FTP 协议客户端程序源代码如下。

```
#include "conio.h"
#include "iostream.h"
#include "string.h"
#include "winsock2.h"

#pragam comment(lib, "ws2_32")

#define MAX_SIZE  4096
Char CmdBuf[MAX_SIZE];
Char  Command[MAX_SIZE];
Char ReplyMsg[MAX_SIZE];

Int nReplyCode;
Bool bConnected = false;
SOCKET SocketControl;
SOCKET SocketData

//接收 FTP 服务器应答
Bool RecvReply()
{

    //通过控制连接接受应答信息
    Int nRev =0;
    Memset(ReplyMsg, 0,MAX_SIZE);
    nRecv = recv(SocketControl, ReplyMsg, MAX_SIZE, 0)
    if(nRecv == SOCKET_ERROR)
    {
        Cout<<endl << "socket recv failed!"<<endl;
        Closesocket(SocketControl);
        Return false;
    }
   //获取应答码及应答信息
   If(nRecv >4)
    {
        Char *ReplyCodes = new char[3];
        Memset(replyCodes, 0,3);
        Memcpy(ReplyCodes, ReplyMsg, 3);
        nReplyCode = atoi(ReplyCodes);
    }
    Return True;
}

//向 FTP 服务器发送命令
Bool SendCommand()
{
   //通过控制连接发送命令
    Int nSend;
    nSend = send(SocketControl, command, strlen(command),0);
    if(nSend == SOCKET_ERROR)
    {
        Printf("SocketControl create error:%d\n",WSAGetLastError());
```

```
        Return false;
    }
    Return true;
}

//建立数据连接
Bool DataConnect(char * ServerIpAddr)
{

    //向 FTP 服务器发送 PASV 命令
    Memset(command, 0, MAX_SIZE);
    Memcpy(command, "PASV", strlen("PASV"));
    Memcpy("command + strlen("PASV"),"\r\n",2);
    If(!sendCommand())
    {
        Return false;
    }
    //获得 PASV 命令的应答消息
    If(recvReply())
    {
        If(nReplyCode != 227)
        {
            Printf("PASV 命令应答错误！");
            Closesocket(socketControl);
            Return false;
        }
    }

    //解析 PASV 命令和应答消息
    Char *part[6];
    if (strtok(replyMsg,"("))
    {
        For (int  I =0; I <5; i++)
        {
            Part[i]  = strtok(NULL,",");
            If(!part[i])
            {
                Return false;
            }
        }
        Part[5]  = strtok(NULL, ")");
        If(!part[5])
            Return false;
    }
    Else
       Return false

//获得 FTP 服务器的数据端口号
Unsigned short serverPort;
serverPort = unsigned short((atoi(part[4]) << 8) + atoi(part[5]));
//创建数据 SOCKET
SocketData = socket(AF_INET,SOCK_STREAM,0);
if(SocketData = INVALID_SOCKET)
```

```
{
    Printf("data socket creat error: %d", WSAGetLastError());
    Return false;
}

Sockaddr_in server_addr;
Memset(&server_addr, 0, sizeof(server_addr));
Server_addr.sin_family = AF_INET;
Server_addr.sin_port = htons(severPort);
Server_addr.sin_addr.s_un.s_addr = inet_addr(serverIpAddr);

//与 FTP 服务器发送建立数据 TCP 连接请求
Int nConnect = connect(SocketData, (sockaddr *)&server_addr,sizeof(server_addr));
If(nConnect == SOCKET_ERROR)
{
    Printf("create data TCP connection error : %d\n", WSAGetLastError());
    Return false;
}
Return true;
}


Void main(int argc, char *argv[])
{
  //检查命令行参数
  If(argc != 2)
  {
      Printf("please input param as the following: ftpclient  ftpIPaddr\n");
       Return ;
   }

   WSADATA WSAData;
   If((WSAStartup(MAKEWORD(2,2), &WSAData)!=0)
   {
       Printf("WSAStartup error!\r\n");
       Return;
   }

//创建控制连接 socket
SocketControl = socket(AF_INET,SOCK_STREAM,0);
If(SocketControl == INVALID_SOCKET)
{
   Print("creat TCP Control socket error!");
   Return;
}

//定义 FTP 服务器控制连接地址和端口号
socketaddr_in server_addr;
memset(&server_addr,0,sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(21);
server_addr.sin_addr.s_un.s_addr = inet_addr[argv[1]];
```

```
//向 FTP 服务器发送控制连接请求
Printf("FTP control connect……");
int nConnect = connect(SockControl, (sockaddr *) &server_addr,sizeof(server_addr));
if(nConnect == SOCKET_ERROR)
{
   Printf("client could not establish the FTP control connection with
               server\n");
   Return;
}

//获取控制连接上的应答消息
If(recvReply())
{
     If(nReplyCode == 220)   //判断应答 Code
        Printf("%s \n", replyMsg);
     Else
     {
        Printf("the reply msg is error\n");
        Closesocket(socketControl);
        Return ;
      }
}

//向服务器发送 USER 命令
Printf("FTP->USER:");
Memset(cmdBuf, 0, MAX_SIZE);
gets(cmdBuf,MAX_SIZE)  //输入用户名并保存

Memset(command, 0, MAX_SIZE);
Memcpy(command, "USER", strlen("USER"));
Memcpy(command + strlen("USER"), cmdBuf, strlen(cmdBuf));
Memcpy(command + strlen("USER")+strlen(cmdBuf),"\r\n",2);
If(!sendCommand())
    Return;

//获得 USER 命令的应答信息
If(recvReply())
{
   If(nReplyCode == 230 || nReplyCode ==331)
      Printf("%s", ReplyMsg);
   Else
   {
      Printf("USER 命令应答错误\n");
      Closesocket(SocketControl);
      Return
   }
}

If(nReplyCode == 331)
{
    //向 FTP 服务器发送 PASV 命令
    Printf("FTP > PASV:");
    Memset(cmdBuf,0,MAX_SIZE);
```

```
    For(int I = 0; i<MAX_SIZE; i++)
    {
        cmdBuf[i]  = getch(); //输入用户密码
        if(cmdBuf[i]  == '\r')
        {
           cmdBuf[i] = '\0';
           break;
       }
       Else
          Printf("   *  \r\n");
    }
    Memset(command, 0, MAX_SIZE);
    Memcpy(command, "PASV", strlen("PASV"));
    Memcpy(command + strlen("PASS"), cmdBuf, strlen(cmdBuf));
    Memcpy(command + strlen("PASS")+strlen(cmdBuf),"\r\n",2);

//获得 PASV 命令的应答信息
If(recvReply())
{
   If(nReplyCode == 230)
         Printf("%s", ReplyMsg);
   Else
   {
      Printf("PASV 命令应答错误\n");
      Closesocket(socketControl);
      Return
   }
}
}

//向 FTP 服务器发送 LIST 命令
Printf ( “ftp>LIST\r\n”) ;
Char ftpserver[MAX_SIZE];
Memset(ftpserver,0,MAX_SIZE);
Memcpy(ftpserver,argv[1],strlen(argv[1]));
If(!DataConnect(ftpserver))
   Return;
Memset(Command,0,MAX_SIZE);
Memcpy(command,”LIST”,strlen(“LIST”));
Memcpy(command+strlen(“LIST”),”\r\n”,2);
If(!sendcommand())
   Return;

//获得 LIST 命令的应答信息
If(RecvReply())
{
   If(nRelyCode == 125|| nRelyCode == 150|| nRelyCode == 226)
     Cout << ReplyMsg;
   Else
   {
      Printf( “LIST 命令应答错误! \r\n”);
      Closesocket(socketcontrol);
      Return;
```

```
    }
}

//获取 LIST 命令的目录信息
Int nRecv;
Char ListBuffer[MAX_SIZE];
While(true)
{
   Memeset(ListBuffer, 0, MAX_SIZE);
   nRecv = recv(SocketData,ListBuffer,MAX_SIZE, 0);
   if(nRecv == Socket_error)
   {
      Printf( “数据接收错误!\r\n”);
      Closesocket(SocketData);
      Return;
   }
   If (nRecv <=0)
     Break;
   Cout <<ListBuffer;
}

Closesocket(SocketData);

//获取 LIST 命令的应答信息
If(RecvReply())
{
   If(nRelyCode == 226)
     Cout << ReplyMsg;
   Else
   {
     Printf(“LIST 命令应答错误!\r\n”);
      Closesocket(SocketControl);
      Return;
   }
}


//向 FTP 服务器发送 quit 命令
Printf("FTP->QUIT:");
Memset(command, 0, MAX_SIZE);
Memcpy(command, "QUIT", strlen("QUIT"));
Memcpy(command + strlen("QUIT"),"\r\n",2);
If(!sendCommand())
    Return;

//获得 quit 命令的应答信息
If(recvReply())
{
    If(nReplyCode ==221)
    {
       Printf("%s", ReplyMsg);
       bConnected = false;
       closesocket(SocketControl);
```

```
            return;
        Else
        {
            Printf("QUIT 命令应答错误\r\n");
            Closesocket(SocketControl);
            Return
        }
    }

    WSACleanup();
    }
```