littlered@localhost:~/桌面

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
lab6_client.c(.text+0xc18): undefined reference to `pthread_join'
/tmp/cc7eLq0q.o: In function `Pthread_detach':
lab6_client.c(.text+0xc4a): undefined reference to `pthread_detach'
/tmp/cc7eLq0q.o: In function `Pthread_once':
lab6_client.c(.text+0xcaa): undefined reference to `pthread_once'
/tmp/cc7eLq0q.o: In function `Sem_init':
lab6_client.c(.text+0xcd2): undefined reference to `sem_init'
/tmp/cc7eLq0q.o: In function `P':
lab6_client.c(.text+0xcfa): undefined reference to `sem_wait'
/tmp/cc7eLq0q.o: In function `V':
lab6_client.c(.text+0xd22): undefined reference to `sem_post'
collect2: ld 返回 1
[littlered@localhost 桌面]$ gcc lab6_client.c -o client -lpthread
[littlered@localhost 桌面]$ gcc lab6_server.c -o server -lpthread
lab6_server.c:16:8: 警告: 缺少结尾的 " 字符
lab6_server.c: 在函数 main'中:
lab6_server.c:16: 错误: 缺少结尾的 " 字符
lab6_server.c:18:2: 警告: 缺少结尾的 " 字符
lab6_server.c:18: 错误: 缺少结尾的 " 字符
lab6_server.c:18: 错误: h'未声明(在此函数内第一次使用)
lab6_server.c:18: 错误:(即使在一个函数内多次出现, 每个未声明的标识符在其
lab6_server.c:18: 错误: 所在的函数内也只报告一次。)
lab6_server.c:20: 错误: expected expression before '>' token
[littlered@localhost 桌面]$
```

lab6_server.c (~桌面) - gedit

文件(F) 编辑(E) 查看(V) 搜索(S) 工具(T) 文档(D) 帮助(H)

打开 保存 撤消

lab6_server.c

```
#include "csapp.h"
int main(int argc, char **argv) {
int listenfd, connfd, port, clientlen;
struct sockaddr_in clientaddr;
struct hostent *hp;
char *haddrp;
port = atoi(argv[1]); /* the server listens on a port passed
on the command line */
listenfd = open_listenfd(port);
while (1) {
clientlen = sizeof(clientaddr);
connfd = Accept(listenfd, (SA *)&clientaddr, &clientlen);
hp = Gethostbyaddr((const char *)&clientaddr.sin_addr.s_addr,
sizeof(clientaddr.sin_addr.s_addr), AF_INET);
haddrp = inet_ntoa(clientaddr.sin_addr);
printf("server connected to %s (%s)\n", hp->h_name, haddrp);
echo(connfd);
Close(connfd);
}
}
```

C 跳格宽度: 8 行 16, 列 36

这里继续报错，说是没有定义echo这个方法，我找了一下老师给的实验需求，发现给出了这个echo函数，复制过来进行定义，就可以通过编译了


```

void echo(int connfd)
{
    size_t n;
    char buf[MAXLINE];
    rio_t rio;

    Rio_readinitb(&rio, connfd);
    while((n = Rio_readlineb(&rio, buf, MAXLINE)) != 0) {
        printf("server received %d bytes\n", n);
        Rio_writen(connfd, buf, n);
    }
}

```

```

lab6_server.c (~/.桌面) - gedit
文件(F) 编辑(E) 查看(V) 搜索(S) 工具(T) 文档(D) 帮助(H)

#include "csapp.h"
void echo(int connfd)
{
    size_t n;
    char buf[MAXLINE];
    rio_t rio;
    Rio_readinitb(&rio, connfd);
    while((n = Rio_readlineb(&rio, buf, MAXLINE)) != 0) {
        printf("server received %d bytes\n", n);
        Rio_writen(connfd, buf, n);
    }
}

int main(int argc, char **argv) {
    int listenfd, connfd, port, clientlen;
    struct sockaddr_in clientaddr;
    struct hostent *hp;
    char *haddrp;
    port = atoi(argv[1]); /* the server listens on a port passed
on the command line */
    listenfd = open_listenfd(port);
    while (1) {
        clientlen = sizeof(clientaddr);
        connfd = Accept(listenfd, (SA *)&clientaddr, &clientlen);
    }
}

```

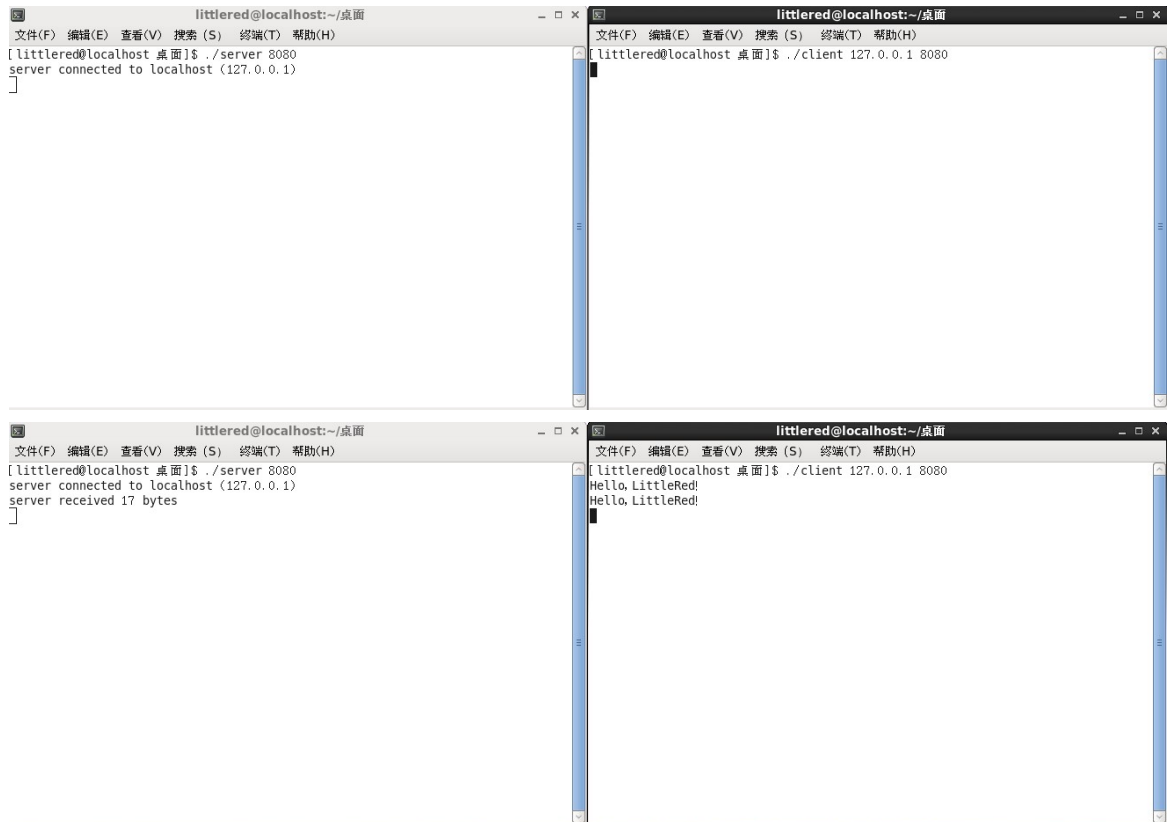
C 跳格宽度: 8 行 12, 列 2

```
littlered@localhost:~/桌面
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
/tmp/cc7eLq0q.o: In function `Sem_init':
lab6_client.c:(.text+0xcd2): undefined reference to `sem_init'
/tmp/cc7eLq0q.o: In function `P':
lab6_client.c:(.text+0xcfa): undefined reference to `sem_wait'
/tmp/cc7eLq0q.o: In function `V':
lab6_client.c:(.text+0xd22): undefined reference to `sem_post'
collect2: ld 返回 1
littlered@localhost 桌面]$ gcc lab6_client.c -o client -lpthread
littlered@localhost 桌面]$ gcc lab6_server.c -o server -lpthread
lab6_server.c:16:8: 警告：缺少结尾的 " 字符
lab6_server.c: 在函数 `main'中:
lab6_server.c:16: 错误：缺少结尾的 " 字符
lab6_server.c:18:2: 警告：缺少结尾的 " 字符
lab6_server.c:18: 错误：缺少结尾的 " 字符
lab6_server.c:18: 错误：`h'未声明(在此函数内第一次使用)
lab6_server.c:18: 错误：(即使在一个函数内多次出现，每个未声明的标识符在其
lab6_server.c:18: 错误：所在的函数内也只报告一次。)
lab6_server.c:20: 错误：expected expression before `>' token
littlered@localhost 桌面]$ gcc lab6_server.c -o server -lpthread
/tmp/cckWzLHV.o: In function `main':
lab6_server.c:(.text+0x1493): undefined reference to `echo'
collect2: ld 返回 1
littlered@localhost 桌面]$ gcc lab6_server.c -o server -lpthread
littlered@localhost 桌面]$
```

打开两个终端，一个终端运行服务端，一个终端运行客户端，可以看到，已经成功建立连接，并且可以传输数据，但是这个运行结果并不美观，跟老师给的要求也不太一样：

1. 客户端输入了信息，会自动重复输出一遍用户的输入

2. 服务器端不能显示用户输入信息的内容，只是给出一个信息大小



要解决这两个问题，我首先仔细阅读了一下源程序，发现了

```
Fputs(buf, stdout)
```

这条语句就是导致重复输出一遍用户输入的始作俑者，于是把它注释掉

```
//Fputs(buf, stdout);
```

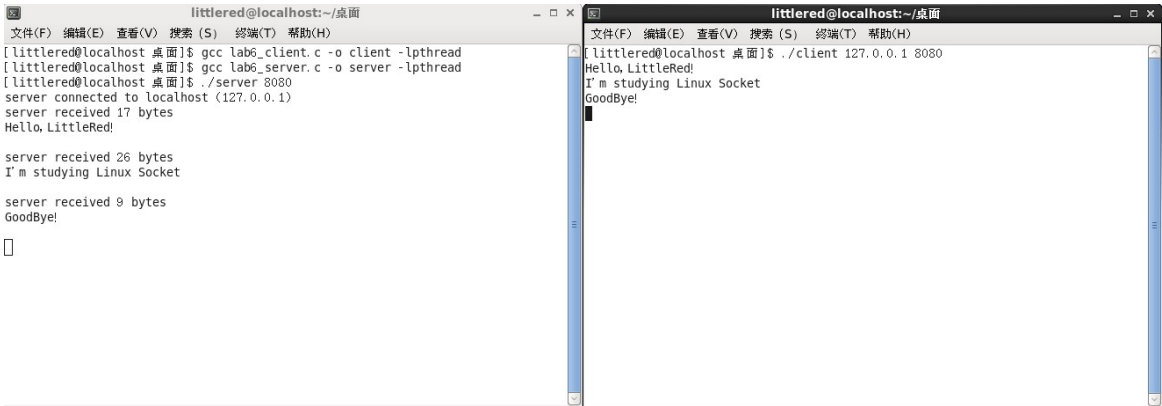
同时，我在服务器端文件找到输出客户端传过来信息大小的部分，在里面加上一句

```
printf("%s\n", buf)
```

其中buf就是客户端输入的文字内容，这样就解决了上述两个问题了

```
while((n = Rio_readlineb(&rio, buf, MAXLINE)) != 0) {  
    printf("server received %d bytes\n", n);  
    printf("%s\n", buf);  
    Rio_writen(connfd, buf, n);  
}
```

重新运行，发现结果和预计的一样



接下来就是使用Golang语言完成Socket编程了，关于这个任务，我在网上查阅了一些资料，最后决定参考[golang socket 服务端与客户端](#)内的写法。

原 golang socket 服务端与客户端

2017年03月18日 10:01:21

阅读数：498

服务端代码如下：

```
package main

import (
    "fmt"
    "net"
    "log"
    "os"
    "bytes"
)

func main() {
```

但是这位兄贵的程序中，是由程序预先在客户端输入内容进行传送，并且只能传送一次数据，所以并不符合老师的要求，但是他的服务端和客户端的连接创建是非常清晰且正确的，所以应用过来，自己修改了一些部分，代码如下，这里我的主要思想就是实现一个简单的函数嵌套调用，在连接建立函数中设置一个send函数，这个send函数就由用户端自己输入内容，而在main函数中，不停的循环传送数据，并设置一个flag变量来确定是否结束循环



```
package main

import (
    "fmt"
    "net"
    "os"
)

func sender(conn net.Conn) {
    var words string
    fmt.Scanln(&words)
    conn.Write([]byte(words))
}

func buildConnetion(){
    server := "127.0.0.1:9999"
    tcpAddr, err := net.ResolveTCPAddr("tcp4", server)
    if err != nil {
        fmt.Fprintf(os.Stderr, "Fatal error: %s", err.Error())
        os.Exit(1)
    }

    conn, err := net.DialTCP("tcp", nil, tcpAddr)
    if err != nil {
        fmt.Fprintf(os.Stderr, "Fatal error: %s", err.Error())
        os.Exit(1)
    }

    sender(conn)
    return
}

func main() {
    var connect_Flag int
    connect_Flag = 1
    for connect_Flag != 0{
        buildConnetion()
        fmt.Println("End by inputing 0")
        fmt.Scanln(&connect_Flag)
    }
}
```

```
lab6_client.go X lab6_server.go X
package main

import (
    "fmt"
    "net"
    "log"
    "os"
    "bytes"
)

func main() {
    //建立socket, 监听端口
    netListen, err := net.Listen("tcp", "127.0.0.1:9999")
    CheckError(err)
    defer func(l net.Listener) {
        fmt.Println("Close")
        l.Close()
    }(netListen)
    Log("Waiting for clients")
    for {
        conn, err := netListen.Accept()
        if err != nil {
            continue
        }
        go handleConnection(conn)
    }
}
//处理连接
func handleConnection(conn net.Conn) {
    buffer := make([]byte, 2048)

    for {
        n, err := conn.Read(buffer)
        if err != nil {
            return
        }
        Log(conn.RemoteAddr().String(), "\n", string(buffer[:n]))
        if len(string(buffer[:n])) > 25 {
            sender(conn)
        }
    }
}
```



```

func sender(conn net.Conn) {
    var buffer bytes.Buffer
    //var sl []string
    buffer.WriteString("<?xml version=\"1.0\" encoding=\"GBK\"?>")
    buffer.WriteString("<message>")
    buffer.WriteString("<head>")
    buffer.WriteString("<field name=\"ReceiveTime\">112823</field>")
    buffer.WriteString("<field name=\"ReceiveDate\">20151101</field>")
    buffer.WriteString("</head>")
    buffer.WriteString("<body>")
    buffer.WriteString("<field name=\"Host\">20151101</field>")
    buffer.WriteString("</body>")
    buffer.WriteString("</message>")

    Log(buffer.Bytes())
    Log("地址为—" + conn.RemoteAddr().String())
    //conn.Write([]byte(strings.Join(sl, "")))

    // ar := []byte {1, 1,1, 1}
    //for i:= 0;i< len(buffer.Bytes()); i++ {
    //    ar = append(ar,buffer.Bytes()[i])
    //}
    //Log(ar)
    Log(buffer.String())
    conn.Write(buffer.Bytes())
    Log("send over")
}

func Log(v ...interface{}) {
    log.Println(v...)
}

func CheckError(err error) {
    if err != nil {
        fmt.Fprintf(os.Stderr, "Fatal error: %s", err.Error())
        os.Exit(1)
    }
}

```

最后是运行结果，由此可见结果是正确的

```

littlered@localhost:~/桌面
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[littlered@localhost 桌面]$ go run lab6_server.go
2018/05/27 15:24:54 Waiting for clients
2018/05/27 15:25:05 127.0.0.1:35038
Hello
2018/05/27 15:25:18 127.0.0.1:35040
Golang123

```

```

littlered@localhost:~/桌面
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[littlered@localhost 桌面]$ go run lab6_client.go
Hello
End by inputing 0
1
Golang123
End by inputing 0
0
littlered@localhost 桌面$

```

实验思考

这次的实验是做socket有关的编程，分为服务端和客户端两个方面进行，在实验过程中遇到了很多问题，但都通过询问老师和查询资料得以解决，这次的实验不仅让我学会

了基础的linux socket编程知识，也对我在web开发在线聊天室的项目有了很大的启发，非常感谢徐老师能够给我们这一次实验的机会，我也会在今后的学习中不断努力，不断进取。

参考资料

- [1] American, Petersen Richard: Linux Programming[M].