文件传输协议（FTP）

本备忘录状态

本备忘录是文件传输协议（FTP）的正式标准。发布本备忘录不受限制。

以下新的可选指令包含在本版规范中：

CDUP （回到上一级目录）
SMNT （结构加载）
STOU （唯一保存）
RMD （删除目录）
MKD （创建目录）
PWD （打印目录）和
SYST （系统）

注意：本规范兼容以前的版本。

1.介绍

FTP 的目标是：（1）提高文件的共享性（计算机程序和/或数据），（2）鼓励间接地（通过程序）使用远程计算机，（3）保护用户因主机之间的文件存储系统导致的变化，（4）为了可靠和高效地传输，虽然用户可以在终端上直接地使用它，但是它的主要作用是供程序使用的。

本规范尝试满足大型主机、微型主机、个人工作站、和 TACs 的不同需求。例如，容易实现协议的设计。

本文假定读者已具备传输控制协议（TCP）[2]和 Telnet 协议的知识。这些文档包含在 ARPA-Internet 协议手册中。

2. 概览

在本节中，我们将讨论 FTP 的历史、术语和模型。本节定义的术语对 FTP 来说非常重要。一些术语对 FTP 模型来说很特殊，有些读者可能在需要术语时再次阅读本节。

2..1 历史

FTP 这些年有了很大的发展。附录 III 是一个按年代排序的与 FTP 有关的 RFC 文档列表。包含最初被提议的 1971 年在 M.I.T.的主机上执行发展起来的文件传输机制（RFC 114），对该文做注解和讨论的 RFC 141。

RFC 172 提供了一个用户级的在两台主机之间（包括 IMPs 终端）的定向协议。RFC 265 做了修订，重申了 FTP 的附加功能。RFC 281 提出了更进一步的改进建议。“设置数据类型”处理的应用在 1982 年的 RFC 294 中被提议。

RFC 354 废除了 RFC 264 和 265。文件传输协议此时被定义为一个用于两台 ARPANET 上的主机之间文件传输的协议，FTP 主要的功能被定义为在主机间可靠高效地传输文件并允许方便地使用远程文件存储能力。

RFC 385 进一步为错误、重点和协议增加的部分做了注释。RFC414 提供了服务器和用户在使用 FTP 工作时的一个状态报告。1973 年发布的 RFC 430 对 FTP（在其他很多的 RFC 文档中被援引）做了更进一步的介绍。最后，RFC 454 作为一个“正式的”FTP 文档出版了。

1973 年 7 月，最后版本的 FTP 发布后又有相当多的变化，但是一般的结构仍然没变。RFC 542 作为一个新的“正式”规范反映了这些变化。不过，许多基于旧规范的执行没有更新。

1974 年，RFC 607 和 614 对 FTP 继续进行注释。RFC 624 被提议改进设计和修正。1975 年，RFC 686 论述了早期的和后来的 FTP 版本之间的不同。RFC 691 对 RFC 686 关于打印文件的主题提出了一个较小的修订。

随着 NCP 到 TCP 的转变，出现了 RFC 765，它是使用 TCP 的 FTP 规范。

现在的这个版本的 FTP 规范，修正了一些较小的文档错误，改进了一些协议特征的说明，增加了一些新的可选指令。

特别是这些包含在本规范中的新的可选指令：

CDUP —— 回到上一级目录
SMNT —— 结构加载
STOU —— 唯一保存
RMD —— 删除目录
MKD —— 创建目录
PWD —— 打印目录
SYST —— 系统

本规范兼容以前的版本。

2.2 术语

ASCII

ASCII 字符集是在 ARPA-Internet 协议手册中定义的。在 FTP 里，ASCII 字符被定义为 8 位的编码集。

权限控制

权限控制定义了用户在一个系统中可使用的权限和对系统中文件操作的权限。权限控制在防止未被授权或意外地使用文件时是必需的。server-FTP 过程有调用权限控制的特权。

字节大小

FTP 中有两种类型的字节大小：文件的逻辑字节大小，和用于数据传输的传输字节大小。传输字节大小通常是 8 位。传输字节不必等于系统中存储数据的字节大小，也不必对数据结构进行解释。

控制连接

控制连接是建立在 USER-PIT 和 SERVER-PI 之间用于交换命令与应答的通信链路。该连接遵从 Telnet 协议。

数据连接

数据连接是在特定的模式和类型下，传输数据的全双工连接。传输数据可以是文件的一部分、整个文件或数个文件。链路可以建立在服务器 DTP

和用户 DTP 之间也可以建立在两个服务器 DTP 之间。

数据端口

为了建立数据连接，被动数据传输过程需要在一个端口“监听”主动传输过程的消息。

DTP

数据传输过程，建立和管理数据连接，DTP 可以是主动的也可以是被动的。

End-of-Line

End-of-Line定义了打印行时的分隔符。它是“回车符”。

EOF

end-of-file 是传输的文件的结尾标志。

EOR

end-of -record 是传输的记录的结尾标志。

错误恢复

一个允许用户在主机系统或文件传输失败时可以从特定的错误恢复的程序。在 FTP 中，错误恢复也包括在给定一个检查点时重新开始文件传输。

FTP 指令

包含从 user-FTP 到 server-FTP 的过程的控制信息的指令集。

文件

一个计算机数据的有序集合（包括程序），可以是任意长度的，由唯一的路径名来标识。

模式

数据的模式通过数据连接传输。模式定义了传输期间包含 EOR 和 EOF 的格式。

NVT

在 Telnet 协议中定义的网络虚拟终端。

NVFS

网络虚拟文件系统。是一个定义了拥有标准指令和路径名约定的标准的网络文件系统。

页

一个文件独立的部分的集合称为页。FTP 支持由独立的索引页组成的不连续文件的传送。

路径名

路径名是用户为了识别文件输入到文件系统的字符串。路径名通常包含设备和/或目录的名字，和指定的文件名。FTP 还没有指定一个标准的路径名约定。每个用户必须遵从文件系统有关文件传输的文件命名约定。

PI

协议解释器。用户和服务器各拥有明确的任务user-PI和server-PI。

记录

一个顺序文件可以由数个称为记录的连续部分组成。FTP支持记录结构，除非文件不需要文件结构。

回应

回应是对FTP指令作出的应答（肯定的或否定的），经由控制连接从服务器发送到客户端。通常回应的形式是一个完成码（包括错误码）再跟随

一个文本字符串。代码是供程序使用的，文本则通常提供给人类用户。

server-DTP

数据传输过程，一般是“主动的”状态，建立一个含有“监听”端口的数据连接。为传输和存储设置参数，从它的PI通过指令传输数据。DTP被设置为“被动的”状态收听消息，比开按就连接到数据端口的效果要好。

server-FTP过程

在和user-FTP过程，或者可能是其他服务器合作，完成文件传输过程功能的过程，由多个处理组成的集合，该功能由一个协议解释器（PI）和一个数据传输过程（DTP）组成。

server-PI

服务器协议解释器在端口L“监听”，以期来自user-PI的连接连接，建立一个控制通信连接。它从user-PI收到标准的FTP指令，然后发出回应，接着管理server-DTP。

类型

数据表示类型用来进行数据传输和存储。类型意味着在数据存储和数据传输的时间内特定的转换。

用户

一个期望获得文件传输服务的人或过程。人类用户可以直接影响server-FTP过程，但是，应首选user-FTP过程，因为它有利于协议设计朝自动控制方向发展。

user-DTP

为了来自server-FTP过程的数据连接，数据传输过程“监听”数据端口。如果两个服务器之间传输数据，user-DTP就停止了。

user-DTP过程

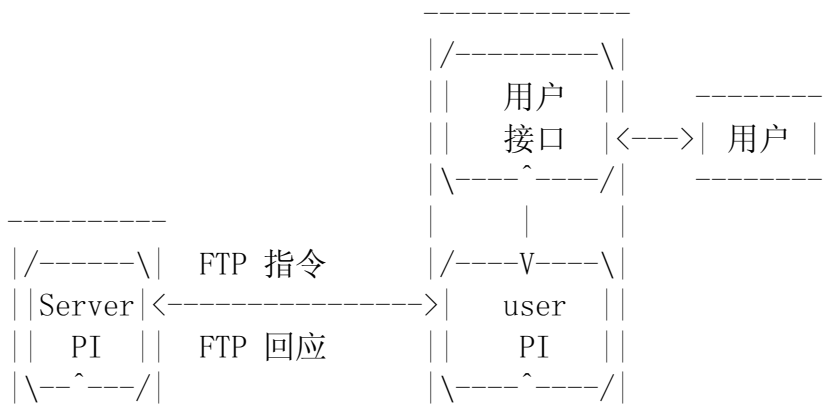
与一个或多个server-FTP过程合作，为了完成文件传输功能的功能集合。包括一个协议解释器，一个数据传输过程和一个用户界面。用户界面允许使用本地语言显示指令回应的对话。

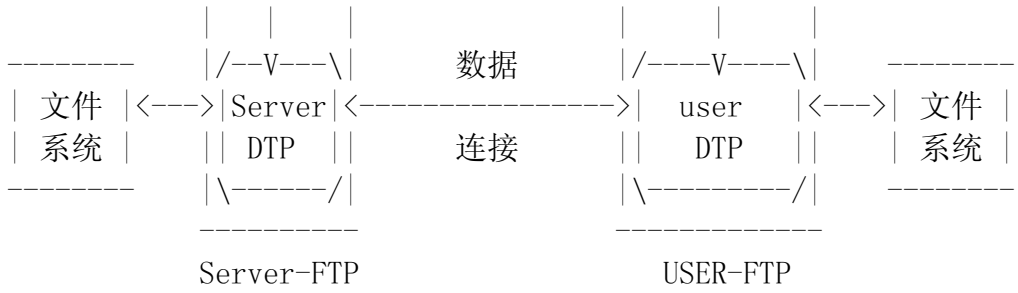
user-PI

用户协议解释器开始控制连接从它的U端口到server-FTP过程，如果这个过程是文件传输的一部分，接着初始化FTP指令，然后管理user-FTP。

2.3FTP模型

根据上面的定义，下面的模型（见图1）是FTP服务示意图：





注意：1.数据连接是双向的。

2.数据连接无需在整个时间存在。

图1　FTP用户模型

图1描述的模型中，user-PI开始控制连接。控制连接遵从Telnet协议。在用户初始化时，user-PI产生标准的FTP指令，并经由控制连接传送到服务器过程。（用户可以建立一个直接的控制连接到server-FTP，比如从TAC终端，并且发出独立的标准FTP指令，这样就绕过了user-FTP过程。）标准的回应通过控制连接从server-PI发送到user-PI。

FTP指令可以为数据连接和文件系统操作的状态（存储，下载，设置数据文件的搜索路径，删除等）指定参数（数据端口，传输模式，表现类型和结构）。

user-DTP应该在指定的数据端口“监听”，服务器初始数据连接，数据用特定的参数保证一致地传输。注意，数据端口不必和初始化FTP指令的主机是同一台主机，但是用户或user-FTP过程必须保证在指定的数据端口“监听”。还要注意，数据连接可能同时也在发送和接收。

另一种情况是，用户希望在两台主机间传送文件，没有一台是本地主机。用户在两台主机间建立控制连接，并准备在它们之间进行数据连接。在这种方式下，控制信息经过user-PI，但数据在服务器数据传输过程间传输。下图是这种服务器-服务器交互方式的模型：

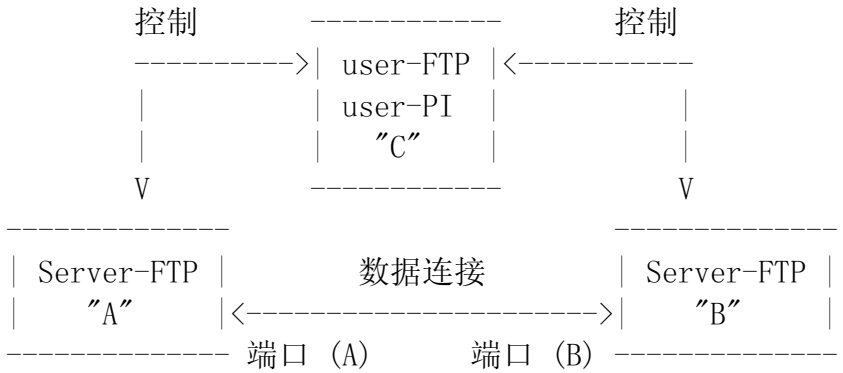


图2

协议要求数据传输在处理时打开控制连接。在完成FTP服务后，用户的责任是请求关闭控制连接，而服务器具体操作。如果在未接收命令时关

闭了控制连接，服务器也会关闭数据传输。

FTP和Telnet的关系：

FTP在控制连接中使用Telnet协议。可以通过两种方式达到目的：第一种，user-PI或server-PI按Telnet的规则直接在它们的过程中实现；第二种，user-PI或server-PI可以利用系统中已存在的模块实现。

实现很容易，共享代码，第二种方式有利于程序模块化，第一种方式有利于传输效率和独立性。实际上，FTP对Telnet的依赖非常小，用第二种方式实现的代码量也不大。

3．数据传输功能

文件通过数据连接传输。控制连接用来传输指令，它描述了执行的功能和指令的回应(参看 FTP 回应一节)。有几个指令和主机间数据传输有关。这些指令包括：传输时指定数据位数的 MODE 指令，用来定义数据表现的 STRU 指令和 TYPE 指令。数据传输和表现基本上是独立的，但是，“流”模式依赖于文件结构属性，如果使用“压缩”传输模式，填充字节的状态依赖于表示类型。

3.1数据表示和存储

数据从发送主机的一个存储设备传输到接收主机的一个存储设备。因为通常两个系统的数据表示是不同的，所以需要对数据进行特定的转换。比如NVT-ASCII在不同的系统中有不同的数据表示。例如，DEC TOPS-20中一般以5个7位的ASCII字符存储NVT-ASCII，IBM的大型机使用8位的EBCDIC码存储NVT-ASCII，而Multics用4个9位字符存储Multics。在不同的系统间传输文本时，要想令人满意，就要把这些字符都转换成标准的NVT-ASCII表示。发送和接收的站点完成在标准表示和内部表示之间的必要的转换。

另一个问题是，在不同的字长的主机间传送二进制数据（不是字符编码）时如何表示。通常不清楚发送方如何发送数据，也不清楚接收方如何存储。例如，当从32位字长的系统传送32位的字节到36位字长的系统时，需要（为了高效和易用）将32位的字节存储成36位的字。有时，用户要有可选的特定数据表示与传输功能。注意，FTP提供非常有限的数据类型表示。超过FTP提供功能的那一部分要用户自己实现。

3.1.1数据类型

数据表示是由用户指定的表示类型，类型可以隐含地（比如ASCII或EBCDIC）或明确地（比如本地字节）定义一个字节的长度，提供像“逻辑字节长度”这样的表示。注意，在数据连接上传输时使用的字节长度称为“传输字节长度”，和上面说的“逻辑字节长度”不要弄混。例如，NVT-ASCII的逻辑字节长度是8位。如果该类型是本地类型，那么TYPE指令必须在第二个参数中指定逻辑字节长度。传输字节长度通常是8位的。

3.1.1.1ASCII类型

这是所有FTP执行必须承认的默认类型。它主要用于传输文本文件，除非两台主机用EBCDIC类型更方便。

发送方把内部字符表示的数据转换成标准的8位NVT-ASCII表示(参看

Telnet规范）。接收方把数据从标准的格式转换成自己内部的表示形式。

与NVT标准保持一致，要在行结束处使用<CRLF>序列。（参看数据表示和存储一节末尾有关文件结构的讨论。）

使用标准的NVT-ASCII表示的意思是数据必须转换为8位的字节。

ASCII和EBCDIC的格式参数在下面讨论。

3.1.1.2 EBCDIC类型

这种类型在使用EBCDIC作为内部字符的主机间能提供高效的传输。

为了传输，数据被表示为8位的EBCDIC字符。仅在类型的功能描述上有一些差别。

行结束符（对应的记录结束符请参看结构的讨论）在EBCDIC类型中很少用来做结构表示，但是它需要使用<NL>字符。

3.1.1.3 IMAGE类型

数据以连续的位传输，并打包成8位的传输字节。接收站点必须以连续的位存储数据。存储系统的文件结构（或者对于记录结构文件的每个记录）必须填充适当的分隔符，分隔符必须全部为零，填充在文件末尾（或每个记录的末尾），而且必须有识别出填充位的办法，以便接收方把它们分离出去。填充的传输方法应该充分地宣传，使得用户可以在存储站点处理文件。

IMAGE 格式用于有效地传送和存储文件和传送二进制数据。推荐所有的FTP在执行时支持此类型。

3.1.1.4 本地类型

以逻辑字节传输的数据必须在第二个参数中指定字节的大小。字节大小的值必须是十进制整数，它没有缺省值。逻辑字节大小不必和传输字节大小一样。如果使用不同的字节大小，那么逻辑字节应使用连续的方式打包，忽略传输字节的分隔符，并且无须在末尾进行任何填充。

当数据到达接收方主机，它的转换方式依赖于逻辑字节大小和主机的特性。该转换必须是可逆的（就是说，使用同样的参数可以重新得到这个文件），并且应对FTP的执行者进行充分的宣传。

例如，用户发送36位浮点数到一个用32位的字表示的主机，数据可以用逻辑字节大小为36的逻辑字节发送。接收方主机为了存储这个逻辑字节，需要做简单的操作：在本例中，把36位的逻辑字节转换成64位的双字就足够了。

另一个例子，2台用36位的字表示的主机，可以使用36位的逻辑字发送数据到对方。被发送的数据以8位的字节打包，所以9个传输字节传送了2个主机字。

3.1.1.5 格式控制

ASCII和EBCDIC类型也使用了第二个（可选的）参数：它用于指出纵向格式控制的类型，或者是任何与文件关联的类型。下面的数据表示类型已在FTP中定义：

一个字符文件是为了下列三种目的之一传送到远程主机的：为了打印，为了存储和以后信息的检索，或为了处理。如果一个文件为了打印而发送，

接收方主机必须知道纵向格式控制是如何表示的。为了第二种目的，必须可以在主机存储文件并且以后检索时要格式正确。最后一种目的，应该可以移动这个文件到其他主机并且可以在第二台主机上处理而没有以外的麻烦。单独的ASCII或EBCDIC格式不能满足所有这些条件。所以，这些类型需要第二个参数指定下列三种格式之一：

3.1.1.5.1 非打印

如果省略掉第二个（格式）参数的话，这是缺省的格式。非打印格式必须被所有的FTP执行者承认。

文件不需要包含纵向格式信息。如果它经过一个打印过程，那么该过程将以假定的标准间隔和边距值来处理。

通常，该格式用来处理文件或仅仅用来存储。

3.1.1.5.2 TELNET格式控制

文件包含打印过程可以正确解释的ASCII/EBCDIC纵向格式控制（比如，<CR>，<LF>，<NL>，<VT>，<FF>）。<CRLF>，在这个序列里也表示行结束符。

3.1.1.5.2 托架(走纸)控制（ASA）

文件包含ASA (FORTRAN)的纵向控制字符。（参看RFC 740 附录 C，和ACM通信，606页，Vol. 7, No. 10,1964年10月。）在有格式的行或记录中，遵照ASA的标准，第一个字符不能打印。它用来限定被打印的记录静止时的走纸量。

ASA标准指定了下列字符：

字符	垂直间距
空格	纸张上移一行
0	纸张上移二行
1	纸张移到下页顶端
+	不移动，比如套印

很明显，打印过程必须有识别结构实体末尾的方式。如果文件采用了记录结构（见下文）不会有问题，记录可以在传输和存储期间明确地标出。如果文件不是采用记录结构，<CRLF>行结束符用来分隔打印行，但是这些格式控制符超出了ASA的控制。

3.1.2 数据结构

除了不同的数据表示类型，FTP还允许指定文件的结构。FTP中定义了三种文件结构：

文件结构，它没有内部结构，文件由连续的数据字节组成。
记录结构，文件由连续的记录组成。
页结构，文件由独立的索引页组成。

如果没有使用STRUcture指令，文件结构就被假定是缺省的。文件和记录结构必须被所有的FTP实现以“文本”文件（比如，ASCII或EBCDIC类型的文件）的方式承认。文件的格式影响它的传输模式（参看传输模式一节）和它的解释和存储。

文件的“自然的”格式依赖于主机的存储。一个源代码文件通常在IBM的大型机中是以固定长度的记录存储的，而在DEC TOPS-20机上它是以分隔（比如用<CRLF>来分隔）成行的字符流存储的。如果想让文件在完全不同的站点传输，就必须有办法让一个站点承认另一个站点关于该文件的结构表示。

有些站点是面向文件的，还有些站点是面向记录的，如果从一个主机到另一个文件结构不同的主机传送文件时就会出问题。如果一个文本文件以记录结构传送，而接收方的主机却是面向文件的，那么这个主机就必须对该文件进行基于记录结构的转换。显然，这样的转换很有效，但是这个转换的过程必须也是可逆的。

在一个文件以文件结构传送到面向记录的主机时，存在一个问题，该主机要用什么标准把文件划分成记录以便于本地处理。如果需要这样的划分，那么FTP在执行时使用的行结束符，对ASCII文件用<CRLF>，对EBCDIC文件用<NL>作为分隔符。如果FTP的实现采用了这个技术，它就必须准备好在这个文件要重新转成文件结构时，能进行逆转换操作。

3.1.2.1 文件结构

如果没有使用STRUcture指令，文件结构就被假定是缺省的。

文件结构没有内部的结构，被认为是数据字节的连续序列。

3.1.2.2 记录结构

记录结构必须被所有的FTP实现以“文本”文件（比如，ASCII或EBCDIC类型的文件）的方式承认。

记录结构的文件由连续的记录组成。

3.1.2.3 页结构

为了传送不连续的文件，FTP定义了一种页结构。这样的文件结构已知的往往是“随机存取文件”或“holey文件”。这种结构有时候通过其他信息把文件联系成一个整体（比如，一个文件描述符），或者是文件的一部分（比如，页面存取控制）。在FTP里，这些文件的部分成为页。

为了提供不同的页大小和关联信息，每个页在传送时附加一个页头。页头定义了下列域：

页头长度：包含这个字节在内的页头的逻辑字节数。最小的页头长是4。

页索引：该部分在文件中的逻辑页编号。这不是页的传输序列号，该索引用来指出文件的这一页。

数据长度：页中数据的逻辑字节数。最小的数据长度是0。

页类型：页的类型。定义了如下类型：

0 = 最后页

用来指出页结构文件传输的末尾。页头长度必须是4，数据长度必须是0。

1=简单页

这是的简单分页文件的一般类型，它没有与控制有关的页等级信息。页头长度必须是4。

2=描述符页

该类型用来传输文件总体的描述信息。

3=访问控制页

该类型包含了一个附加的页头域，用来传送页分级访问控制信息。页头长度必须是5。

可选的域：为提供每个页的控制信息，可以使用有更多的页头域。比如，每页访问控制。

所有的域都有一个逻辑字节长度。逻辑字节大小由TYPE指令定义。参看附录一可获取更详细的信息，和页结构的一个特殊情况。

关于参数的警告：如果检索的版本和最初传送的版本一样，文件就必须用同样的参数存储和检索。相反地，如果使用了相同的参数来存储和检索数据，FTP的执行必须返回同一个文件。

3.2 建立数据连接

传输数据的细节包括在适当的端口建立数据连接和选择传输参数。用户和server-DTP有缺省的数据端口。用户进程缺省的数据端口与控制连接端口相同（比如，U）。服务器进程缺省的数据端口与控制连接端口相临（比如，L-1）。

传输字节是8位的字节。字节的大小仅与实际的数据传输有关，它和主机文件系统内的数据表示无关。

被动数据传输进程（可能是一个user-DTP，或一个第二server-DTP）在发送数据请求指令前应先在数据端口“监听”。FTP请求指令决定数据传输的方向。服务器在接收到请求以后，将初始化该端口的数据连接。当连接建立后，数据传输在DTP之间传送，server-PI对user-PI返回确认应答。

每个FTP的实现必须支持缺省数据端口的使用，并且只有user-PI可以初始化改变到一个非缺省的数据端口。

用户可能使用PORT指令指定轮流的数据端口。用户可能要求一个文件被转移到TAC行式打印机或从第三方主机接收。在后一种情况下，user-PI在两个server-PI间建立控制连接。一个服务器被命令（通过使用FTP指令）去“监听”另一个服务器将要初始化的连接。user-PI给一个server-PI发送一条包含另一台服务器的数据端口的PORT指令。最后双方发送相应的传送命令。用户控制器和服务器之间发送的指令和回应的严格顺序在FTP回应一节中定义。

通常，服务器负责维护数据连接——初始化和关闭它。例外的情况是，user-DTP在传输模式下要求关闭连接。在下列情况下服务器必须关闭数据连接：

1. 服务器在传输模式下完成了数据的发送，要求通过EOF关闭。
2. 服务器收到用户发来的ABORT指令。
3. 用户使用一个指令改变了端口。

4. 控制连接合法关闭或因其他原因关闭。

5. 发生了无可挽救的错误。

否则服务器有选择关闭的权利，此时服务器必须对用户进程用250或226回应指出。

3.3 数据连接管理

缺省的数据连接端口：所有FTP实现必须支持缺省的数据连接端口，只有user-PI能够初始化非缺省的端口。

确定非默认数据端口：user-PI可以使用PORT指令指定一个非缺省用户端数据端口。user-PI可以使用PASV指令要求服务器端指定一个非缺省服务器端口。因为连接是由双方的地址定义的，任一方的改变都会导致不同的数据连接，在结束数据连接后允许双方使用新的端口发送指令。

数据连接的重用：在使用流式数据传输模型时，文件结束通过关闭连接指示。如果要传送多个文件，由于FTP需要暂时地控制连接记录以保证可靠的传输，此时就会出麻烦。所以连接不能立刻重新开始。

解决的方法有两个，一个是确定非默认端口，另一个是使用另一种传输模式。就传输模式而言，因其固有的特点，流传输模式是不可靠的，因此无法确定连接是暂时还是永久关闭。其它传输模式（块模式，压缩模式）不通过关闭连接指出文件末尾。它们有足够的FTP编码分析数据连接以确定文件的末尾。因此使用这些传输模式可以在保持连接的情况下传送多个文件。

3.4 传输模式

下面考虑传输数据可选择的传输模式。有三种传输模式：一个是格式化数据并允许重新开始程序；一个是为有效的传输而压缩数据；一个是对数据进行少量的处理或不进行处理。在最后一种模式下，结构属性和处理类型相互影响。在压缩模式下，表示类型决定填充的字节。

所有数据传输必须以一个文件结束符EOF结束，它可以显式给出，也可以通过关闭连接隐式给出。对于记录文件，所有记录结束符标记（EOR）是显式的，包括最后一个记录。对于以页结构传送的文件，使用“最后页”表示结束。

注意：本节从这里开始，除明确指出外，字节的意思是“传输字节”。

为了传输的标准化，传送主机必须把行结束或记录结束的内部表示转化为传输模式和文件结构指定的形式传送，接收方则进行相反的工作。IBM大型机的记录计数域可能不能为其它主机识别，所以记录结束标记在流模式下以双字节控制码传送，在块或压缩模式下以标记位传送。没有记录结构的ASCII或EBCDIC的文件，行结束符则用<CRLF>或<NL>分别指示。因为这样的转换对有些系统意味着额外的工作，所以相同的系统在传送非记录结构的文本文件时采用二进制或流表示比较合适。

FTP定义了下列传输模式：

3.4.1 流模式

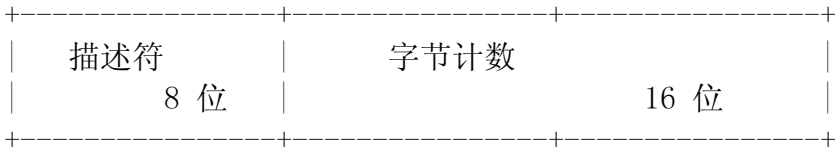
数据以字节流的方式传送。使用的表示类型没有限制，允许记录结构。在记录结构文件中 EOR 和 EOF 表示为双字节控制码。控制码的第一个字

节全是 0。第二个字节的值为 1 时表示 EOR，为 2 时表示 EOF，如果要同时表示 EOR 和 EOF，值为 3。全 1 字节作为数据发送时必须使用双字节传送，其中数据保存在第二个字节内。如果是文件结构，通过发送方关闭连接表示 EOF，接收到的所有数据就是文件内容。

3.4.2 块模式

文件以一系列数据块的方式传送，有一个或多个头字节。块头字节包括一个计数域和描述符代码。计数域指出了数据块的总长度字节数，以此标记出下一个数据块的开始（不需要填充位）。描述符代码定义了：文件的最后块（EOF），记录的最后块（EOR），重新开始标记（参看错误恢复和重新开始一节）或可疑数据（比如，数据开始传输被怀疑是错误的或不可靠的）。最后的代码不是为了在FTP中进行差错控制。它是为了站点间交换特定类型的数据（比如，地震或天气数据），并不管本地错误（像“磁带读错误”这样的错误）接收所有数据。而只管传送，但是传送时要指出这部分是可疑的。该模式允许使用记录结构，页可以使用任何表示类型。

块头包含3个字节。24位头信息中，低16位是字节计数，高8位表示描述符代码，如下所示：



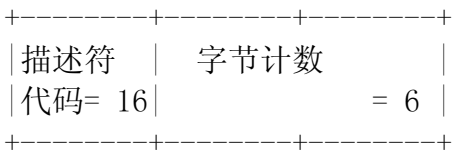
描述符代码由在描述符字节中的位标记说明。每个代码的十进制值是字节中相应的位：

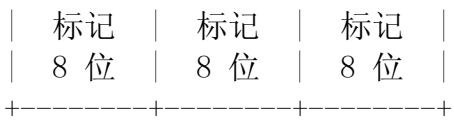
代码	含义
128	数据块结束是 EOR
64	数据块结束是 EOF
32	数据块内有可疑错误
16	数据块是重新开始标记

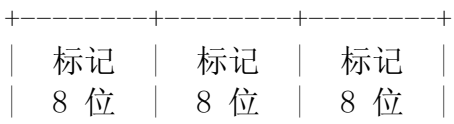
以这种编码，对于特定块可能存在多个描述子编码条件，所需要的位必须全部设置。重新开始标记包括在数据流中，它作为8位整数代表在控制连接上使用语言的可打印字节，但<SP>不得出现在其中。例如要传送6字节标记，以这种编码，对于特定块可能存在多个描述符代码的情况。所需要的位必须全部被标记。

重新开始标记包括在数据流中，它作为8位字节整数代表在控制连接上使用语言的可打印字符（比如，缺省--NVT-ASCII）。<SP>（空格，在合适的语言中）不可以使用在重新开始标记里。

例如，要传输一个6字符标记，如下：



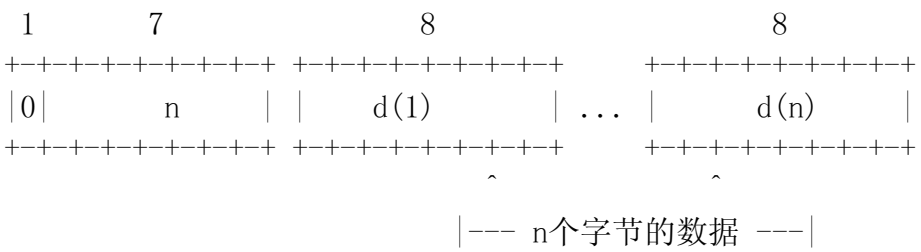




3.4.3 压缩模式

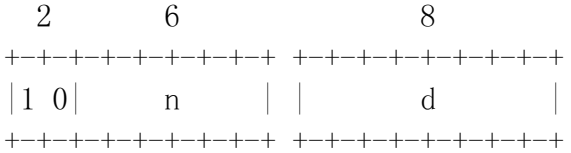
有三种类型的信息可以被传送：常规数据，用字节串方式传送；压缩数据，由重复和填充组成；控制信息，用双字节的escape序列发送。如果常规发送的数据n>0字节（小于等于127），这些n字节前加一个字节，最左位是0，右7位是数字n。

字节串：



d(1),..., d(n)表示n个字节的数据串，计数n必须准确。
要压缩一个数据位d的N个重复字符串，用下面的两个字节传送：

重复的字节:



一串n个填充字节可以压缩为一个字节，而填充字节根据表示类型的不同而不同。如果类型是ASCII或EBCDIC，填充字节是<SP>（空格，ASCII代码32，EBCDIC代码64），如果是IMAGE或本地类型，填充字节是是一个0字节。

填充字符串：

```
+-+-+-+-+-+-+-+-+
|1 1|       n       |
+-+-+-+-+-+-+-+-+
```

escape序列是一个双字节，第一个字节是escape字节（全0），第二个字节包含了块模式定义的描述符代码。描述符代码的含义和块模式中的一样，它作用于其后串中的字节。

压缩模式对增加带宽有用，一个非常大的网络传输只需很小的CPU开销。它也可以最有效地减少像RJE主机那样产生的打印文件的大小。

3.5 错误恢复和重新开始

数据传输没有提供对位丢失或和混乱的检测，这个差错控制级由TCP实现。不过，重新开始程序提供了保护用户系统失败（包括主机失败，FTP进程失败，或潜在的网络失败）的方法。

重新开始程序仅对块和压缩的数据传输模式有效。它要求数据的发送方在数据流中插入一个特殊的标记码来提供标记信息。标记仅对发送方有意义，但必须包含缺省的或控制连接的协商语言的可打印字符（ASCII 或EBCDIC）。标记可以表示位数，记录数，或任何系统识别一个数据检查点的其他信息。数据的接收方如果执行了重新开始程序，就会用接收方系统中的标记码标出相应的位置，然后把这个信息返回给用户。

发生系统失败事件后，用户可以通过FTP重新开始程序，指定标记点来重新开始数据传输。下面是重新开始程序使用的例子：

数据的发送方在数据流中合适的点插入一个适当的标记块。接收方主机在它的文件系统中标记相应的数据点，通过直接地或控制连接的110回应（依赖于用户的身份）把最后知道的标记信息告诉发送方。发生系统失败事件后，用户或控制进程使用一个重新开始指令让服务器在最后一个标记的位置重新开始，指令中的一个参数指出了最后一次服务器的标记。重新开始指令通过控制连接传送，并紧跟在系统发生失败时正在执行的指令（比如RETR，STOR或LIST）之后。

4. 文件传输功能

从user-PI到server-PI的通信信道建立在从用户到标准的服务器端口的TCP连接上。用户协议解释器负责发送FTP指令和解释收到的回应；server-PI解释指令，发送应答，并指导它的DTP建立数据连接和传送数据。如果数据传输（被动传输进程）的第二方是user-DTP，通过user-FTP主机的内部协议对它进行控制；如果第二方是server-DTP，由user-PI发来的指令经过它自己的PI进行控制。如果清楚相关的可能回应，对理解本节描述的少数指令会有帮助。

4.1 FTP指令

4.1.1 访问控制指令

下列指令指定访问控制标识符（指令码在括号内给出）。

用户名（USER）

它的参数是用来指定用户的Telnet字串。它用来进行用户鉴定，服务器对赋予文件的系统访问权限。该指令通常是建立数据连接后（有些服务器需要）用户发出的第一个指令。有些服务器还需要通过password或account指令获取额外的鉴定信息。服务器允许用户为了改变访问控制和/或帐户信息而发送新的USER指令。这会导致已经提供的用户，口令，帐户信息被清空，重新开始登录。所有的传输参数均不改变，任何正在执行的传输进程在旧的访问控制参数下完成。

口令（PASS）

它的参数是用来指定用户口令的Telnet字符串。

此指令紧跟用户名指令，在某些站点它是完成访问控制不可缺少的一步。因为口令信息非常敏感，所以它的表示通常是被“掩盖”起来或什么也不显示。服务器没有十分安全的方法达到这样的显示效果，因此，user-FTP进程有责任去隐藏敏感的口令信息。

帐户（ACCT）

它的参数是用来指定用户帐户的Telnet字串。此指令不需要与USER指令相关，一些站点可能需要帐户用于登录，另一些仅是为了特殊的访问，比如存储文件。在后一种情况下，此命令可在任何时候发送。

用下面的回应码来区分这些情况以实现自动控制：如果登录需要帐户信息，对成功的PASSword指令的回应码是332。另一方面，如果帐户信息不是登录时必须的，对成功的PASSword指令的回应码是230。如果帐户信息在以后的某指令发出后需要，服务器应返回332或532回应码，这要依赖于它是存储（尚不能确定ACCounT指令的接收）或放弃该指令。

改变工作路径（CWD）

该指令允许用户在不改变它的登录和帐户信息的状态下，为存储或下载文件而改变工作目录或数据集。传输参数不会改变。它的参数是指定目录的路径名或其他系统的文件集标志符。

回到父目录（CDUP）

该指令是CWD的一种特殊情况，它用来简化传输目录树的程序，因为各操作系统对父目录命名使用不同的语法。回应码与CWD的回应码相同。参看附录二获得更详细的信息。

结构加载（SMNT）

该指令允许用户在不改变登录和帐户信息的状态下加载不同的文件系统数据结构。传输参数保持不变。它的参数是指定目录的路径名或其他系统的文件集标志符。

重新初始化（REIN）

该指令终止一个用户，清除所有I/O和帐户信息，允许任何正在运行的

传输完成。所有参数复位到缺省的设置，控制连接关闭。其后可以跟随USER指令。

退出登录（QUIT）

该指令终止一个用户，如果没有正在执行的文件传输，服务器将关闭控制连接。如果有数据传输，在得到传输响应后服务器关闭控制连接。如果用户进程正在向不同的用户传输数据，不希望对每个用户关闭然后再打开，可以使用REIN指令代替QUIT。

对控制连接的意外关闭，可以导致服务器运行中止(ABOR)和退出登录(QUIT)。

4.1.2 传输参数指令

所有的数据传输参数都有缺省值，仅当要改变缺省的参数值时才使用此指令指定数据传输的参数。缺省值是最后一次指定的值，如果没有指定任何值，那么就使用标准的缺省值。这意味着服务器必须“记住”合适的缺省值。在FTP服务请求之后，指令的次序可以任意。下列指令指定数据传输参数：

数据端口（PORT）

该指令的参数是用来进行数据连接的数据端口。客户端和服务器均有缺省的数据端口，并且一般情况下，此指令和它的回应不是必需的。如果使用该指令，则参数由32位的internet主机地址和16位的TCP端口地址串联组成。地址信息被分隔成8位一组，各组的值以十进制数（用字符串表示）来传输。各组之间用逗号分隔。一个端口指令：

```
PORT h1,h2,h3,h4,p1,p2
```

这里h1是internet主机地址的高8位。

被动（PASV）

该指令要求server-DTP在一个数据端口（不是缺省的数据端口）“监听”以等待连接，而不是在接收到一个传输指令后就初始化。该指令的回应包含服务器正监听的的主机地址和端口地址。

表示类型（TYPE）

该指令的参数指出了“数据表示和存储”一节中描述的数据表示。有些类型还有第二个参数。第一个参数由单独的Telnet字串指出，第二个参数的格式是ASCII和EBCDIC。第二个参数用十进制整数指出字节大小。参数间用<SP>（空格，ASCII码32）分隔。

为类型分配了下列代码：

```
              \    /
A - ASCII |       | N - 非打印
```

```
          |->X<-| T - Telnet格式控制符
E - EBCDIC|     | C - 走纸控制 (ASA)
          /     \
I - Image
L <字节大小> - 本地字节大小
```

缺省的表示类型是ASCII非打印类型。如果格式参数改变了，并且以后只改变了第一个参数，那么格式将返回非打印的缺省值。

文件结构（STRU）

该指令的参数用Telnet字串指出了“数据表示和存储”一节中描述的文件结构。

下面的代码分配给了不同的结构：

F - 文件（非记录结构）
R - 记录结构
P - 页结构

缺省的结构是文件。

传输模式（MODE）

该指令的参数用Telnet字串指出了“传输模式”一节中描述的数据传输模式。

下面的代码分配给了不同的传输模式：

S - 流模式
B - 块模式
C - 压缩模式

缺省值是流模式。

4.1.3 FTP服务指令

FTP服务指令表示用户要求的文件传输或文件系统功能。FTP服务指令的参数通常是一个路径名。路径名的语法必须符合服务器站点的规定和控制连接的语言规定。隐含的缺省值是使用最后一次指定的设备，目录或文件名，或本地用户定义的标准缺省值。指令顺序通常没有限制，只有“rename from”指令后面必须是“rename to”，重新启动指令后面必须是中断服务指令（比如，STOR或RETR）。除确定的报告回应外，FTP服务指令的响应总是在数据连接上传输。下列指令指定FTP服务请求：

获得文件（RETR）

该指令让 server-DTP 用指定的路径名传送一个文件的复本到数据连接另一端的 server-DTP 或 user-DTP。该服务器站点上文件状态和内容不受影响。

保存（STOR）

该指令让server-DTP通过数据连接接收数据传输，并且把数据存储为服务器站点的一个文件。如果指定的路径名的文件在服务器站点已存在，那么它的内容将被传输的数据替换。如果指定的路径名的文件不存在，那么将在服务器站点新建一个文件。

唯一保存（STOU）

此命令和STOR类似，它要求被建立的文件在当前目录下的文件名是唯一的。250回应必须包含产生的文件名。

追加（和新建）（APPE）

该指令让server-DTP通过数据连接接收数据传输，并且把数据存储为服务器站点的一个文件。如果指定的路径名的文件在服务器站点已存在，那么数据将会追加到该文件之后。如果指定的路径名的文件不存在，那么将在服务器站点新建一个文件。

分配（ALLO）

此指令用于在一些主机上为新传送的文件提供足够的存储空间。参数是表示用来存储文件所需字节数的十进制数（使用逻辑字节大小）。如果是记录或页结构，页或记录的最大大小（使用逻辑字节大小）也需要，这在指令的第二个参数内以十进制指定。第二个参数是可选的，如果它存在，它和第一个参数以三个Telnet字符<SP> R <SP>分隔。此命令在STOR或APPE命令后。对于不需要分配存储空间的机器，它的用途等于NOOP。这些服务器只对最大的记录或页大小感兴趣，它从第一个参数中接收一个哑元值，并忽略它。

重新开始（REST）

该指令的参数代表服务器要重新开始的文件传输的个标记。此命令并不传送文件，而是跳到文件的指定数据检查点。此命令后应该紧跟合适的使数据重传的FTP服务指令。

重命名（RNFR）

该指令指定了要被改名的文件的旧路径名。此指令必须紧跟一个“重命名为”指令来指明新的文件路径名。

重命名为（RNTO）

此指令指出了上面的“重命名”指令提到的文件的新路径名。这两个指令共同完成文件的重命名。

放弃（ABOR）

此指令告诉服务器放弃此前的FTP服务指令，和任何相关的数据传输。此指令可能需要“特殊的动作”（在“FTP指令”一节讨论），来增强服务器的识别。如果此前的指令都已完成（包括数据传输），那么不需要其他动作。

服务器不会关闭控制连接，但是会关闭数据连接。

服务器在收到这个指令时会有两种情况：（1）FTP服务指令已经完成。（2）FTP服务指令仍在执行。

第一种情况下，服务器关闭数据连接（如果已建立），然后返回226回应，指出这个放弃指令已经成功执行。

第二种情况下，服务器放弃正在执行的FTP服务，关闭数据连接，返回426回应，指出服务请求被异常终止。然后服务器发送226回应，指出这个放弃指令已经成功执行。

删除（DELE）

此指令从服务器站点删除指定路径名的文件。如果需要额外级别的保护（比如，询问“你真的要删除此文件吗？”），它需要由user-FTP进程提供。

删除目录（RMD）

此指令删除指定路径名的目录（如果是绝对路径）或删除当前工作目录的子目录（如果是相对路径）。参看附录二。

创建目录（MKD）

此指令创建指定路径名的目录（如果是绝对路径）或在当前工作目录创建子目录（如果是相对路径）。参看附录二。

打印工作目录（PWD）

此指令在回应中返回当前工作目录名。参看附录二。

列表（LIST）

此指令让服务器发送列表到被动DTP。如果路径名指定了一个路径或其他的文件集，服务器会传送指定目录的文件列表。如果路径名指定了一个文件，服务器将传送文件的当前信息。不使用参数意味着使用用户当前的工作目录或缺省目录。数据传输在数据连接上进行，使用ASCII类型或EBCDIC类型。（用户必须保证表示类型是ASCII或EBCDIC）。因为一个文件的信息从一个系统到另一个系统差别很大，所以此信息很难被程序自动识别，但对人类用户却很有用。

名字列表（NLST）

此指令把目录的列表从服务器传送到用户站点。路径名应指出一个目录或其他特殊系统的文件集描述符。没有参数意味着是当前目录。服务器将返回一个文件名的信息流，没有其他信息。数据以ASCII或EBCDIC类型通过数据连接传输，有效的路径名用<CRLF> 或 <NL>分隔。（用户必须保证类型是正确的。）该指令用来为程序进一步的自动处理提供信息。

站点参数（SITE）

该指令给出服务器可提供的基本文件传输服务，而不是协议的全部指令。这些服务的特性和它们的语法规范可以从 HELP SITE 指令的回应中查看。

系统（SYST）

该指令用来查找服务器的操作系统类型。指令的回应是该系统当前版本名的第一个单词，它遵从编号的分配文档[4]。

状态（STAT）

此指令以回应的形式从控制连接传送一个状态回应。指令可能在文件传输期间被发送（与Telnet IP和同步信号一起——参看“FTP指令”一节），这时服务器将回应正在执行操作的状态。指令也可以在文件传输间隔时被发送，这时指令带一个参数域，如果参数是路径名，除了数据在控制连接传输以外，指令和“列表”指令类似。如果是一个局部路径名，服务器以文件名列表或相关的属性回应。如果没有参数，服务器会返回关于服务器FTP进程的一般状态信息，它包括所有的传输参数和连接状态。

帮助（HELP）

该指令让服务器通过控制连接给用户发送关于它执行状态的帮助信息。该指令可以带一个参数（比如，任何指令名），以回应的方式返回更多的细节信息。它的回应是211或214。建议允许在输入USER指令前允许使用HELP指令。服务器可以使用此回应给出详细的站点参数，比如，HELP SITE。

等待（NOOP）

此指令不影响任何参数或此前输入的指令。除了服务器发送OK回应以外不做任何操作。

文件传输协议在控制连接下的所有通信遵从Telnet协议规范。因为Telnet通信使用的语言可以是商谈的可选项，所有下面两节中的参考内容都与“Telnet语言”和相应的“Telnet行结束码”相关。没有其他Telnet协议的细节被引用。

FTP指令由“Telnet行结束码”的“Telnet字串”终止。本节描述了指令的指令码和语义，详细的指令语法在“指令”一节中说明，回应的顺序在“指令和回应的先后顺序”一节中讨论，使用指令的相关图表在“典型的FTP过程”一节中提供。

FTP指令可以分为这些指定的访问控制标识符，数据传输参数，或FTP服务请求。确定的指令（比如ABOR，STAT，QUIT）可以在数据传输过程中在控制连接上传输。有些服务器不能同时监视数据和控制链路，这样服务器就要采取专门的措施了。暂且推荐下面安排好的格式：

1.用户系统插入Telnet“中断进程”（IP）信号到Telnet信息流。

2.用户系统发送Telnet“同步”信号。

3.用户系统插入指令（比如，ABOR）到Telnet信息流。

4.server-PI在收到“IP”后，扫描Telnet信息流查找唯一的一个FTP指令。

（对其他服务器来说，这种方式不是必需的，但是上面列出的动作没有意外的效果。）

4.2 FTP回应

FTP命令的回应是为了确保数据传输请求和过程进行同步，也是为了保证用户进程总能知道服务器的状态。每条指令最少产生一个回应，虽然可能会产生多于一个的回应。对后一种情况，多个回应必须容易分辨。另外，有些指令是连续产生的，比如USER, PASS 和ACCT, 或RNFR 和 RNTO。如果的此前指令已经成功，回应显示一个中间的状态。其中任何一个命令的失败会导致全部指令序列重新开始。

指令-回应顺序的细节在下面的状态图中清楚地表示。

一个FTP回应包含三个数字（以三个数字字符传送），并跟随一些文本。这些数字被用来自动检测下一步要进入的状态，文本则提供给人类用户。三个数字包含了足够多的编码信息，所以user进程（user-PI）无需检查文本，用户就可以适当地决定是丢弃它或允许它通过。特别是，文本可能是依赖于服务器的，很可能对同一个回应码而有不同的文本。

回应包括3个数字码，后面跟空格<SP>，再跟一行文本（最大行长已经指定），用Telnet的行结束码终止。不过也有多于一行文本的情况，此时应把完整的文本用括号括起来，让用户进程知道它何时可以停止读取回应（比如，停止正在控制连接上的输入过程）去做其他事。需要在第一行中用特殊的格式来指明文本多于一行，在最后一行指明它是最后一行。它们中至少要有一个回应码用来指明处理的状态。为了满足所有的功能，第一行和最后一行的回应码应一样。

所以，多行回应的格式，第一行首先是必需的回应码，紧跟一个连字符，“-”（也就是大家熟知的减号），然后跟随文本。最后一行以相同的回应码开头，紧跟一个空格<SP>，任意的文本，和Telnet行结束码。

例如：

123-第一行
第二行
  234 以数字开头的行
123 最后一行

用户进程只需在一行的开头查找第二次出现的相同回应码，后面跟着<SP>（空格），忽略所有的中间行。如果中间行有一个3位的数字，服务器必须在它前面填充空格以避免混淆。

该方案允许标准系统程序在第一行和最后一行添加“人为的”回应信息（比如STAT回应）。特殊情况下，这些程序可能会在某一行的开头产生三个数字和一个空格，所以每一行文本的开头应添加一些随意的文本，比如空格。

该方案还假定多行的回应没有嵌套。

回应的三个数字每个代表一个特殊的含义。这样做的目的是允许用非常简单的组合为用户进程提供非常复杂的回应。第一个数字表示回应是好的，坏的，或未完成的（参看状态图）。只通过第一个数字就可以让用户进程决定下一步怎么做（继续执行，重新执行，删除）。第二个数字为用户进程提供发生错误的大概类型（比如，文件系统错误，指令语法错误）。第三个数字提供最详细的信息（比如，RNTO指令前没有RNFR）。

回应码的第一个数字有5个值：

1yz　确定预备应答

请求的操作正在被初始化，在操作下一个指令前等待另外的应答。（用户进程在回应未完成时发送其他指令会妨碍协议的执行，不过如果此前进程中有指令，server-FTP进程可以把到达的指令排队。）此回应类型指出指令已经被接受，现在用户进程可以注意数据连接，因为同时监控执行有困难。server-FTP进程对每条指令最多回应一条1yz回应。

2yz　确定完成回应

请求的操作已经成功完成。可以开始新的请求。

3yz　确定中间回应

指令已经接受，但是请求的操作被暂停，等待接受更多的信息。用户应发送其他指令提供这些信息。此回应为指令序列组产生。

4yz　暂时拒绝完成回应

指令没有被接受，请求的操作没有发生，但是错误的条件是暂时的，可以继续请求该操作。用户应该返回指令序列的开头执行。理解“暂时”的意思很困难，尤其是对两个差异很大的站点。每个4yz类回应有一个稍微不同的时间值，目的是鼓励用户再试一次。一个简单的规则是，如果允许在指令的形式和用户与服务器的特性没有任何改变时（比如，指令的拼写和参数均相同，用户没有改变文件访问权限或用户名，服务器没有进行新的执行）可以重复指令，那么如果回应既可以是4yz也可以是5yz（永久拒绝），就使用4yz。

5yz　永久拒绝完成应答

指令未被接受，请求的操作没有发生。用户进程若重复相同的请求（相同的指令序列）仍会被拒绝。即使“永久”错误条件已经改正，人类用户想让他的用户进程在未来重新执行指令序列也不可以。（比如，拼写错误已改正，或用户已改变了目录状态。）

下面是第二个数字的功能编码：

x0z　语法——该类回应指出语法错误。指令虽拼写正确但不适合任何功能种类；未实现的或不必要的指令。

x1z　信息——该类回应请求信息，比如像状态或帮助信息。

x2z　连接——该类回应关于控制连接和数据连接。

x3z　认证和帐户——该类回应关于登录过程和帐户程序。

x4z　保留

x5z　文件系统——该类回应指出服务器文件系统的状态，与请求的传输或其他文件系统操作有关。

第三个数字给出了第二个数字指出的功能类别的更详细的信息。回应的列表在下面图示。注意，与每个回应关联的文本是推荐的，而不是强制

的，甚至可以依照指令改变这些关联的回应。另一方面，回应码必须严格遵守最后一节的规范说明，就是说，服务器的执行不能为其他情况发明新的指令，宁可使用稍有差别的描述，也不要改变已经定义好的回应码。

有些命令如TYPE或ALLO，它们的成功不为用户进程提供任务新信息，只有200应答返回。如果因为不适合计算机系统，比如ALLO用在TOPS20站点上，导致特定的server-FTP进程没有执行该指令，仍要返回一个确定完成应答，以便让简单的用户进程知道它可以按自己的操作过程继续进行。202回应的文本可以是“没有可分配的存储空间”。另外，如果要求的不是可选实现的命令，而这个命令确实没有实现，那就要返回代码502。如果一个指令被执行，但是有未生效的参数，那么返回504回应。

4.2.1 按功能分组的回应码

200　指令成功。
500　语法错误，未被承认的指令。它包含像指令行太多这样的错误。
501　因参数或变量导致的语法错误。
202　指令未执行，对这个站点来说是不必要的。
502　指令未执行。
503　错误的指令顺序。
504　指令因参数的原因没有执行。

110　重新开始标记应答。在这种情况下文本是确定的，它必须读取：
MARK yyyy=mmmm
其中yyyy是用户进程数据流标记，mmmm是服务器等价的标记（注意，两个标记之间用“=”连接）。
211　系统状态，或系统帮助回应。
212　目录状态。
213　文件状态。
214　帮助消息。如何使用服务器或一个不标准的指令的特殊含义。该回应只对人类用户有用。
215　名字系统类型　该名字是编号分配文档里列出的正式系统名。

120　服务在 nnn 分钟内准备好。
220　服务为新用户准备好。
221　服务关闭控制连接。适当时退出。
421　服务无效，关闭控制连接。当服务知道必须关机时对任何指令做出的回应。
125　数据连接已经打开，传输开始。
225　数据连接打开，没有传输进行。
425　不能建立数据连接。
226　关闭数据连接。请求的文件操作成功（比如，文件传输或文件终止）。
426　连接关闭；传输终止。
227　进入被动模式（h1,h2,h3,h4,p1,p2）。

230　用户登录，继续。

530 没有登录。
331 用户名正确，需要口令。
332 需要登录帐户。
532 需要存储文件的帐户。

150 文件状态正确；关于建立数据连接。
250 请求的文件操作正确，完成。
257 “路径名”已建立。
350 请求的文件操作需要更多的信息。
450 请求的文件操作没有执行。文件无效（比如，文件忙）。
550 请求的操作没有执行。文件无效（比如，文件没找到，没有权限）。
451 请求操作终止。进程发生局部错误。
551 请求操作终止。页类型未知。
452 请求操作没有执行。系统存储空间不足。
552 请求的文件操作终止。超出可分配的存储空间（对当前目录或数据集来说）。
553 请求的操作没有执行。

4.2.2 按数字排序的回应码

110 重新开始标记应答。在这种情况下文本是确定的，它必须读取：
MARK yyyy=mmmm
其中yyyy是用户进程数据流标记，mmmm是服务器等价的标记（注意，两个标记之间用"="连接）。
120 服务在nnn分钟内准备好。
125 数据连接已经打开，传输开始。
150 文件状态正确；关于建立数据连接。

200 指令成功。
202 指令未执行，对这个站点来说是不必要的。
211 系统状态，或系统帮助回应。
212 目录状态。
213 文件状态。
214 帮助消息。如何使用服务器或一个不标准的指令的特殊含义。该回应只对人类用户有用。
215 名字系统类型　该名字是编号分配文档里列出的正式系统名。
220 服务为新用户准备好。
221 服务关闭控制连接。适当时退出。
225 数据连接打开，没有传输进行。
226 关闭数据连接。请求的文件操作成功（比如，文件传输或文件终止）。
227 进入被动模式（h1,h2,h3,h4,p1,p2）。
230 用户登录，继续。
250 请求的文件操作正确，完成。
257 “路径名”已建立。

331 用户名正确，需要口令。
332 需要登录帐户。
350 请求的文件操作需要更多的信息。

421 服务无效，关闭控制连接。当服务知道必须关机时对任何指令做出的回应。
425 不能建立数据连接。
426 连接关闭；传输终止。
450 请求的文件操作没有执行。文件无效（比如，文件忙）。
451 请求操作终止。进程发生局部错误。
452 请求操作没有执行。系统存储空间不足。

500 语法错误，未被承认的指令。它包含像指令行太多这样的错误。
501 因参数或变量导致的语法错误。
502 指令未执行。
503 错误的指令顺序。
504 指令因参数的原因没有执行。
530 没有登录。
532 需要存储文件的帐户。
550 请求的操作没有执行。文件无效（比如，文件没找到，没有权限）。
551 请求操作终止。页类型未知。
552 请求的文件操作终止。超出可分配的存储空间（对当前目录或数据集来说）。
553 请求的操作没有执行。

5. 公布的规范

5.1 最小实现

为了使FTP可以工作，并且没有不必要的错误消息，要求所有服务器实现下面的最小实现：

类型：ASCII 非打印
模式：流模式
结构：文件结构，记录结构
指令：USER, QUIT, PORT,
TYPE, MODE, STRU,
需要缺省值
RETR, STOR,
NOOP。
传输的缺省值是：
类型：ASCII 非打印
模式：流模式
结构：文件结构
所有的主机必须接受上面的标准缺省值。

5.2 连接

服务器协议解释器会在端口L侦听，用户或用户协议解释器初始化全双工控制连接。服务器和用户进程应该遵从ARPT-Internet协议手册[1]说明的Telnet协议进行。服务器不提供对命令行的编辑功能，由用户主机负责这一切。在全部传送和应答结束后，在用户的请求下服务器关闭控制连接。

user-DTP必须在指定的数据端口上“监听”，它可以是默认端口U或由PORT指令指定的端口。服务器从它自己的缺省数据端口（L-1）初始化数据连接。传输的方向和使用的端口由FTP服务指令决定。

注意，所有的FTP执行必须支持使用缺省的端口进行数据传输，只有user-PI可以初始化使用非缺省的端口。

当数据在两个服务器A和B之间传送时（如图2），用户PI，C，在两个服务器PI之间建立控制连接。其中一个服务器，比如A，接收到PASV指令，告诉它在自己数据端口“监听”，而不是当它接收到传输服务指令时开始连接。当user-PI接收到PASV指令确认时，该指令包含正在监听的同一主机和端口号，user-PI把A的端口号a用PORT指令发送给B，接收到B的回应，然后user-PI就可以给A和B发送相应的服务指令了。服务器B初始连接和传输进程。指令和回应的顺序在下面列出，有些消息是纵向同步的，有些是横向异步的。

```
User-PI - 服务器 A                  User-PI - 服务器 B
------------------                  ------------------

C->A : 连接                              C->B : 连接
C->A : PASV
A->C : 227 进入被动模式. A1,A2,A3,A4,a1,a2
                                         C->B : PORT A1,A2,A3,A4,a1,a2
                                         B->C : 200 Okay
C->A : STOR                              C->B : RETR
          B->A : 连接到主机A, 端口a
```

图3

数据连接在出现“建立数据连接”一节中描述的条件时会被服务器关闭。如果数据连接在数据传输后被关闭，这时就不必指定文件结束符，可以直接关闭连接，服务器必须马上执行。直到新的传输指令不被允许后才可以停止等待，因为用户进程已经在检测数据连接，查看是否需要“监听”（记住，在发送传输请求之前，用户必须“监听”已关闭的数据端口）。为了防止空转情况的出现，服务器应该在关闭数据连接后发送一个226回应（或者，如果连接已经关闭，发送“文件传输完成”的250回应，user-PI在发送新的传输指令前监听其中的一个回应）。

任何时候用户和服务器发现另一方关闭了连接，它应该迅速读取任何连接中的剩余数据，并关闭自己这边的连接。

5.3 指令

“FTP指令”一节描述的指令是Telnet字串，它们在控制连接上传输。指令功能和语义已在“访问控制指令”,“传输参数指令”“FTP服务指令”各节中描述。指令语法也已指定。

指令由指令码开始，后跟参数域。指令码是四个或更少的文字字符。文字字符不区分大小写。所以，任何下列指令都表示获得文件指令：

RETR　　Retr　　retr　　ReTr　　rETr

用符号表示的参数值也不区分大小写，如ASCII类型的A或a。指令码和参数域由一个或更多的空格分隔。

参数域由可变长的字符串组成，结尾是NVT-ASCII表示的字符序列<CRLF>（回车，换行）。其他的谈判语言可能使用不同的行结束符。注意，服务器在未接收到行结束符时不会采取任何动作。

语法由下面的NVT-ASCII码指定。参数域的所有的字符，都是ASCII字符，包括用ASCII表示的十进制整数。方括号表示一个可选的参数域。如果可选的参数没有，就使用默认的缺省值。

5.3.1 FTP指令

下面是FTP指令：

```
USER <SP> <username> <CRLF>
PASS <SP> <password> <CRLF>
ACCT <SP> <account-information> <CRLF>
CWD  <SP> <pathname> <CRLF>
CDUP <CRLF>
SMNT <SP> <pathname> <CRLF>
QUIT <CRLF>
REIN <CRLF>
PORT <SP> <host-port> <CRLF>
PASV <CRLF>
TYPE <SP> <type-code> <CRLF>
STRU <SP> <structure-code> <CRLF>
MODE <SP> <mode-code> <CRLF>
RETR <SP> <pathname> <CRLF>
STOR <SP> <pathname> <CRLF>
STOU <CRLF>
APPE <SP> <pathname> <CRLF>
ALLO <SP> <decimal-integer>
    [<SP> R <SP> <decimal-integer>] <CRLF>
REST <SP> <marker> <CRLF>
RNFR <SP> <pathname> <CRLF>
RNTO <SP> <pathname> <CRLF>
ABOR <CRLF>
```

```
DELE <SP> <pathname> <CRLF>
RMD  <SP> <pathname> <CRLF>
MKD  <SP> <pathname> <CRLF>
PWD  <CRLF>
LIST [<SP> <pathname>] <CRLF>
NLST [<SP> <pathname>] <CRLF>
SITE <SP> <string> <CRLF>
SYST <CRLF>
STAT [<SP> <pathname>] <CRLF>
HELP [<SP> <string>] <CRLF>
NOOP <CRLF>
```

5.3.2 FTP指令的参数变量

上面指令的参数域的语法（使用BNF符号表示）：

```
<username> ::= <string>
<password> ::= <string>
<account-information> ::= <string>
<string> ::= <char> | <char><string>
<char> ::= any of the 128 ASCII characters except <CR> and
<LF>
<marker> ::= <pr-string>
<pr-string> ::= <pr-char> | <pr-char><pr-string>
<pr-char> ::= printable characters, any
              ASCII code 33 through 126
<byte-size> ::= <number>
<host-port> ::= <host-number>,<port-number>
<host-number> ::= <number>,<number>,<number>,<number>
<port-number> ::= <number>,<number>
<number> ::= any decimal integer 1 through 255
<form-code> ::= N | T | C
<type-code> ::= A [<sp> <form-code>]
              | E [<sp> <form-code>]
              | I
              | L <sp> <byte-size>
<structure-code> ::= F | R | P
<mode-code> ::= S | B | C
<pathname> ::= <string>
<decimal-integer> ::= any decimal integer
```

5.4 指令和回应的先后顺序

用户和服务器之间的通信是交互进行的。用户给服务器发送一条FTP指令，服务器给一个回应。用户在发出其他指令前应等待原来指令的成功或失败响应。

有些指令还需要第二个回应，用户也需等待。这些回应会报告进程或文件传输的完成或关闭数据连接的情况。它们是文件传输指令的二级回应。

最重要的一组报告的回应之一是连接问候语。一般情况下，当连接成功后，服务器会发送220回应“等待输入”。用户在发送其他指令之前应等待问候语消息。如果服务器不能马上接受输入，会马上发送一个120回应“延迟稍侯”，之后220回应准备好。这样用户就知道虽然有一个小的延迟但是系统没有挂断。

自动回应

有时“系统”会自动地给用户（通常是所有用户）发送一个消息。比如，“系统将在15分钟后关闭”。 FTP没有提供服务器直接给用户发送这样的消息的功能。推荐把这样的信息放到server-PI里排队，在下一个回应里发送给user-PI（可能是一个多行的回应）。

下表列出每个指令的成功和失败响应。必须严格遵守：服务器可以替换回应的文本，但是回应码的含义和暗指的动作，还有特定指令回应的顺序都不能改变。

指令和回应的顺序

本节讨论指令和回应的先后顺序。每个指令列出了它的可能的回应，指令组列在一起。首先列出预备回应，然后是确定完成回应和拒绝完成回应，最后是中间回应。这些列出的格式是状态图的基础，状态图单独给出。

建立连接

```
        120
            220
        220
        421
```

登录

```
        USER
            230
            530
            500, 501, 421
            331, 332
        PASS
            230
            202
            530
            500, 501, 503, 421
            332
        ACCT
            230
            202
```

530
500, 501, 503, 421
CWD
250
500, 501, 502, 421, 530, 550
CDUP
200
500, 501, 502, 421, 530, 550
SMNT
202, 250
500, 501, 502, 421, 530, 550

注销退出

REIN
120
220
220
421
500, 502
QUIT
221
500

传输参数

PORT
200
500, 501, 421, 530
PASV
227
500, 501, 502, 421, 530
MODE
200
500, 501, 504, 421, 530
TYPE
200
500, 501, 504, 421, 530
STRU
200
500, 501, 504, 421, 530

文件操作指令

ALLO
200
202

```
   500, 501, 504, 421, 530
REST
   500, 501, 502, 421, 530
   350
STOR
   125, 150
      (110)
      226, 250
      425, 426, 451, 551, 552
   532, 450, 452, 553
   500, 501, 421, 530
STOU
   125, 150
      (110)
      226, 250
      425, 426, 451, 551, 552
   532, 450, 452, 553
   500, 501, 421, 530
RETR
   125, 150
      (110)
      226, 250
      425, 426, 451
   450, 550
   500, 501, 421, 530
LIST
   125, 150
      226, 250
      425, 426, 451
   450
   500, 501, 502, 421, 530
NLST
   125, 150
      226, 250
      425, 426, 451
   450
   500, 501, 502, 421, 530
APPE
   125, 150
      (110)
      226, 250
      425, 426, 451, 551, 552
   532, 450, 550, 452, 553
   500, 501, 502, 421, 530
```

RNFR
450, 550
500, 501, 502, 421, 530
350
RNTO
250
532, 553
500, 501, 502, 503, 421, 530
DELE
250
450, 550
500, 501, 502, 421, 530
RMD
250
500, 501, 502, 421, 530, 550
MKD
257
500, 501, 502, 421, 530, 550
PWD
257
500, 501, 502, 421, 550
ABOR
225, 226
500, 501, 502, 421

报告指令

SYST
215
500, 501, 502, 421
STAT
211, 212, 213
450
500, 501, 502, 421, 530
HELP
211, 214
500, 501, 502, 421

其他指令

SITE
200
202
500, 501, 530
NOOP
200

500 421

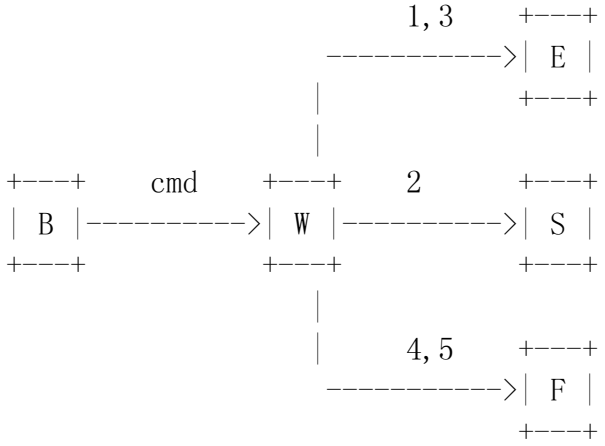
6. 状态图

这里给出状态图以便于理解FTP的执行。只使用了回应码的第一位。每个FTP指令组或指令序列有一个状态图。

指令的分组由每个指令的构造模型来决定，因此把结构模型相同的指令放在一起。

每个指令或指令序列有三个可能的结果：成功（S），失败（F），和错误（E）。下面的状态图中使用符号B代表“开始”，符号W代表“等待回应”。

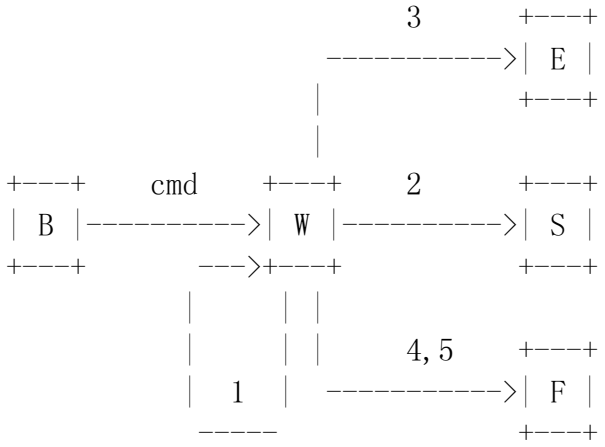
我们先给出FTP指令最大组的状态图：



适合该图模型的指令有：

ABOR, ALLO, DELE, CWD, CDUP, SMNT, HELP, MODE, NOOP, PASV, QUIT, SITE, PORT, SYST, STAT, RMD, MKD, PWD, STRU, 和 TYPE.

另一个指令的大组的表现形式与上图类似：



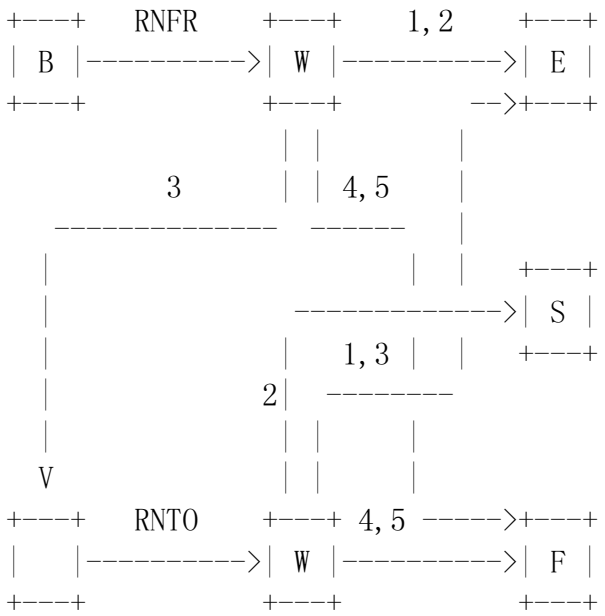
适合该图模型的指令有：

APPE, LIST, NLST, REIN, RETR, STOR, 和 STOU

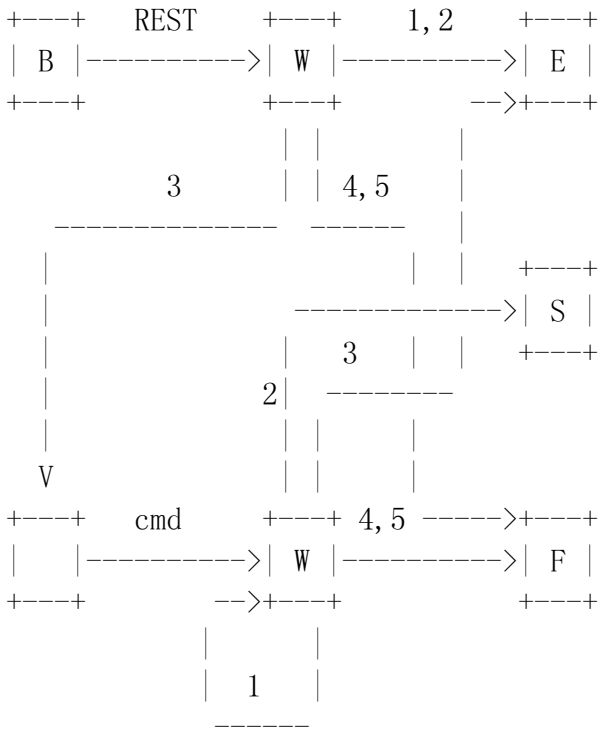
注意，第二个模型也可以用来表示第一组指令，它们仅有的不同是：

第一个模型中100系列回应是预料不到的，因此被视为错误；第二个模型中允许（有时候需要）100系列回应。记住，每个指令最多只能有一个100系列的回应。

剩余的模型中可能重命名的顺序是最简单的：



下图是重新开始的简单模型：

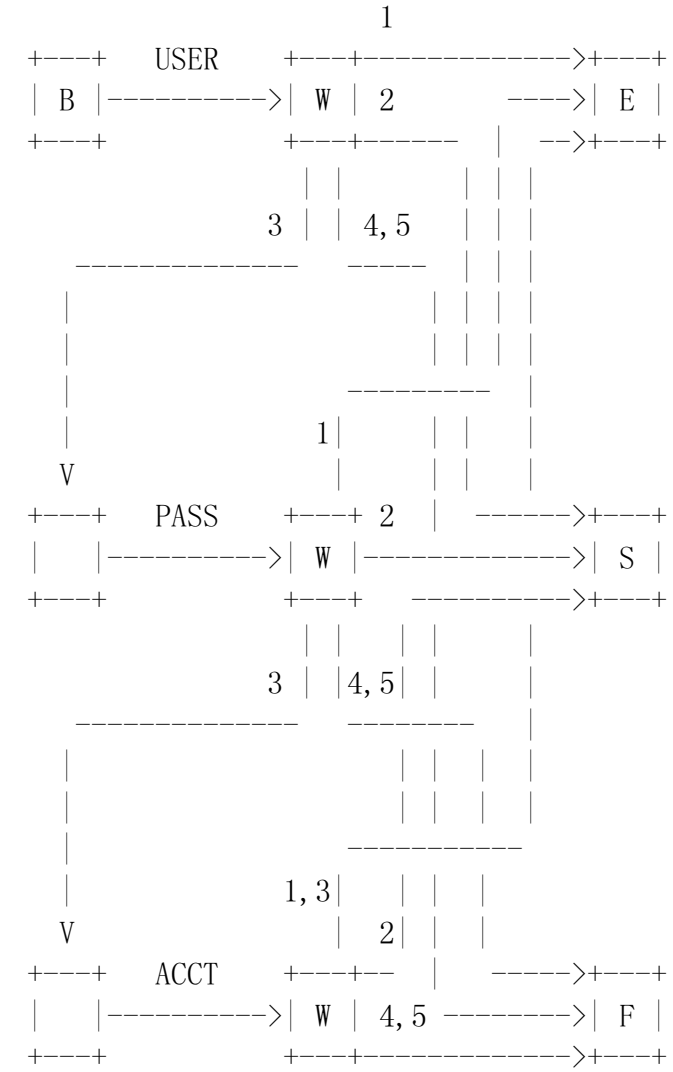


这里的“cwd”是APPE, STOR, 或 RETR。

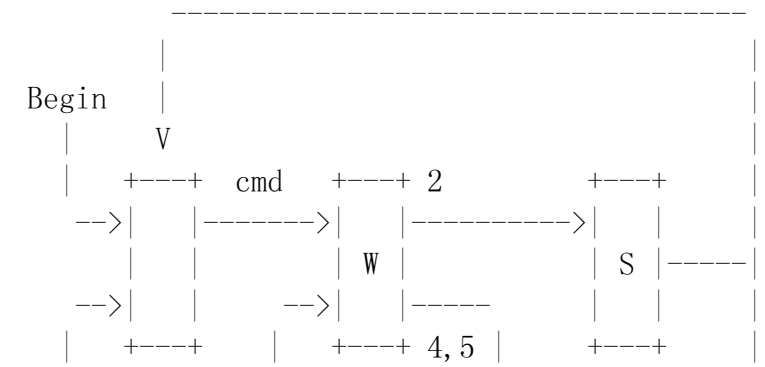
我们注意到上面的三个模型都很相似。重新开始与重命名的不同仅是

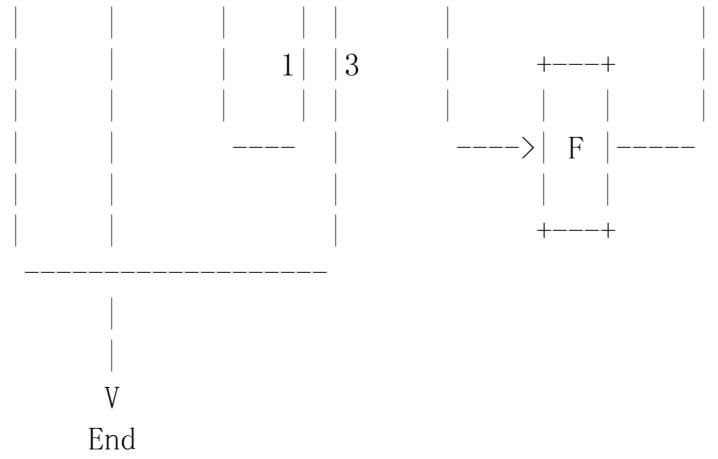
在第二阶段对100系列的回应。模型中允许（有时候需要）100系列回应。记住，每个指令最多只能有一个100系列的回应。

最复杂的状态图是登录顺序：



最后，我们给出一个概括的状态图，它用来表示指令和回应交互的模型：





7. 典型的FTP过程

位于主机U的用户希望与主机S进行文件传输（上传或下载）：

一般地，用户会通过一个间接的user-FTP进程与服务器通信。下面是一个典型的过程。user-FTP的提示用圆括号表示，“---->”表示从主机U到主机S的指令，“<----”表示从主机S到主机U的回应。

用户的本地指令	相关的操作
ftp (host) multics<CR>	连接到主机S的L端口, 建立控制连接。 <---- 220 Service ready <CRLF>.
username Doe <CR>	USER Doe<CRLF>----> <---- 331 User name ok, need password<CRLF>.
password mumble <CR>	PASS mumble<CRLF>----> <---- 230 User logged in<CRLF>.
retrieve (local type) ASCII<CR>	
(local pathname) test 1 <CR>	User-FTP opens local file in ASCII.
(for. pathname) test.pl1<CR>	RETR test.pl1<CRLF> ----> <---- 150 File status okay; about to open data connection<CRLF>. 服务器于端口 U 建立数据连接 <---- 226 Closing data connection, 文件传输成功<CRLF>.
type Image<CR>	TYPE I<CRLF> ----> <---- 200 Command OK<CRLF>
store (local type) image<CR>	
(local pathname) file dump<CR>	User-FTP opens local file in Image.
(for.pathname) >udd>cn>fd<CR>	STOR >udd>cn>fd<CRLF> ----> <---- 550 Access denied<CRLF>

terminate	QUIT <CRLF> ----> 服务器关闭所有连接。

8. 连接的建立

FTP控制连接是通过TCP在用户进程端口U和服务器进程端口L之间建立的。该协议分配服务器的端口号是21（八进制25），也就是L=21。

附录一 页结构

FTP对页结构的支持主要是因为对TOPS-20系统间文件高效传输的支持需要。

TOPS-20的文件系统基于页的概念。这个操作系统在使用页方式操作文件时是最高效的。该操作系统为文件系统提供了一个界面，所以许多应用程序查看文件就想连续的字符流。不过，有些应用程序从根本上是使用页结构的。

一个TOPS-20磁盘文件由4部分组成：一个路径名，一个页表，一个页面集（可能是空的），一个属性集。

路径名用RETR和STOR指令指定。它包括目录名，文件名，文件扩展名和产生的编号。

页表包含2**18个条目。每个条目可以是空的，页或者指向一个页面。如果不是空的，页有一些页面特性访问位；不是文件所有的页面都需要相同的访问保护。

一个页面大约512个字，每个字36位。

文件的属性写在文件描述符块（FDB）里，包括建立时间，写入时间，读取时间，作者的字节大小，文件结束指示器，读写的次数，备份系统磁带编号，等。

注意：没有要求页面表的条目相临。可以有空的页表。文件结束指示器只是一个编号。没有要求它实际指向文件的“最后一个”数据。TOPS-20里一般的连续I/O调用会引起文件结束指示器离开最后写入过的位置，但是其他的操作不会引起这样的后果。

TOPS-20页结构文件可以用FTP传输参数来发送：TYPE L 36, STRU P, and MODE S（实际上，任何模式都可以）。

每个信息页有一个页头。每个页头域是一个逻辑字节，因为类型是L 36，所以是一个TOPS-20字。

页头域包括：

字0：页头长度。

页头长度是5。

字1：页面索引。

如果数据是一个磁盘文件页面，那么这就是文件页面地图中页面的编号。文件中的空页（holes）不会被发送。注意，这个hole页和全是0的页面不一样。

字2：数据长度。

本页的数据字数量，跟随在页头的后面。所以，传输的总长度是页头长度加上数据长度。

字3：页类型。

页面类型的代码。数据页的类型是3，FDB页的类型是2。

字4：页面访问控制。

文件页面地图中与页相关的访问位。

页头后是数据长度数据字。数据长度一般对于数据页是512，对于FDB是31。当跟踪到0磁盘文件页面时就丢弃，这样会使数据长度小于512。

附录二 目录指令

UNIX的树状目录结构可以提供简单的操作，FTP服务器把它扩展利用到机器中创建目录。因为ARPA-Internet上的其他树状目录结构的主机（包括TOPS-20和Multics），这些指令可能会用到。

FTP加入了四个目录指令：

MKD 路径名

创建一个名为“路径名”的目录。

RMD 路径名

移动名为“路径名”的目录。

PWD

打印当前工作目录名。

CDUP

转到当前工作目录的父目录。

“目录名”参数是在当前工作目录下要创建（或删除）的子目录，除非“路径名”包含了足够的信息指定了另外的服务器，比如，“路径名”是绝对路径名（在 UNIX 和 Multics中），或者路径名是TOPS-20中像“<abso.lute.path>”的路径。

回应码

CDUP指令是CWD的特殊情况，它对在不同的操作系统间传输目录树的程序有简化执行的作用，因为不同的操作系统对父目录的命名语法不同。CDUP的回应码与CWD的回应码一样。

RMD的回应码与DELE对文件的回应一样。

MKD的回应码稍微复杂一些。一个刚建立的目录可能是未来CWD指令要操作的对象。不幸的是，通常MKD的参数不是CWD的参数。比如这种情况，当TOPS-20建立了一个子目录，那么，对于TOPS-20的服务器FTP，该指令序列

```
MKD MYDIR
CWD MYDIR
```

将会失败。新目录只能使用“绝对”名字来访问。比如，如果MDK指令运行在连接的目录<DFRANKLIN>上，那么新子目录只能使用名字<DFRANKLIN.MYDIR>。

不过，甚至是在UNIX和Multics上，MKD的参数也不是合适的。如果是一个“相对”路径名（比如，一个相对于当前路径的路径名），用户为了到

达子目录，就需要到同样的当前目录。由于依赖于应用程序，这样做有困难。在此情况下非常不坚固。

为了解决这些问题，在成功完成一个MKD指令后，服务器应返回一个这样格式的行：

257<space>"<directory-name>"<space><commentary>

就是说，服务器要告诉用户如果要引用已建立的目录应使用什么字符串。目录名可以包含任意的字符。要嵌入的双引号需要两个双引号。

例如，一个用户连接到目录/usr/dm，并且建立了一个子目录，命名为pathname：

```
CWD /usr/dm
200 directory changed to /usr/dm
MKD pathname
257 "/usr/dm/pathname" directory created
```

一个嵌入双引号的例子：

```
MKD foo"bar
257 "/usr/dm/foo""bar" directory created
CWD /usr/dm/foo"bar
200 directory changed to /usr/dm/foo"bar
```

如果此前存在一个相同名字的的子目录，此时服务器必须返回一个“访问拒绝”的错误回应。

```
CWD /usr/dm
200 directory changed to /usr/dm
MKD pathname
521-"/usr/dm/pathname" directory already exists;
521 taking no action.
```

MKD的失败回应与STOR的回应类似。如果要在一个子目录中创建文件，而此前已有一个相同名字的文件存在，那么返回“访问拒绝”的回应以避免冲突（这在UNIX中是个问题，但在TOPS-20中不是）。

本质上因为PWD指令返回和成功的MKD指令相同类型的信息，所以成功的PWD指令的回应也使用257回应码。

细微之处

因为这些指令大多数用于从一个机器到另一个机器传输子目录，MKD的参数被解释成当前工作目录的子目录，除非它包含足够多的目标主机的信息。一个假想的在TOPS-20世界中使用的例子：

```
CWD <some.where>
200 Working directory changed
MKD overrainbow
257 "<some.where.overrainbow>" directory created
CWD overrainbow
431 No such directory
CWD <some.where.overrainbow>
200 Working directory changed

CWD <some.where>
200 Working directory changed to <some.where>
MKD <unambiguous>
257 "<unambiguous>" directory created
CWD <unambiguous>
```

注意，第一个例子的结果是连接目录的子目录。相应的，第二个例子的参数包含了足够多的TOPS-20信息，说明<unambiguous>是一个顶级目录。还要注意，第一个例子中用户“违反”了协议，因为他尝试访问刚刚建立的子目录。问题的起因是已经有了一个<overrainbow>目录，它造成TOPS-20系统执行时的内部混淆。相似的情况也会出现在RMD指令上。除非指明是绝对路径，否则主机会把MKD和RMD的参数视为子目录。MKD的257回应必须包含创建目录的绝对路径名。

附录三 关于FTP的RFC

Bhushan, Abhay, "文件传输协议", RFC 114 (NIC 5823), MIT-Project MAC, 1971年4月16日。

Harslem, Eric, and John Heafner, "对RFC 114的注释(文件传输协议)", RFC 141 (NIC 6726), RAND, 1971年4月29日。

Bhushan, Abhay, et al, "文件传输协议", RFC 172 (NIC 6794), MIT-Project MAC, 1971年6月23日。

Braden, Bob, "对DTP 和 FTP 提议的注释", RFC 238 (NIC 7663), UCLA/CCN, 1971年9月29日。

Bhushan, Abhay, et al, "文件传输协议", RFC 265 (NIC 7813), MIT-Project MAC, 1971年11月17日。

McKenzie, Alex, "对文件传输协议的一个附加建议",RFC 281 (NIC 8163), BBN, 1971年12月8日。

Bhushan, Abhay, "文件传输协议中“设置数据格式”处理的使用", RFC 294 (NIC 8304), MIT-Project MAC, 1972年1月25日。

Bhushan, Abhay, "文件传输协议", RFC 354 (NIC 10596), MIT-Project MAC, 1972年7月8日。

Bhushan, Abhay, "关于文件传输协议(RFC 354)的注释",RFC 385 (NIC 11357), MIT-Project MAC, 1972年8月18日。

Hicks, Greg, "用户 FTP 文件", RFC 412 (NIC 12404), Utah, 1972年11月27日。

Bhushan, Abhay, "文件传输协议 (FTP) 的状态和更进一步的注释", RFC 414 (NIC 12406), MIT-Project MAC, 1972年11月20日。

Braden, Bob, "关于文件传输协议的注释", RFC 430 (NIC 13299), UCLA/CCN, 1973年2月7日。

Thomas, Bob, and Bob Clements, "FTP 服务器对服务器的交互",RFC 438 (NIC 13770), BBN, 1973年1月15日。

Braden, Bob, "在FTP下打印文件", RFC 448 (NIC 13299), UCLA/CCN, 1973年2月27日。

McKenzie, Alex, "文件传输协议", RFC 454 (NIC 14333), BBN, 1973年2月16日。

Bressler, Bob, and Bob Thomas, "通过FTP进行邮件恢复", RFC 458 (NIC 14378), BBN-NET and BBN-TENEX, 1973年2月20日。

Neigus, Nancy, "文件传输协议", RFC 542 (NIC 17759), BBN, 1973年7月12如，

Krilanovich, Mark, and George Gregg, "文件传输协议的注释", RFC 607 (NIC 21255), UCSB, 1974年1月7日。

Pogran, Ken, and Nancy Neigus, "对RFC 607的回应 - 文件传输协议的注释", RFC 614 (NIC 21530), BBN, 1974年1月28日。

Krilanovich, Mark, George Gregg, Wayne Hathaway, and Jim White, "文件传输协议的注释", RFC 624 (NIC 22054), UCSB, Ames Research Center, SRI-ARC, 1974年2月28日。

Bhushan, Abhay, "FTP注释and 对RFC 430的回应", RFC 463 (NIC 14573), MIT-DMCG, 1973年2月21日。

Braden, Bob, "FTP 数据压缩", RFC 468 (NIC 14742), UCLA/CCN, 1973年3月8日。

Bhushan, Abhay, "FTP and 网络邮件系统", RFC 475 (NIC 14919), MIT-DMCG, 1973年3月6日。

Bressler, Bob, and Bob Thomas "FTP 服务器对服务器的交互 - II",RFC 478 (NIC 14947), BBN-NET and BBN-TENEX, 1973年3月26日。

White, Jim, "通过NIC杂志使用FTP", RFC 479 (NIC 14948), SRI-ARC, 1973年3月8日。

White, Jim, "依赖主机的 FTP 参数", RFC 480 (NIC 14949), SRI-ARC, 1973年3月8日。

Padlipsky, Mike, "一个 FTP 指令命名问题", RFC 506 (NIC 16157), MIT-Multics, 1973年6月26日。

Day, John, "FTP族的备忘录 (文件访问协议的建议)", RFC 520 (NIC 16819), Illinois, 1973年6月25日。

Merryman, Robert, "UCSD-CC Server-FTP工具", RFC 532 (NIC 17451), UCSD-CC, 1973年6月22日。

Braden, Bob, "TENEX FTP 问题", RFC 571 (NIC 18974), UCLA/CCN, 1973年11月15日。

McKenzie, Alex, and Jon Postel, "Telnet and FTP 执行 -进度表的改变", RFC 593 (NIC 20615), BBN and MITRE, 1973年11月29日。

Sussman, Julie, "适合于更可靠的邮件服务的FTP错误指令用法", RFC 630 (NIC 30237), BBN, 1974年4月10日。

Postel, Jon, "修订的FTP回应码", RFC 640 (NIC 30843), UCLA/NMC, 1974年6月5日。

Harvey, Brian, "不要弄巧成拙", RFC 686 (NIC 32481), SU-AI, 1975年5月10日。

Harvey, Brian, "再试一次FTP", RFC 691 (NIC 32700), SU-AI, 1975年5月28日。

Lieb, J., " FTP的CWD指令", RFC 697 (NIC 32963), 1975年7月14日。

Harrenstien, Ken, "FTP 扩展: XSEN", RFC 737 (NIC 42217), SRI-KL, 1977年10月31日。

Harrenstien, Ken, "FTP 扩展: XRSQ/XRCP", RFC 743 (NIC 42758), SRI-KL, 1977

年12月30日。

Lebling, P. David, ″ FTP Mail and MLFL的调查″, RFC 751, MIT, 1978年12月10日。

Postel, Jon, ″文件传输协议规范″, RFC 765, ISI, 1980年6月。

Mankins, David, Dan Franklin, and Buzz Owen, ″适合目录的FTP指令″, RFC 776, BBN, 1980年12月。

Padlipsky, Michael, ″FTP 唯一名字存储指令″, RFC 949, MITRE, 1985年7月。

参考书目

[1] Feinler, Elizabeth, ″Internet协议转换手册″,Network Information Center, SRI International, 1982年3月。

[2] Postel, Jon, ″传输控制协议- DARPA Internet程序协议规范″, RFC 793, DARPA, 1981年9月。

[3] Postel, Jon, and Joyce Reynolds, ″Telnet 协议规范″, RFC 854, ISI, 1983年5月。

[4] Reynolds, Joyce, and Jon Postel, ″编号的分配″, RFC 943, ISI, 1985年4月。

原文：RFC 959《File Transfer Protocol》
译者：comehope 2002年5月
博客：http://www.comehope.com