1、udp通信

1.1 udpServer.hpp

```
#pragma once

#include <iostream>
#include <string>
#include<cerrno>
#include<cstring>
#include<cstdlib>
#include<functional>
#include<unistd.h>
#include<strings.h>
#include<netinet/in.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<arpa/inet.h>
namespace Server
{
    using namespace std;
    static const string defaultIp="0.0.0.0";//默认绑定的ip
    static const int gnum=1024;

    enum {USAGE_ERR=1,SOCKET_ERR ,BIND_ERR};

    typedef function<void(string,uint16_t,string)> func_t;
    class udpServer
    {
    public:
        udpServer(const func_t cb,const uint16_t &port,const string &ip=defaultIp)
        : _callback(cb),_port(port),_ip(ip),_sockfd(-1)
        {}
```

```
    void initServer()
    {
        //1、创建socket
        _sockfd=socket(AF_INET,SOCK_DGRAM,0);//创建套接字
        if(_sockfd==-1)
        {
            cerr<<"socket error: "<<errno<<" : "<<strerror(errno)<<endl;
            exit(SOCKET_ERR);
        }
        cout<<"socket success: "<<" : "<<_sockfd<<endl;
        //2、绑定port ip
        struct sockaddr_in local;//定义一个结构体变量
        bzero(&local,sizeof(local));//将结构体变量填充为0
        local.sin_family=AF_INET;//使用的协议家族
        local.sin_port=htons(_port);//主机序列转换为网络序列
        //一般不会这么绑定ip
        local.sin_addr.s_addr=inet_addr(_ip.c_str());//1、将string类型的ip转换为
uint31_t的ip  2、从主机序列转换为网络序列  inet_addr就搞定
        //local.sin_addr.s_addr=INADDR_ANY;//任意地址绑定 才是服务器的真实写法
        int n=bind(_sockfd,(struct sockaddr*)&local,sizeof(local));
        if(n==-1)
        {
            cerr<<"bind error: "<<errno<<" : "<<strerror(errno)<<endl;
            exit(BIND_ERR);
        }
    }
    void start()
    {
        //服务器的本质其实就是一个死循环
        char buffer[gnum];
        for(;;)
        {
            //读取数据
            struct sockaddr_in peer;  //注意此时是输入输出型 所以只需要ip
            socklen_t len=sizeof(peer);
            //最后两个参数是输出型参数 用来获取是谁向服务端发的数据
            ssize_t s=recvfrom(_sockfd,buffer,sizeof(buffer)-1,0,(struct
sockaddr*)&peer,&len);
            //1、数据是什么 2、谁发的
            if(s>0)
            {
                buffer[s]=0;
```

```
                string clientip=inet_ntoa(peer.sin_addr);//1、网络序列 2、int->点分十进制ip  inet_ntoa就可以搞定
                uint16_t clientport=ntohs(peer.sin_port);
                string message=buffer;
                cout<<clientip<<"["<<clientport<<"]"<<message<<endl;\

                _callback(clientip,clientport,message);
            }

        }
    }
    ~udpServer() {}

  private:
    uint16_t _port;  // 服务器绑定的端口
    string _ip; // 服务器绑定的ip地址
    int _sockfd;
    func_t _callback;//回调
  };
}
```

1.2 udpServer.cc

```
#include "udpServer.hpp"
#include<memory>

using namespace std;
using namespace Server;
static void Usage(string proc)
{
  cout<<"\nUsage:\n\t"<<proc<<" local_port\n\n";
}
void handlerMessage(string clientip,uint16_t clientport,string message)
{
  //就可以对message进行特定的业务处理，而不关心message怎么来的 -----server通信和业务逻辑解耦
}
int main(int argc,char * argv[])
```

```
{
    if(argc !=2)
    {
        Usage(argv[0]);
        exit(USAGE_ERR);
    }
    uint16_t port=atoi(argv[1]);
    //string ip=argv[1];
    std::unique_ptr<udpServer>usvr(new udpServer(handlerMessage,port));
    usvr->initServer();
    usvr->start();
    return 0;
}
```

1.3 udpClient.hpp

```
#pragma once

#include <iostream>
#include <string>
#include<cerrno>
#include<cstring>
#include<cstdlib>
#include<unistd.h>
#include<strings.h>
#include<netinet/in.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<arpa/inet.h>

namespace Client
{
    using namespace std;
    class udpClient
    {
        public:
        udpClient(const string &serverip,const uint16_t &port)
        :_serverip(serverip),_serverport(port),_sockfd(-1)
        {
```

```
    }
    void initClient()
    {
        //1、创建socket
        _sockfd=socket(AF_INET,SOCK_DGRAM,0);
        if(_sockfd == -1)
        {
            cerr<<"socket error: "<<errno<<" : "<<strerror(errno) << endl;
            exit(2);
        }
        cout<<"socket success: "<<" : "<<_sockfd<<endl;
        //2、 client需要bind 但是不需要程序员自己显式的bind 有os自动形成端口进行
bind

    }
    void run()
    {
        struct sockaddr_in server;
        memset(&server,0,sizeof(server));
        server.sin_family=AF_INET;
        server.sin_addr.s_addr=inet_addr(_serverip.c_str());
        server.sin_port=htons(_serverport);
        string message;
        while(!_quit)
        {
            cout<<"Please Enter# ";
            cin>>message;

            sendto(_sockfd,message.c_str(),message.size(),0,(struct
sockaddr*)&server,sizeof(server));
        }
    }
    ~udpClient()
    {

    }

    private:
    int _sockfd;
    string _serverip;
    uint16_t _serverport;
```

```
        bool _quit;
    };
}
```

1.4 udpClient.cc

```
#include "udpClient.hpp"
#include<memory>
using namespace Client;
static void Usage(string proc)
{
    cout<<"\nUsage:\n\t"<<proc<<" local_port\n\n";
}
int main(int argc,char *argv[])
{
    if(argc!=3)
    {
        Usage(argv[0]);
        exit(1);
    }

    string serverip=argv[1];
    uint16_t serverport = atoi(argv[2]);
    unique_ptr<udpClient> ucli(new udpClient(serverip,serverport));
    ucli->initClient();
    ucli->run();
    return 0;
}
```