<https://blog.csdn.net/zangyongcan/article/details/52139116>

可通过以下程序测试socket发送和接收时，缓冲区buf的大小：

**server端：**

**struct** ps{

**int** st;

    pthread\_t \*thr;

};

**#define** MAXBUF 131072

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

**int** status = 0;

**void** \***recvsocket**(**void** \*arg) //接收client端socket数据的线程

{

**struct** ps \*p = (**struct** ps \*)arg;

**int** st = p->st;

**char** s[1024];

**char** \*pp = **malloc**(MAXBUF);

**while**(1)

    {

**char** buf[100];

       read(STDIN\_FILENO, buf, **sizeof**(buf));

**memset**(s, 0, **sizeof**(s));

**int** rc = recv(st, pp, MAXBUF, 0);

**printf**("rc =%d\n", rc);

**if**(rc <= 0) //如果返回小于等于0，代表socket已经关闭或者出错

**break**;

    //  printf("%s\n",s);

    }

    pthread\_mutex\_lock(&mutex);

    status = 0;

    pthread\_mutex\_unlock(&mutex);

    pthread\_cancel(\*(p->thr)); //被cancel掉的线程内部没有使用锁。

**return** NULL;

}

**void** \***sendsocket**(**void** \*arg) //发送server端socket数据的线程

{

**int** st = \*(**int** \*)arg;

**char** s[1024];

**while**(1)

    {

**memset**(s, 0, **sizeof**(s));

       read(STDIN\_FILENO, s, **sizeof**(s)); //从键盘读取用户输入信息

       send(st, s, **strlen**(s), 0);

    }

**return** NULL;

}

**int main**(**int** arg, **char** \*args[])

{

**if**(arg < 2)

    {

**return** -1;

    }

**int** port = **atoi**(args[1]);

**int** st = socket(AF\_INET, SOCK\_STREAM, 0); //初始化socket

**int** on = 1;

**if**(setsockopt(st, SOL\_SOCKET, SO\_REUSEADDR,&on, **sizeof**(on)) == -1)

    {

**printf**("setsockoptfailed %s\n", **strerror**(errno));

**return** EXIT\_FAILURE;

    }

**struct** sockaddr\_in addr; //定义一个IP地址的结构

**memset**(&addr, 0, **sizeof**(addr));

    addr.sin\_family = AF\_INET; //设置addr结构的属性定位为TCP/IP地址

    addr.sin\_port = htons(port); //将本地字节顺序转化为网络字节数据

    addr.sin\_addr.s\_addr = htonl(INADDR\_ANY); //INADDR\_ANY代表这个server上所有的地址

    //将IP地址与server程序绑定

**if**(bind(st, (**struct** sockaddr \*)&addr, **sizeof**(addr)) == -1)

    {

**printf**("connectfailed %s\n", **strerror**(errno));

**return** EXIT\_FAILURE;

    }

    //server端开始listen，

**if**(listen(st, 20) == -1)

    {

**printf**("listenfailed %s\n", **strerror**(errno));

**return** EXIT\_FAILURE;

    }

**int** client\_st = 0;//client端socket

//  socklen\_t len = 0;

**struct** sockaddr\_in client\_addr; //表示client端的IP地址

    //void \*p = &client\_addr;

    pthread\_t thrd1, thrd2;

**while**(1)

    {

**memset**(&client\_addr, 0, **sizeof**(client\_addr));

       socklen\_t len = **sizeof**(client\_addr);

       //accept会阻塞，直到有客户端连接过来，accept返回client的socket描述符

       client\_st = accept(st, (**struct** sockaddr \*)&client\_addr,&len);

       pthread\_mutex\_lock(&mutex); //为全局变量加一个互斥锁，防止与线程函数同时读写变量的冲突

       status++;

       pthread\_mutex\_unlock(&mutex); //解锁

**if**(status > 1) //代表这是第二个socket连接，关闭（确保只能连接一个socket）

       {

           close(client\_st);

**continue**;

       }

**if**(client\_st == -1)

       {

**printf**("acceptfailed %s\n", **strerror**(errno));

**return** EXIT\_FAILURE;

       }

**printf**("accept by%s\n", inet\_ntoa(client\_addr.sin\_addr));

**struct** ps ps1;

       ps1.st = client\_st;

       ps1.thr = &thrd2;

       pthread\_create(&thrd1, NULL, recvsocket, &ps1);

       pthread\_detach(thrd1); //设置线程为可分离

//     pthread\_create(&thrd2, NULL, sendsocket, &client\_st);

//     pthread\_detach(thrd2); //设置线程为可分离

    }

    close(st); //关闭server端listen的socket

**return** EXIT\_SUCCESS;

}

**client端：**

**#define** MAXBUF 131072\*4

**void** \***recvsocket**(**void** \*arg) //接收client端socket数据的线程

{

**int** st = \*(**int** \*)arg;

**char** s[1024];

**while**(1)

    {

**memset**(s, 0, **sizeof**(s));

**int** rc = recv(st, s, **sizeof**(s), 0);

**if**(rc <= 0) //如果返回小于等于0，代表socket已经关闭或者出错

       {

**break**;

       }

**printf**("%s\n", s);

    }

**return** NULL;

}

**void** \***sendsocket**(**void** \*arg) //发送server端socket数据的线程

{

**int** st = \*(**int** \*)arg;

**char** s[1024];

**char** \*p = **malloc**(MAXBUF); //

**while**(1)

    {

**memset**(s, 0, **sizeof**(s));

       read(STDIN\_FILENO, s, **sizeof**(s)); //从键盘读取用户输入信息

**int** rc = send(st, p, MAXBUF, 0);

**printf**("rc =%d\n", rc);

    }

**return** NULL;

}

**int main**(**int** arg, **char** \*args[])

{

**if**(arg < 3)

**return** -1;

**int** port = **atoi**(args[2]);

**int** st = socket(AF\_INET, SOCK\_STREAM, 0); //初始化socket

**struct** sockaddr\_in addr; //定义一个IP地址的结构

**memset**(&addr, 0, **sizeof**(addr));

    addr.sin\_family = AF\_INET; //设置结构地址类型为TCP/IP地址

    addr.sin\_port = htons(port); //指定一个端口号：8080，htons:将short类型从host字节类型到net字节类型转化

    addr.sin\_addr.s\_addr = inet\_addr(args[1]); //将字符串类型的IP地址转化为int，赋给addr结构

    //调用connect连接到结构addr指定的IP地址和端口号

**if**(connect(st, (**struct** sockaddr \*)&addr, **sizeof**(addr)) == -1)

    {

**printf**("connectfailed %s\n", **strerror**(errno));

**return** EXIT\_FAILURE;

    }

    pthread\_t thrd1, thrd2;

    pthread\_create(&thrd1, NULL, recvsocket, &st);

    pthread\_create(&thrd2, NULL, sendsocket, &st);

    pthread\_join(thrd1, NULL);

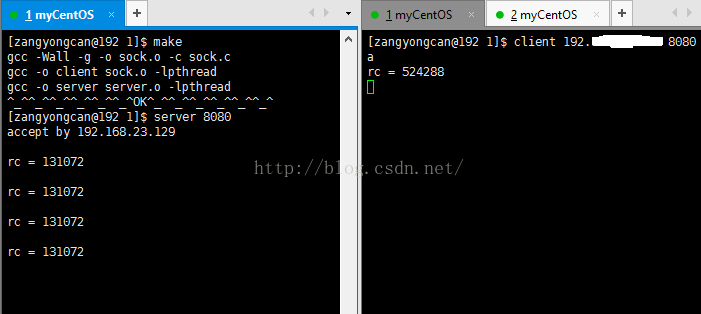
    //pthread\_join(thrd2, NULL);

    close(st); //关闭socket

**return** EXIT\_SUCCESS;

}

运行结果如下：



一次发送了521k的数据，服务器端分4次接收（服务端程序中设置的是每按下回车键一次，接收一次数据）。

经测试，centos64系统的接收端的缓冲区大小是128k。