**FTP 概述**

文件传输协议（FTP）作为网络共享文件的传输协议，在网络应用软件中具有广泛的应用。FTP的目标是提高文件的共享性和可靠高效地传送数据。

在传输文件时，FTP 客户端程序先与服务器建立连接，然后向服务器发送命令。服务器收到命令后给予响应，并执行命令。FTP 协议与操作系统无关，任何操作系统上的程序只要符合 FTP 协议，就可以相互传输数据。本文主要基于 LINUX 平台，对 FTP 客户端的实现原理进行详尽的解释并阐述如何使用 C 语言编写一个简单的 FTP 客户端。

[**回页首**](http://www.ibm.com/developerworks/cn/linux/l-cn-socketftp/#ibm-pcon)

**FTP 协议**

相比其他协议，如 HTTP 协议，FTP 协议要复杂一些。与一般的 C/S 应用不同点在于一般的C/S 应用程序一般只会建立一个 Socket 连接，这个连接同时处理服务器端和客户端的连接命令和数据传输。而FTP协议中将命令与数据分开传送的方法提高了效率。

FTP 使用 2 个端口，一个数据端口和一个命令端口（也叫做控制端口）。这两个端口一般是21 （命令端口）和 20 （数据端口）。控制 Socket 用来传送命令，数据 Socket 是用于传送数据。每一个 FTP 命令发送之后，FTP 服务器都会返回一个字符串，其中包括一个响应代码和一些说明信息。其中的返回码主要是用于判断命令是否被成功执行了。

**命令端口**

一般来说，客户端有一个 Socket 用来连接 FTP 服务器的相关端口，它负责 FTP 命令的发送和接收返回的响应信息。一些操作如“登录”、“改变目录”、“删除文件”，依靠这个连接发送命令就可完成。

**数据端口**

对于有数据传输的操作，主要是显示目录列表，上传、下载文件，我们需要依靠另一个 Socket来完成。

如果使用被动模式，通常服务器端会返回一个端口号。客户端需要用另开一个 Socket 来连接这个端口，然后我们可根据操作来发送命令，数据会通过新开的一个端口传输。

如果使用主动模式，通常客户端会发送一个端口号给服务器端，并在这个端口监听。服务器需要连接到客户端开启的这个数据端口，并进行数据的传输。

下面对 FTP 的主动模式和被动模式做一个简单的介绍。

**主动模式 (PORT)**

主动模式下，客户端随机打开一个大于 1024 的端口向服务器的命令端口 P，即 21 端口，发起连接，同时开放N +1 端口监听，并向服务器发出 “port N+1” 命令，由服务器从它自己的数据端口 (20) 主动连接到客户端指定的数据端口 (N+1)。

FTP 的客户端只是告诉服务器自己的端口号，让服务器来连接客户端指定的端口。对于客户端的防火墙来说，这是从外部到内部的连接，可能会被阻塞。

**被动模式 (PASV)**

为了解决服务器发起到客户的连接问题，有了另一种 FTP 连接方式，即被动方式。命令连接和数据连接都由客户端发起，这样就解决了从服务器到客户端的数据端口的连接被防火墙过滤的问题。

被动模式下，当开启一个 FTP 连接时，客户端打开两个任意的本地端口 (N > 1024 和 N+1) 。

第一个端口连接服务器的 21 端口，提交 PASV 命令。然后，服务器会开启一个任意的端口 (P > 1024 )，返回如“**227 entering passive mode (127,0,0,1,4,18)**”。 它返回了 227 开头的信息，在括号中有以逗号隔开的六个数字，前四个指服务器的地址，最后两个，将倒数第二个乘 256 再加上最后一个数字，这就是 FTP 服务器开放的用来进行数据传输的端口。如得到 227 entering passive mode (h1,h2,h3,h4,p1,p2)，那么端口号是 p1\*256+p2，ip 地址为h1.h2.h3.h4。这意味着在服务器上有一个端口被开放。客户端收到命令取得端口号之后, 会通过 N+1 号端口连接服务器的端口 P，然后在两个端口之间进行数据传输。

**主要用到的 FTP 命令**

FTP 每个命令都有 3 到 4 个字母组成，命令后面跟参数，用空格分开。每个命令都以 "\r\n"结束。

要下载或上传一个文件，首先要登入 FTP 服务器，然后发送命令，最后退出。这个过程中，主要用到的命令有 USER、PASS、SIZE、REST、CWD、RETR、PASV、PORT、QUIT。

USER: 指定用户名。通常是控制连接后第一个发出的命令。“USER gaoleyi\r\n”： 用户名为gaoleyi 登录。

PASS: 指定用户密码。该命令紧跟 USER 命令后。“PASS gaoleyi\r\n”：密码为 gaoleyi。

SIZE: 从服务器上返回指定文件的大小。“SIZE file.txt\r\n”：如果 file.txt 文件存在，则返回该文件的大小。

CWD: 改变工作目录。如：“CWD dirname\r\n”。

PASV: 让服务器在数据端口监听，进入被动模式。如：“PASV\r\n”。

PORT: 告诉 FTP 服务器客户端监听的端口号，让 FTP 服务器采用主动模式连接客户端。如：“PORT h1,h2,h3,h4,p1,p2”。

RETR: 下载文件。“RETR file.txt \r\n”：下载文件 file.txt。

STOR: 上传文件。“STOR file.txt\r\n”：上传文件 file.txt。

REST: 该命令并不传送文件，而是略过指定点后的数据。此命令后应该跟其它要求文件传输的 FTP 命令。“REST 100\r\n”：重新指定文件传送的偏移量为 100 字节。

QUIT: 关闭与服务器的连接。

**FTP 响应码**

客户端发送 FTP 命令后，服务器返回响应码。

响应码用三位数字编码表示：

第一个数字给出了命令状态的一般性指示，比如响应成功、失败或不完整。

第二个数字是响应类型的分类，如 2 代表跟连接有关的响应，3 代表用户认证。

第三个数字提供了更加详细的信息。

第一个数字的含义如下：

1 表示服务器正确接收信息，还未处理。

2 表示服务器已经正确处理信息。

3 表示服务器正确接收信息，正在处理。

4 表示信息暂时错误。

5 表示信息永久错误。

第二个数字的含义如下：

0 表示语法。

1 表示系统状态和信息。

2 表示连接状态。

3 表示与用户认证有关的信息。

4 表示未定义。

5 表示与文件系统有关的信息。

**Socket 编程的几个重要步骤**

Socket 客户端编程主要步骤如下：

1. socket() 创建一个 Socket
2. connect() 与服务器连接
3. write() 和 read() 进行会话
4. close() 关闭 Socket

Socket 服务器端编程主要步骤如下：

1. socket() 创建一个 Socket
2. bind()
3. listen() 监听
4. accept() 接收连接的请求
5. write() 和 read() 进行会话
6. close() 关闭 Socket

[**回页首**](http://www.ibm.com/developerworks/cn/linux/l-cn-socketftp/#ibm-pcon)

**实现 FTP 客户端上传下载功能**

下面让我们通过一个例子来对 FTP 客户端有一个深入的了解。本文实现的 FTP 客户端有下列功能：

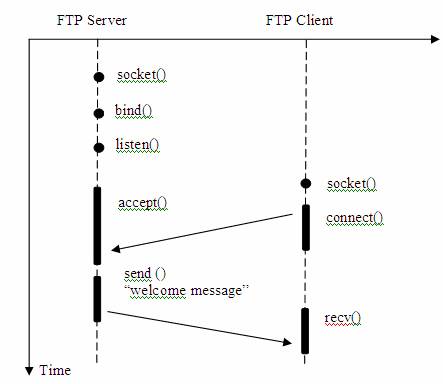
1. 客户端和 FTP 服务器建立 Socket 连接。
2. 向服务器发送 USER、PASS 命令登录 FTP 服务器。
3. 使用 PASV 命令得到服务器监听的端口号，建立数据连接。
4. 使用 RETR/STOR 命令下载/上传文件。
5. 在下载完毕后断开数据连接并发送 QUIT 命令退出。

本例中使用的 FTP 服务器为 filezilla。在整个交互的过程中，控制连接始终处于连接的状态，数据连接在每传输一个文件时先打开，后关闭。

**客户端和 FTP 服务器建立 Socket 连接**

当客户端与服务器建立连接后，服务器会返回 220 的响应码和一些欢迎信息。

**图 1. 客户端连接到服务器端**



**清单 1. 客户端连接到 FTP 服务器，接收欢迎信息**

SOCKET control\_sock;

struct hostent \*hp;

struct sockaddr\_in server;

memset(&server, 0, sizeof(struct sockaddr\_in));

/\* 初始化socket \*/

control\_sock = socket(AF\_INET, SOCK\_STREAM, 0);

hp = gethostbyname(server\_name);

memcpy(&server.sin\_addr, hp->h\_addr, hp->h\_length);

server.sin\_family = AF\_INET;

server.sin\_port = htons(port);

/\* 连接到服务器端 \*/

connect(control\_sock,(struct sockaddr \*)&server, sizeof(server));

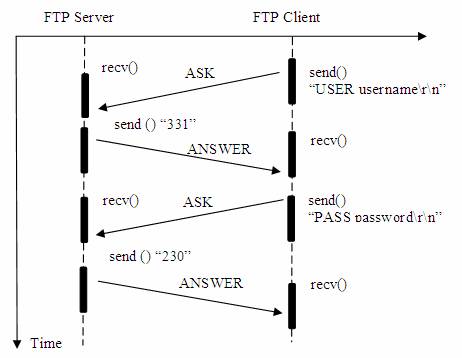
/\* 客户端接收服务器端的一些欢迎信息 \*/

read(control\_sock, read\_buf, read\_len);

**客户端登录 FTP 服务器**

当客户端发送用户名和密码，服务器验证通过后，会返回 230 的响应码。然后客户端就可以向服务器端发送命令了。

**图 2. 客户端登录 FTP 服务器**



**清单 2. 客户端发送用户名和密码，登入 FTP 服务器**

/\* 命令 ”**USER username\r\**n” \*/

sprintf(send\_buf,"USER %s\r\n",username);

/\*客户端发送用户名到服务器端 \*/

write(control\_sock, send\_buf, strlen(send\_buf));

/\* 客户端接收服务器的响应码和信息，正常为 ”**331 User name okay, need password.”** \*/

read(control\_sock, read\_buf, read\_len);

/\* 命令 ”**PASS password\r\n**” \*/

sprintf(send\_buf,"PASS %s\r\n",password);

/\* 客户端发送密码到服务器端 \*/

write(control\_sock, send\_buf, strlen(send\_buf));

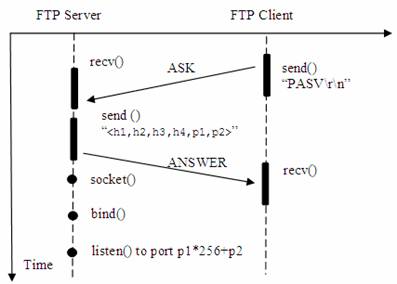
/\* 客户端接收服务器的响应码和信息，正常为 ”**230 User logged in, proceed.”** \*/

read(control\_sock, read\_buf, read\_len);

**客户端让 FTP 服务器进入被动模式**

当客户端在下载/上传文件前，要先发送命令让服务器进入被动模式。服务器会打开数据端口并监听。并返回响应码 227 和数据连接的端口号。

**图 3. 客户端让服务器进入被动模式**



**清单 3. 让服务器进入被动模式，在数据端口监听**

/\* 命令 ”**PASV\r\n**” \*/

sprintf(send\_buf,"PASV\r\n");

/\* 客户端告诉服务器用被动模式 \*/

write(control\_sock, send\_buf, strlen(send\_buf));

/\*客户端接收服务器的响应码和新开的端口号，

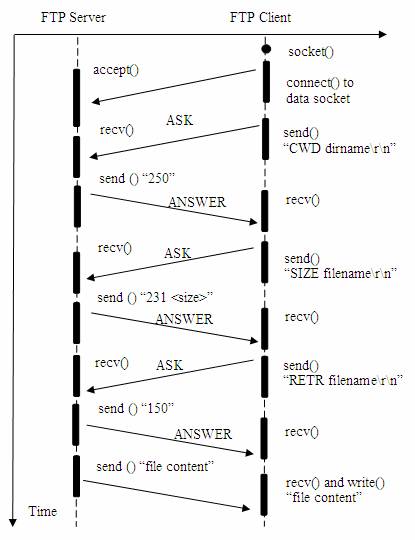
\* 正常为 **”227 Entering passive mode (<h1,h2,h3,h4,p1,p2>)”** \*/

read(control\_sock, read\_buf, read\_len);

**客户端通过被动模式下载文件**

当客户端发送命令下载文件。服务器会返回响应码 150，并向数据连接发送文件内容。

**图 4. 客户端从FTP服务器端下载文件**



**清单 4. 客户端连接到 FTP 服务器的数据端口并下载文件**

/\* 连接服务器新开的数据端口 \*/

connect(data\_sock,(struct sockaddr \*)&server, sizeof(server));

/\* 命令 ”**CWD dirname\r\n**” \*/

sprintf(send\_buf,"CWD %s\r\n", dirname);

/\* 客户端发送命令改变工作目录 \*/

write(control\_sock, send\_buf, strlen(send\_buf));

/\* 客户端接收服务器的响应码和信息，正常为 ”**250 Command okay.”** \*/

read(control\_sock, read\_buf, read\_len);

/\* 命令 ”**SIZE filename\r\n**” \*/

sprintf(send\_buf,"SIZE %s\r\n",filename);

/\* 客户端发送命令从服务器端得到下载文件的大小 \*/

write(control\_sock, send\_buf, strlen(send\_buf));

/\* 客户端接收服务器的响应码和信息，正常为 ”**213 <size>”** \*/

read(control\_sock, read\_buf, read\_len);

/\* 命令 ”**RETR filename\r\n**” \*/

sprintf(send\_buf,"RETR %s\r\n",filename);

/\* 客户端发送命令从服务器端下载文件 \*/

write(control\_sock, send\_buf, strlen(send\_buf));

/\* 客户端接收服务器的响应码和信息，正常为 ”**150 Opening data connection.”** \*/

read(control\_sock, read\_buf, read\_len);

/\* 客户端创建文件 \*/

file\_handle = open(disk\_name, CRFLAGS, RWXALL);

for( ; ; ) {

... ...

/\* 客户端通过数据连接 从服务器接收文件内容 \*/

read(data\_sock, read\_buf, read\_len);

/\* 客户端写文件 \*/

write(file\_handle, read\_buf, read\_len);

... ...

}

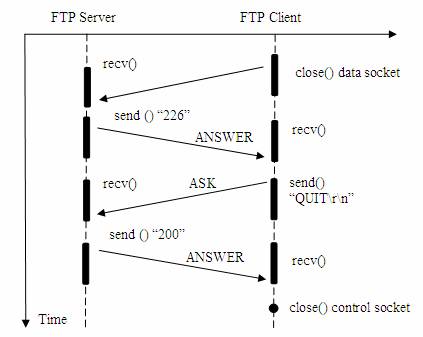
/\* 客户端关闭文件 \*/

rc = close(file\_handle);

**客户端退出服务器**

当客户端下载完毕后，发送命令退出服务器，并关闭连接。服务器会返回响应码 200。

**图 5. 客户端从 FTP 服务器退出**



**清单 5. 客户端关闭数据连接,退出 FTP 服务器并关闭控制连接**

/\* 客户端关闭数据连接 \*/

close(data\_sock);

/\* 客户端接收服务器的响应码和信息，正常为 ”226 Transfer complete.” \*/

read(control\_sock, read\_buf, read\_len);

/\* 命令 ”QUIT\r\n” \*/

sprintf(send\_buf,"QUIT\r\n");

/\* 客户端将断开与服务器端的连接 \*/

write(control\_sock, send\_buf, strlen(send\_buf));

/\* 客户端接收服务器的响应码，正常为 ”200 Closes connection.” \*/

read(control\_sock, read\_buf, read\_len);

/\* 客户端关闭控制连接 \*/

close(control\_sock);

至此，下载文件已经完成。需要注意的是发送 FTP 命令的时候，在命令后要紧跟 “**\r\n**”,否则服务器不会返回信息。回车换行符号 “\r\n” 是 FTP 命令的结尾符号，当服务器接收到这个符号时，认为客户端发送的命令已经结束，开始处理。否则会继续等待。

让我们来看一下 FTP 服务器这一端的响应情况：

**清单 6. 客户端下载文件时，FTP 服务器的响应输出**

(not logged in) (127.0.0.1)> Connected, sending welcome message...

(not logged in) (127.0.0.1)> 220-FileZilla Server version 0.9.36 beta

(not logged in) (127.0.0.1)> 220 hello gaoleyi

(not logged in) (127.0.0.1)> USER gaoleyi

(not logged in) (127.0.0.1)> 331 Password required for gaoleyi

(not logged in) (127.0.0.1)> PASS \*\*\*\*\*\*\*\*\*

gaoleyi (127.0.0.1)> 230 Logged on

gaoleyi (127.0.0.1)> PWD

gaoleyi (127.0.0.1)> 257 "/" is current directory.

gaoleyi (127.0.0.1)> SIZE file.txt

gaoleyi (127.0.0.1)> 213 4096

gaoleyi (127.0.0.1)> **PASV**

gaoleyi (127.0.0.1)> 227 Entering Passive Mode (127,0,0,1,13,67)

gaoleyi (127.0.0.1)> **RETR file.txt**

gaoleyi (127.0.0.1)> 150 Connection accepted

gaoleyi (127.0.0.1)> 226 Transfer OK

gaoleyi (127.0.0.1)> QUIT

gaoleyi (127.0.0.1)> 221 Goodbye

首先，服务器准备就绪后返回 220。客户端接收到服务器端返回的响应码后，相继发送“USER username” 和 “PASS password” 命令登录。随后，服务器返回的响应码为 230 开头，说明客户端已经登入了。这时，客户端发送 PASV 命令让服务器进入被动模式。服务器返回如 “227 Entering Passive Mode (127,0,0,1,13,67)”，客户端从中得到端口号，然后连接到服务器的数据端口。接下来，客户端发送下载命令，服务器会返回响应码 150，并从数据端口发送数据。最后，服务器返回 “226 transfer complete”，表明数据传输完成。

需要注意的是，客户端不要一次发送多条命令，例如我们要打开一个目录并且显示这个目录，我们得发送 CWD dirname，PASV，LIST。在发送完 CWD dirname 之后等待响应代码，然后再发送后面一条。当 PASV 返回之后，我们打开另一个 Socket 连接到相关端口上。然后发送 LIST，返回 125 之后在开始接收数据，最后返回 226 表明完成。

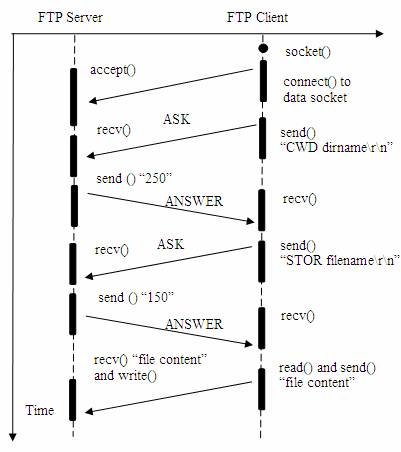
在传输多个文件的过程中，需要注意的是每次新的传输都必须重新使用 PASV 获取新的端口号，接收完数据后应该关闭该数据连接，这样服务器才会返回一个 2XX 成功的响应。然后客户端可以继续下一个文件的传输。

上传文件与下载文件相比，登入验证和切换被动模式都如出一辙，只需要改变发送到服务器端的命令，并通过数据连接发送文件内容。

**客户端通过被动模式向服务器上传文件**

当客户端发送命令上传文件，服务器会从数据连接接收文件。

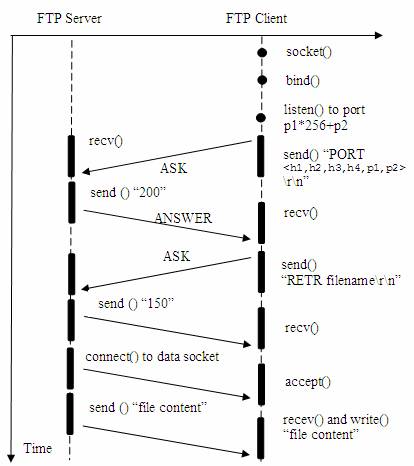
**图 6. 客户端连接到 FTP 服务器的数据端口并上传文件**



**客户端通过主动模式向服务器上传文件**

到目前为止，本文介绍的都是客户端用被动模式进行文件的上传和下载。下面将介绍客户端用主动模式下载文件。

**图 7. 用主动模式从 FTP 服务器下载文件**



**清单 7. 用主动模式从 FTP 服务器下载文件的示例 C 程序**

... ...

SOCKET data\_sock;

data\_sock = socket(AF\_INET, SOCK\_STREAM, 0);

struct sockaddr\_in name;

name.sin\_family = AF\_INET;

name.sin\_addr.s\_addr = htons(INADDR\_ANY);

server\_port = p1\*256+p2;

length = sizeof(name);

name.sin\_port = htons(server\_port);

bind(server\_sock, (struct sockaddr \*)&name, length);

struct sockaddr\_in client\_name;

length = sizeof(client\_name);

/\* 客户端开始监听端口p1\*256+p2 \*/

listen(server\_sock, 64);

/\* 命令 ”**PORT \r\n**” \*/

sprintf(send\_buf,"PORT 1287,0,0,1,%d,%d\r\n", p1, p2);

write(control\_sock, send\_buf,strlen(send\_buf));

/\* 客户端接收服务器的响应码和信息，正常为 ”**200 Port command successful**” \*/

read(control\_sock, read\_buf, read\_len);

sprintf(send\_buf,"RETR filename.txt\r\n");

write(control\_sock, send\_buf, strlen(send\_buf));

/\* 客户端接收服务器的响应码和信息，正常为 ”**150 Opening data channel for file transfer.**” \*/

read(control\_sock, read\_buf, read\_len);

/\* ftp客户端接受服务器端的连接请求 \*/

data\_sock = accept(server\_sock,(struct sockaddr \*)&client\_name, &length);

... ...

file\_handle = open(disk\_name, ROFLAGS, RWXALL);

for( ; ; ) {

... ...

read(data\_sock, read\_buf, read\_len);

write(file\_handle, read\_buf, read\_len);

... ...

}

close(file\_handle);

客户端通过 PORT 命令告诉服务器连接自己的 p1\*256+p2 端口。随后在这个端口进行监听，等待 FTP 服务器连接上来， 再通过这个数据端口来传输文件。PORT 方式在传送数据时，FTP 客户端其实就相当于一个服务器端，由 FTP 服务器主动连接自己。

**断点续传**

由于网络不稳定，在传输文件的过程中，可能会发生连接断开的情况，这时候需要客户端支持断点续传的功能，下次能够从上次终止的地方开始接着传送。需要使用命令 REST。如果在断开连接前，一个文件已经传输了 512 个字节。则断点续传开始的位置为 512，服务器会跳过传输文件的前 512 字节。

**清单 8. 从 FTP 服务器断点续传下载文件**

... ...

/\* 命令 ”**REST offset\r\n**” \*/

sprintf(send\_buf,"REST %ld\r\n", offset);

/\* 客户端发送命令指定下载文件的偏移量 \*/

write(control\_sock, send\_buf, strlen(send\_buf));

/\* 客户端接收服务器的响应码和信息，

\*正常为 ”350 Restarting at <position>. Send STORE or RETRIEVE to initiate transfer.” \*/

read(control\_sock, read\_buf, read\_len);

... ...

/\* 命令 ”**RETR filename\r\n**” \*/

sprintf(send\_buf,"RETR %s\r\n",filename);

/\* 客户端发送命令从服务器端下载文件, 并且跳过该文件的前offset字节\*/

write(control\_sock, send\_buf, strlen(send\_buf));

/\* 客户端接收服务器的响应码和信息，\*

\*正常为 ”150 Connection accepted, restarting at offset <position>” \*/

read(control\_sock, read\_buf, read\_len);

... ...

file\_handle = open(disk\_name, CRFLAGS, RWXALL);

/\* 指向文件写入的初始位置 \*/

lseek(file\_handle, offset, SEEK\_SET);

... ...

[**回页首**](http://www.ibm.com/developerworks/cn/linux/l-cn-socketftp/#ibm-pcon)

**结束语**

本文从应用实现的角度，介绍了 FTP 协议。并用详尽的例子分析了如何用主动模式和被动模式实现 FTP 客户端上传下载文件，如何进行断点续传。通过本文可以让读者对 FTP 客户端的原理有一个深入的了解。

**参考资料**

**学习**

* 了解 FTP 协议，可以参阅 FFC959 文件：[FILE TRANSFER PROTOCOL (FTP)](http://www.w3.org/Protocols/rfc959/)。
* 需要对 FTP 有更全面的了解可以查看：[FILE TRANSFER PROTOCOL](http://en.wikipedia.org/wiki/File_Transfer_Protocol)。
* 在 [developerWorks Linux 专区](http://www.ibm.com/developerworks/cn/linux/) 寻找为 Linux 开发人员（包括 [Linux 新手入门](http://www.ibm.com/developerworks/cn/linux/newto/)）准备的更多参考资料，查阅我们 [最受欢迎的文章和教程](http://www.ibm.com/developerworks/cn/linux/best2009/index.html)。
* 在 developerWorks 上查阅所有 [Linux 技巧](http://www.ibm.com/developerworks/cn/views/linux/libraryview.jsp?search_by=Linux+%E6%8A%80%E5%B7%A7) 和 [Linux 教程](http://www.ibm.com/developerworks/cn/views/linux/libraryview.jsp?type_by=%E6%95%99%E7%A8%8B)。
* 随时关注 developerWorks [技术活动](http://www.ibm.com/developerworks/cn/offers/techbriefings/)和[网络广播](http://www.ibm.com/developerworks/cn/swi/)。

**讨论**

* 欢迎加入 [developerWorks 中文社区](http://www.ibm.com/developerworks/cn/community/)。