RFC959 FTP 传输协议

目 录

1. 介绍

FTP 的目标是：1）促进程序/数据文件的共享；2）鼓励（通过程序）使用远程计算机；3）使用户不必面对不同主机上不同文件系统的差异；4）对数据进行高效可靠的传输。FTP 尽管可以直接在终端上应用，但它主要被设计通过程序来使用。

本规范通过设计简单易实现的协议来试图满足大型机、小型机、个人工作站、TAC 等用户的需要。

本文需要文件传输协议（TCP）[2]以及 Telnet 协议[3]的知识。关于它们的文档可以在 ARPA-互联网协议手册[1]中找到。

2. 概述

本章讨论内容包括历史、术语、FTP 模型。本章所描述的都是关于 FTP 的重要内容。一些术语专用于 FTP 模型；一些读者可能有必要在回顾这些术语时参考关于 FTP 模型的章节。

2.1. 历史

FTP 的发展经历了很多年。附录 III 是按年代编辑的关于 FTP 的 RFC 文档。其中包括最早在 1971 年提出用在 M.I.T.主机上的文件传输机制（RFC 114），以及在 RFC 141 中的注释和讨论。

RFC 172 提供了一个在主机（包括终端 IMP）间基于用户层协议的文件传输方法。RFC 265 做为其修订，通过附加评论重定义了 FTP，RFC 281 建议进一步改进。“Set Data Type”在传输中应用在 1982 年 1 月的 RFC 294 中提出。

RFC 354 废弃了 RFC 264 和 265。文件传输协议被定义为 ARPANET 上主机间的文件传输协议，FTP 的主要作用则被定义为用来在主机间高效可靠地传输文件以及对远程存储的方便使用。RFC 385 进一步讨论了协议的错误，重点和一些附加内容，RFC 414 提供了使用中的 FTP 状态报告。1973 年的 RFC 430（被其它 RFC 多次引用），提供了对 FTP 的进一步讨论。最后，一个“官方的”FTP 文件在 RFC 454 中发布。

至 1973 年 1 月，FTP 协议又有了大量的变化，但总体的结构仍保持一致。为了应对这些变化，RFC 542 中发布了新的“官方”规范。但很多基于老规范的应用并没有被更新。

1974 年，RFC 607 和 614 继续讨论 FTP。RFC 624 提出进一步改变设计和一些小的修补。1975 年，RFC 686 用题为“Leaving Well Enough Alone”讨论早期以及近期 FTP 版本的差别。RFC 691 对 RFC 686 中关于打印文件部分做了小的修订。

在之前所有努力的基础上，通过将底层的传输协议由 NCP 改为 TCP，应用 TCP 的 FTP 协议在 RFC 765 中诞生了。

本文描述的 FTP 规范目的在于纠正之前文档中的某些小错误，进一步解释一些协议特性，以及引入一些可选的命令。

特别地，本版本规范新包含了以下可选命令：
CDUP - 返回父目录
SMNT - 结构装备
STOU - 唯一保存
RMD - 删除目录
MKD - 新建目录
SYST - 系统
本规范兼容之前版本。应用在前一版本的程序应该可以自动应用于本规范。

2.2. 术语

ASCII

ASCII 字符集定义于 ARPA-互联网协议手册。FTP 中 ASCII 字符指 8 位编码集的低半部（也就是最高位为 0）。

访问控制（access controls）

访问控制定义了用户使用系统、系统文件的访问权力。访问控制必要性在于防止未被授权的或意外的对文件的访问。在服务器端使用访问控制正是 FTP 的优势所在。

字节长度（byte size）

有两种字节长度与 FTP 有关：文件的逻辑字节长度和用来传输数据的字节长度。传输字节长度总是 8 位。传输字节长度不必要和系统存储数据的字节长度或用来描述数据结构的逻辑字节长度相同。

控制连接（control connection）

用户 PI 与服务器 PI 用来交换命令和响应的信息路径。这个连接遵守 Telnet 协议。

数据连接（data connection）

用规定的模式和类型进行数据传输的全双向连接。传输的数据可能是文件的一部分、整个文件或一些文件。传输路径可能是服务器 DTP 与用户 DTP 之间或两个服务器 DTP 之间。

数据端口（data port）

数据接收者在数据端口监听，等待数据传输者从此端口建立数据连接。

DTP（data transfer process）

数据传输过程，用以建立并管理数据连接。DTP 可以是被动或主动。

行结束符（End-of-Line）

行结束符序列定义了印刷行间的分隔。行结束符序列为回车符加换行符。

文件结束符（End-of-File）

文件结束符标志定义了传输中一个文件的结束。

记录结束符（End-of-Record）

记录结束符标志定义了传输中一个记录的结束。

错误恢复（error recovery）

允许用户从一些诸如主机系统或传输过程失败中恢复。FTP 中，错误恢复可以是在给定点上重新开始文件传输。

FTP 命令（FTP commands）

用户 FTP 到服务器 FTP 的控制信息流由一些命令集合组成。

文件（file）

计算机数据的有序集合（包括程序），有任意的大小，由路径名唯一指定。

模式（mode）

通过数据连接传输数据的方式。模式定义了包括 EOR 和 EOF 的数据传输格式。FTP 的传输模式在传输模式一章中介绍。

NVT（Network Virtual Terminal）

网络虚拟终端在 Telnet 协议中定义。

NVFS（Network Virtual File System）

网络虚拟文件系统用标准命令和路径规定定义了标准网络文件系统概念。

页（page）

一个文件可能被分为彼此独立部分的结构集合，称为页。FTP 支持将不连续的文件作为独立的页来传输。

路径名（pathname）

路径名定义为用户为了指定文件系统中某个特定文件而输入的字符串。路径一般包括驱动器名，目录名，以及文件名。FTP 尚未规定标准的路径名规则。用户必须传输方文件系统的命名规则。

PI（protocol interpreter）

协议解析器。用户和服务器用其来解析协议，它们的具体实现分别称为用户 PI 和服务器 PI。

记录（record）

顺序文件可能被由很多连续的部分组成称为记录。FTP 支持记录结构，但文件不是必须具备记录结构。

响应（reply）

响应是由服务器发给用户的对 FTP 命令的回应（肯定或否定）。一般的响应格式是完成码（包括错误码）跟上一个文本字符串。完成码用于程序，文本用于用户。

服务器 DTP（server-DTP）

数据传输过程，在通常的“主动”状态下是用“监听”的数据端口建立数据连接。它建立传输和存储参数，并在服务器端 PI 的命令下传输数据。服务器端 DTP 也可以用于“被动”模式，而不是主动在数据端口建立连接。

服务器 FTP 过程（server-FTP process）

与用户 FTP 过程或另一个服务器 FTP 过程配合实现文件传输功能。由协议解析器（PI）和数据传输过程（DTP）组成。

服务器 PI（server-PI）

服务器 PI 在 L 端口“监听”用户协议解析器的连接请求并建立控制连接。它从用户 PI 接收标准的 FTP 命令，发送响应，并管理服务器 DTP

类型（type）

用于数据传输和存储的数据表示类型。类型暗示了在数据存储和数据传输之间的时间变化。FTP 中定义的表示类型在建立数据连接一章中描述。

用户（user）

希望得到文件传输服务的人或程序。用户可能直接与服务器 FTP 过程互相作用，但推荐使用用户 FTP 过程，协议的设计有利于自动化操作。

用户 DTP（user-DTP）

数据传输过程在数据端口“监听”服务器 FTP 过程的连接。如果两个服务器通过它来传输数据，用户 DTP 就为非活跃的。

用户 FTP 过程（user-FTP process）

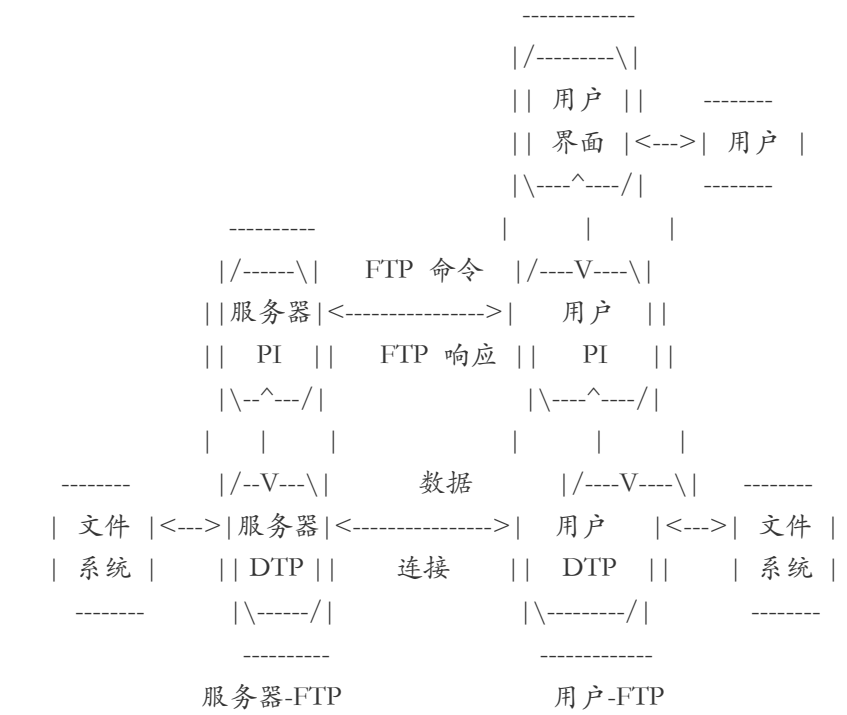
一系列功能集合，包括协议解析器、数据传输过程和用户界面，它们共同与服务器 FTP 过程配合完成文件传输功能。用户界面允许使用本地语言对用户发送命令响应。

用户 PI（user-PI）

用户协议解析器用 U 端口建立到服务器 FTP 过程的控制连接，并在文件传输时管理用户 DTP。

2.3. FTP 模型

了解了上面的定义，下面的模型（图一所示）用来描述一个 FTP 服务。



注意：1. 数据连接可能是任一方向。

2. 数据连接不必须一直存在。

图 1 FTP 使用模型

图 1 中描述的模型中，控制连接由用户 PI 发起。控制连接遵守 Telnet 协议。首先用户由用户 PI 产生标准 FTP 命令通过控制连接传输到服务器过程。（用户可能建立一个到服务器 FTP 的直接连接，例如从 TAC 终端不经过用户 FTP 过程直接产生标准的 FTP 命令。）标准响应由服务器端 PI 通过数据连接发送到用户 PI 作为命令的回应。

FTP 命令指定数据连接参数（端口，传输模式，表示类型，以及结构）和文件系统操作种类（store，retrieve，append，delete 等）。用户 DTP 则应在指定的数据端口“监听”，服务器用相应的参数发起数据连接并传送数据。而数据端口主机不一定必须与发送 FTP 命令

的主机一至，但用户或用户 FTP 过程要保证指定的端口处在“监听”下。
另需指出的是数据连接可能同时用于发送和接收数据。

另一种情形是用户可能要在两台远程主机间传送文件。用户分别与两台服务器建立控制连接，并安排两服务器间的数据连接。这种情况下，控制信息传送到用户 PI，但数据在两服务器间传送。以下是服务器-服务器交互模型：

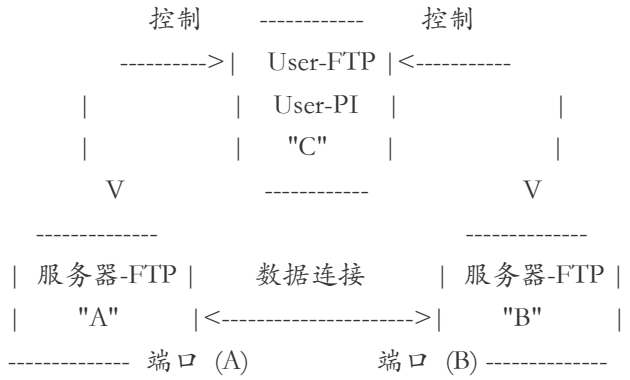


图 2

协议规定在数据传输过程中控制连接必须一直打开。当 FTP 服务使用完以后，用户应该要求服务器关闭控制连接。当没有发送关闭命令但控制连接事实已经关闭的情况下，服务器可能终止数据传送。

FTP 与 Telnet 的关系：

FTP 在数据连接中使用 Telnet 协议。这可能以两种方法实现：第一，用户 PI 或服务器 PI 在它们的实现过程中应用 Telnet 协议。第二，用户 PI 或服务器 PI 可能用系统中已经存在的 Telnet 模块。

第二种方法使用更简单，达到了代码共享，模块化编程的目的。比第一种方法更独立和高效。实际上 FTP 对 Telnet 协议的依赖非常小，因此第一种方法也不一定会引入大量的代码。

3. 数据传输功能

文件只通过数据连接传输。控制连接用来发送操作命令以及相应的命令响应（参见 FTP 响应一章）。一些命令与主机间数据传输有关。这些数据传输命令包括：指定数据位怎样被传输的模式（MODE）命令，以及用来定义数据表示方式的结构（STRU）类型（TYPE）命令。传送和表示基本上是独立的，但“流式”传输模式依赖于文件结构参数，而当使用“压缩”传送模式时，填充字节的表示依赖于表示类型。

3.1. 数据表示和存储

数据由发送端主机存储设备传输到接收端主机的存储设备上。由于两个系统的数据存储形式不同，经常需要将数据转换形式。例如，NVT-ASCII 在不同的系统中有不同的存储表示。DEC TOP-20 一般用 5 个 7 位的 ASCII 字符存储 NVT-ASCII，左对齐成 36 位的字。IBM Mainframe 用 8 位 EBCDIC 编码存储 NVT-ASCII。Multics 将 NVT-ASCII 存储成 4 个 9 位字符组成的字。当在不同的系统中传输字符时理应将其转换成标准的 NVT-ASCII 表示。发送和接收端则应相应地在标准表示法和内部表示法间转换。

当传输二进制数据时表示法的另一个问题就是不同主机有不同的字长度。并不总是明确发送端怎样发送数据以及接收端怎样接收数据。例如，当从一个 32 位字长的系统传输 32 位字节到一个 36 位字长的系统时，应该（为了高效和实用）在后一个系统中将 32 位字节在 36 位字中右对齐。无论哪种情况，用户都应该可以选择数据表示形式和传输功能。应该注意 FTP 提供了非常有限的数据表示形式。传输这些表示形式之外的数据时用户应该自行转换。

3.1.1. 数据类型

当用户指定一个表示类型时，FTP 为我们管理数据表示。这种类型可能隐含的（ASCII/EBCDIC）或显式的（本地字节）为解释器作为“逻辑字节长度”定义字节长度。注意这并不是用于在数据连接传输时的字节长度，被称为“传输字节长度”，两种长度不能互相冲突。例如，NVT-ASCII 具有 8 位的逻辑字节长度。如果类型是本地字节，那么 TYPE 命令必须有第二个参数指定逻辑字节长度。

传输字节长度始终是 8 位。

3.1.1.1. ASCII 类型

这是缺省类型，必须被所有 FTP 实现支持。主要用来传输文本文件，除非主机双方认为 EBCDIC 类型更方便。

发送方将内部字符表示转换为标准的 8 位 NVT-ASCII 表示（参见 Telnet 协议）。接收方将标准格式数据转换为它自己的内部格式。

NVT 规定，<CRLF>序列用来表示文本一行的结尾。（参见本章末文件结构中有关数据表示和存储的讨论）用标准 NVT-ASCII 表示意味着数据必须解释为 8 位字节。

用于 ASCII 和 EBCDIC 格式参数将在下面讨论。

3.1.1.2. EBCDIC 类型

这种类型用来在使用 EBCDIC 编码的主机间高效地传输。

传输时，数制被表示为 8 位的 EBCDIC 字符。EBCDIC 与 ASCII 类型的区别仅仅是字符编码的不同。

行尾（End-of-Line）很少用在 EBCDIC 类型中表示结构，必要时应该用<NL>。

3.1.1.3. 图像类型

数据以 8 位连续字节流传输。接收端必须将数据存储为连续位。存储系统结构可能要将文件（或对于记录结构文件来说，每个记录）填充到合适的边界（字节、字或块）。填充的字节必须为 0，并追加到文件末尾（或每个记录末尾）。必须有方法来指出填充字节，当取得文件时，以将填充字节剔除。填充转换方法应当公开，使用户可以方便的处理文件。

图像类型目的是为了高效地存储和检索文件，以及传输二进制文件。建议所有的 FTP 实现都应该支持这个类型。

3.1.1.4. 本地类型

数据以参数 Byte size 指定的逻辑字节长度传输。字节长度值必须是十进制整数，并且没有缺省值。逻辑字节长度不一定要和传输字节长度一样。如果字节长度不同，那么逻辑字节将忽略传输字节边界连续打包，并在最后做必要的填充。

当数据到达接收端主机时，将以独立的方式被转换为特定主机的逻辑字节长度。这个转换过程必须是可逆的（就是说，用同样的参数会产生同样的文件）并且应该被 FTP 实现者公开。

例如，用户发送 36 位浮点数到一个 32 位字长的主机时可以以本地字节长度 36 来发送。接收端主机将存储逻辑字节，以方便操作。在这个例子中，将 36 位的逻辑字节放入 64 位双字中将满足需要。

另一个例子中，一对用 36 位字长的主机间用“TYPE L 36”传送数据。数据将用 8 位传输字节包装，因此 9 个传输字节代表两个主机字。

3.1.1.5. 格式控制

ASCII 和 EBCDIC 类型也支持第二个可选的参数。这代表了一种纵向的文件格式控制。以下数据表示类型在 FTP 中定义：

一个被传输到主机的字符文件可能具有以下三个目的之一：为了打印；为了存储用来以后重现；为了处理。如果文件传送是为了打印，接收主机必须知道垂直控制是如何被表示的。第二种目的中，可能需要在主机中存储为一个文件，日后将其重现为一样的格式。最后一种目的中，必须保证将文件从一主机传输到另一主机并在第二台主机上处理文件并不带来麻烦。单独的 ASCII 或 EBCDIC 格式不能满足所有这些条件。因此，这些类型具有第二个参数指定以下三种格式之一：

3.1.1.5.1. 非打印（NON PRINT）

如果第二个格式参数被省略，这将是缺省格式。非打印格式必须被所有 FTP 实现支持。

文件不要包含垂直格式信息。如果文件被送到打印过程，那么打印过程将假定使用标准的间距和页边距值。

一般来说，这个格式被用作处理或存储。

3.1.1.5.2. TELNET 格式控制

文件包含 ASCII/EBCDIC 垂直格式控制（也就是，<CR>,<LF>,<NL>,<VT>,<FF>），打印过程将适当的解释这些控制符。<CRLF>表示行末。

3.1.1.5.3. CARRIAGE CONTROL(ASA)

文件包含 ASA（FORTRAN）垂直格式控制字符。（参见 RFC 740 附录 C;《Communications of the ACM》7 卷，10 章，606 页，1964 十月）ASA 标准规定，每行或记录的头一个字符不用来打印。这个字符用来决定本行或记录打印机的垂直走纸量。

ASA 标准指定如下控制字符：

字符 垂直间距

空 走纸一行

0 走纸两行

1 走纸到下页顶

＋ 不移动，也就是覆盖打印

打印机过程必须有方法区分结构体的结束。如果文件具有记录结构（后面将介绍）就没有问题；记录会在传输与存储中显式的标记。如果文件没有记录结构，将以<CRLF>行尾标记作为打印时行的分隔，但这些格式符会被 ASA 控制符覆盖。

3.1.2. 数据结构

由于表示类型的不同，FTP 允许文件具有指定的结构。FTP 中定义了以下三种文件结构：

文件结构，内部没有结构，文件被视为连续的数据字节流记录结构，文件由连续的记录组成页结构，文件由独立的具有索引的页组成。如果没有使用结构命令（STRU），文件结构是缺省值。但在所有的 FTP 实现中，文件和记录结构必须用在“文本”文件（就是说，带有 TYPE ASCII 或 EBCDIC 的文件）里。文件结构将影响文件的传输模式（参见传输模式一章）以及文件的表示和存储。

文件的“自然”结构取决于存储文件的的主机。源代码文件在 IBM Mainframe 上以固定长度的记录存储，而在 DEC TOPS-20 以用类似于<CRLF>的行分隔符行分开的字符流存储。如果文件在这两种站点间传输，就必须要让其中的一个站点知道另一个站点的文件结构。

在基于文件结构的主机和基于记录结构的主机间传输文件时可能会出现问题。如果文件是基于记录结构传输到基于文件结构的主机，应该在内部将记录结构转换为文件结构。显然这种转换应该能够可逆，以便可以再转回记录结构。

在将文件从基于文件结构的主机传输到基于记录结构的主机时，存在如何将文件切分成记录的问题。如果必须切分一个文本文件，那么 FTP 实现应该使用行末符，ASCII 中是<CRLF>，EBCDIC 中是<NL>，作为分隔。

3.1.2.1. 文件结构

如果没有使用结构命令（STRU），文件结构就默认使用。

在文件结构中没有内部结构，文件被当作连续的字节流。

3.1.2.2. 记录结构

在所有的 FTP 实现中，必须支持“文本”文件（就是，使用 TYPE ASCII 或 EBCDIC）的记录结构。

在记录结构文件中，文件由连续的记录组成。

3.1.2.3. 页结构

为了传输不连续的文件，FTP 定义了页结构。一般说的“随机存取文件”或“多穴文件”属于这个类型。对于这些文件，一般有另外的对应整个文件信息（例如，文件描述符），或者对应文件部分信息（例如，页存取控制），或两者都有。在 FTP 中，文件的部分称为页。

为了提供不同页大小以及相关信息，每页传输时将额外包括一个页头。页头有如下定义的域：

头长度

包括这个字节在内的头逻辑长度。最小头长度是 4。

页索引

文件区域的逻辑页号。并不是传输的序列号，而是标识本页的索引号。

数据长度

页中数据的逻辑字节数。最小数据长度为 0。

页类型

标示了页的类型。定义了如下的类型：

0 = 最末页

用来标示页结构传输结束。页头长度必须为 4，数据长度必须为 0。

1 = 单独页

对于没有页相关控制信息的单独页来说这是普通的类型。页头长度必须为 4。

2 = 描述页

这个类型用来传输整个文件的描述信息。

3 = 存取控制页

此类型包括一个额外的指定页存取信息的头域。头长度必须为 5。

可选域

其他的头域可能用来提供每页的控制信息，例如，每页的存取控制。

所有域都是一个逻辑字节。逻辑字节长度由 TYPE 命令指定。参见附录 I 中的更详细信息以及一个页结构的例子。

关于参数需要注意的一点：必须用相同参数续传相同的文件。相应的，FTP 实现在用相同参数续传文件时要保证传输的文件与原始文件相同。

3.2. 建立数据连接

传输数据的过程包括在指定端口建立数据连接选择传输参数。用户和服务器 DTP 都有缺省的端口号。用户过程缺省的数据端口与控制连接端口相同（也就是，端口 U)。服务器过程的默认端口与控制连接的端口相邻（也就是 L-1)。

传输字节长度是 8 位字节长。这个字节长度只与实际传输数据有关；而与主机文件系统的数据表示无关。

被动数据传输过程（可能是用户 DTP 或另一服务器 DTP）应该在发送 FTP 请求命令之前“监听”在数据端口。FTP 请求命令决定了数据传输方向。服务器在接到传输请求后将建立到指定端口的连接。当连接建立后，数据将在两端 DTP 间传输，同时服务器 PI 向用户 PI 发送确认回复。

每个 FTP 实现必须支持使用缺省的数据端口，只有用户 PI 可以使用变化的非缺省端口。

用户可能会用 PORT 命令指定一个其他的数据端口。用户可能想将文件下载到 TAC 行式打印机或者从第三方主机下载。后种情况下，用户 PI 同时建立到两服务器 PI 的控制连接。一个服务器（用 FTP 命令）等待连接，另一个服务器建立连接。用户 PI 给一个服务器 PI 发送 PORT 命令指示另一服务器的数据端口。最后，向两端发送合适的传输命令。用户控制端与服务器间传送的详细命令以及回复顺序定义在 FTP 响应一章。

一般来说，维护数据连接是服务器的责任，包括连接的建立与关闭。例外的情况是当用户 DTP 在传输模式下发送数据时需要关闭连接表示文件结束。服务器必须在以下条件下关闭数据连接：

1. 服务器在传输模式下完成数据传输，需要关闭连接，表示文件结束。
2. 服务器收到用户发来的 ABORT 命令。
3. 用户用命令改变了端口设定。
4. 控制连接合法地或由于其他原因关闭。
5. 发生了不可挽回的错误。

其他情况下是否关闭连接是服务器可选择的，这种情况下服务器必须用 250 或 226 号响应通知用户过程。

3.3. 数据连接管理

缺省数据连接端口：所有 FTP 实现必须支持使用缺省数据连接端口，只有用户 PI 可能使用非缺省端口。

协商非缺省端口：用户 PI 可能使用 PORT 命令指定非缺省用户端口。用户 PI 可能要求服务器端用 PASV 命令指定非缺省端口。连接用一对地址指定，上面两种动作之一都会得到一个不同的连接，仍然允许同时使用两个命令在两端指定新的端口。

数据连接复用：当使用流模式传输数据时，在文件传输结束后必须关闭连接。如果有多个文件传输时可能带来的问题是 TCP 为了保证传输可靠要保持连接记录一段时间。因此不能马上重新连接。

有两种解决方案。第一种是协商一个非缺省端口。第二种是使用另一种传输模式。

对于传输模式，流式传输模式有天生的不可靠性，不能确定连接是否过早的关闭。其他的传输模式（块，压缩）不用关闭连接来指示文件结束。他们使用 FTP 编码来确定文件结束。因此使用这些模式可以在多文件传输时保持使用同一个数据连接。

3.4. 传输模式

传输数据时下一个要考虑的问题是选择合适的传输模式。有三种传输模式：一个对数据格式化，并允许重新开始过程；一个压缩数据提供高效传输；一个不加修改的传输数据。最后一种模式与结构属性配合决定处理过程。在压缩模式中，表示类型决定填充字节。

所有的数据传输必须显式的或隐式的用关闭数据连接来指示文件结束（EOF）。对记录结构的文件，所有的记录结束标志（EOR）都是显式的，包括最后一个记录。对于使用文件结构的传输，使用“末页”的页类型。

注意：在本章其他部分，除非明确指出，字节都表示“传输字节”。

为了使传输标准化，发送主机将根据传输模式和文件结构把行尾或记录尾转换成传输时的格式，接收主机将进行相反的转换。IBM Mainframe 的记录记数域可能不会被另一台主机识别，因此在流模式中记录尾信息可能以两字节的控制码来传输，在块模式或压缩模式中作为标志位。没有记录结构的 ASCII 或 EBCDIC 文件中的行尾应该分别表示为<CRLF>或<NL>。这些转换工作在某些系统中可能意味着额外的工作，同样的系统在传输非记录结构文本文件时可能希望用流模式直接传输二进制流。

FTP 定义了如下传输模式：

3.4.1. 流模式

数据以字节流传输。对表示类型没有限制；可以使用记录结构。

在记录结构文件中，EOR 和 EOF 将分别用两个字节的控制码表示。第一个字节都是同样的 escape 字符。第二个字节中，EOR 将低位置一，其他位置零；EOF 则是将第二低位置一；也就是这个字节对于 EOR 来说是 1，对于 EOF 来说是 2。EOR 和 EOF 可能在传输结束时通过使最低两置一来同时指定（就是值 3）。如果想发送 escape 字符，要在第二个字节再重复一次。

如果结构是文件结构，则使用关闭主机连接来指示 EOF，传输的所有数据字节就是原始字节。

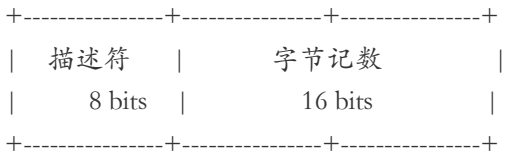
3.4.2. 块模式

文件以连续的带有数据头的数据块来传输。数据头包括一个计数域和描述码。计数域指示了数据块整个长度，由此可以算出下一数据块的开始位置（没有填充位）。描述码定义了：

文件最后一块（EOF），记录最后块（EOR），重开始标记（参见错误恢复和重开始章）或者怀疑数据（也就是被怀疑在传输中可能不可靠的数据）。最后的描述符不是FTP 错误控制的一部分。它用来在站点间交换指定类型的数据（比如地震或天气数据）而且简略本地错误（比如磁带读错误）。记录结构可以在这种模式下使用，而且可以用任何表示类型。

头包括 3 个字节。在这 24 位的头信息中，低 16 位表示字节记数，高 8 位表示描述符。

块头



描述符字节由各个标志位组成。指定了 4 个描述码，每一个描述码为描述符的十进制值。

描述码　　意义

128 数据块结束是 EOR

64 数据块结束是 EOF

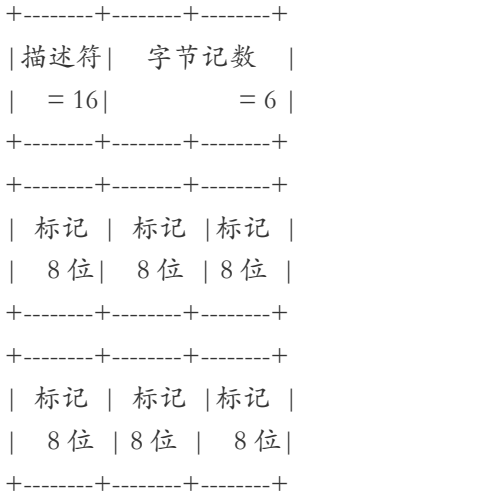
32 怀疑数据块有错

16 数据块是重开始标志

通过对不同的标志位置一，每个数据块可以使用不同的描述符组合。

重开始标志是在数据流中的 8 位整数，表示在控制连接中使用的可打印字符（比如，缺省的 NVT-ASCII）。在重开始标志中不能使用<SP>（空格）

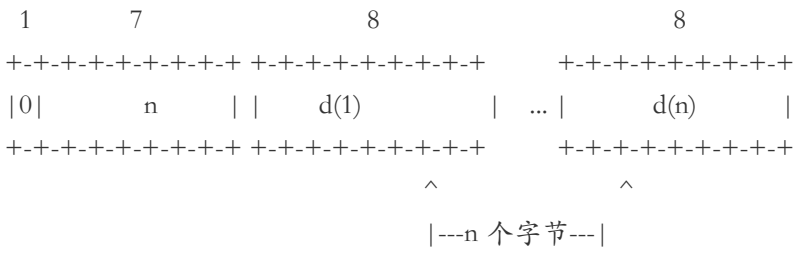
例如，要传输 6 个字符标记，应该按如下发送：



3.4.3. 压缩模式

此模式下，有三种信息要发送：常规数据，以字节串发送；压缩数据，包括复本或填充；控制信息，以两字节的转义字符传送。如果发送 n>0（最多 127）个字节的常规数据，这 n 个字节前要有一头字节，这字节的最高位为 0，其余 7 位代表数 n.

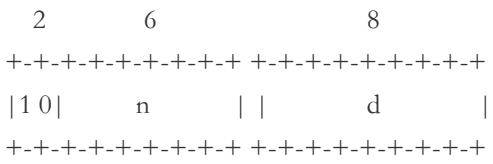
字节串：



n 字节的字节串 d(1),...,d(n)数 n 必须为正。

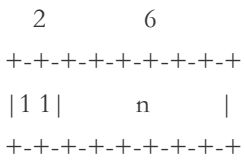
为了压缩 n 字节的复本，下面两个字节要发送：

复制字节：



一串长度 n 的填充字节可以压缩成一个字节，填充字节与表示类型有关。如果类型是 ASCII 或 EBCDIC，填充字节是<SP>（空格，ASCII 码是 32，EBCDIC 码是 64）。如果类型是图像或本地字节，填充字节为 0。

填充字节：



转义序列由两个字节组成，第一个字节为转义字节（全 0）第二个字节包括块模式中定义的描述码。这里的描述码与块模式中的描述码意义相同，并对后面的字节串有效。

压缩模式适用于在传输大数据时以较小的 CPU 代价换来一定的网络带宽。最适合用在减少 RJE 主机产生的打印文件大小。

3.5. 错误恢复和重开始

这里并没有提供是否在传输中存在丢失字节或者数据包混乱的方法。这个级别的错误由 TCP 控制。但必须提供一个重开始的方法来应对系统错误（包括主机、FTP 过程、或网络的失败）。

重开始过程只定义在块模式和压缩模式下。它要求数据发送者在数据流中插入一个特殊的标记。这个标记信息只对发送方有意义，但必须由缺省或协商的控制连接语言（ASCII 或

EBCDIC）中的可打印字符组成。标记要表示一个位记数，一个记录记数，或者可以表示数据检查点的信息。数据的接收方，如果要实现重开始过程，将用此标记指定数据位置，并将此信息返回给用户。

系统失败的情况下，用户可以用标记的位置信息重新开始传送过程。下面的例子演示了重开始过程的使用。

数据的发送方在数据流中的适当的位置插入了一个标记块。接收主机在它的文件系统中标记对应的数据点，并直接或用 110 号控制连接响应（取决于发送方是谁）向用户传达最后的标记信息。在系统失败时，用户或控制过程使用重开始命令并以标记信息为其参数来重开始传输。重开始命令通过控制连接传输，后面跟上系统失败时正在执行的命令（比如 RETR, STOR 或 LIST）。

4. 文件传送功能

从用户 PI 到服务器 PI 的传输通道是通过一个从用户到标准服务器端口的 TCP 连接建立的。用户 PI 负责发送 FTP 命令并解析接收到的响应；服务器 PI 解析命令，发送响应以及控制 DTP 建立数据连接并传送数据。如果数据传输（被动传输过程）的另一端是用户 DTP，则用户 DTP 由用户 FTP 主机的内部协议控制；如果另一端是另一个服务器 DTP，则这个服务器 DTP 由用户 PI 通过发送命令来控制。FTP 的响应将在下一部分讨论。这部分描述的一些命令对理解可能出现的响应会有帮助。

4.1. FTP 命令

4.1.1. 访问控制命令

下面的命令表示访问控制标识符（括号中表示命令代码）

用户名 （USER）

这个命令的参数域是一个用来标识用户的 Telnet 字符串。用户识别对于服务器控制文件系统存取权限是必需的。这个命令通常是控制连接建立后从用户端发送的第一条命令(一些服务器可能需要保证这一点)。一些服务器可能还需要附加的识别信息如密码或帐号命令。为了改变控制权限和/或帐户信息，服务器可能在任何时候都允许接受一个新的 USER 命令，来更换存取权限或帐户信息。产生的效果是刷新早先登录的用户名、密码和帐户信息，并重新开始一个登录过程。所有的传输参数不发生变化，并且所有正在传输中的文件传输过程均在原来的访问控制权限下完成。

密码 （PASS）

这个命令的参数域是一个用来指定用户密码的 Telnet 字符串。这个命令必须紧跟在用户名命令之后，在某些站点上，它用来完成用户访问权限识别。因为密码信息非常敏感，一般应该使用掩码代替或者禁止回显。显然服务器没有安全的办法做到这一点，所以隐藏敏感的密码信息就成了用户 FTP 进程的责任。

帐户 （ACCT）

这个命令的参数域是一个用来识别用户帐户的 Telnet 字符串。这个命令不需要和 USER 命令相关，某些站点可能需要一个帐户用来登录，另一些站点仅用于特殊访问权限，比如存储文件。后一种情况下这个命令可能在任何时候收到。

有一些响应代码用来自动地区分这些情况：当登录过程必须要求帐户信息的时候，PASS 命令成功的响应代码是 332。相应，如果登录过程不要求帐户信息时，PASS 命令成功的响应代码是 230；如果帐户信息需要在随后的对话命令中给出，服务器应该根据是保留（等待收到 ACCT 命令）还是放弃命令来相应的返回 332 或 532。

改变工作目录 （CWD）

这个命令允许用户在不改变登录用户和帐户信息的情况下改变工作目录或数据集。传输参数保持不变。这个命令的参数是一个路径名，用来指定相应的目录或者其他系统上的文件组名。

返回上层目录 （CDUP）

这个命令是CWD命令的特例，因为在不同的操作系统下表达父目录可能有不同的语法，所以可以用这个命令简化目录树的传输实现。它的响应代码应该和 CWD 的响应代码相同。更多信息参看附录 II。

结构装备 （SMNT）

这个命令允许用户在不改变用户和帐户信息的情况下装备一个不同的文件系统数据结构。传置传输参数不会改变。它的参数是一个用来标识目录或者其他系统中依赖文件组的路径名。

重新初始化 （REIN）

此命令除允许当前正在传输过程完成外，终止一个用户，刷新所有的 I/O 和帐户信息。所有参数重设为默认值，并保持控制连接。此时等同于控制连接刚刚建立的状态。这条命令之后可能需要 USER 命令。

注销 (QUIT)

此命令终止一个用户，并且当没有文件正在传输的话，服务器将关闭控制连接。如果当前有文件正在传输，连接会保持并等待回应，之后服务器将关闭连接。如果用户进程想以不同的用户名传输文件，而不想关闭然后再重建立连接的情况下，应该使用 REIN 命令而不是 QUIT。

控制连接的意外关闭将会导致服务器产生等同于放弃（ABOR）和注销（QUIT）动作。

4.1.2. 传输参数命令

所有的数据传输参数都有默认值,只有在默认值需要改变的时候才需要用命令去指定传送数据传输参数。默认值是最后一次指定的值，或者如果未被指定，则是标准默认值。这意味着服务器必须“记住”可用的默认值。这些命令可以在 FTP 服务请求前以任何顺序执行。下面这些命令用来指定数据传输参数:

数据端口（PORT）

这个参数是用来指定数据连接时的主机数据端口。对于用户和服务器都有默认的数据端口值，并且一般情况下这条命令以及它的响应都不需要。如果使用了这条命令，那它的参数是一个 32 位的因特网主机地址和一个 16 位 TCP 端口号。地址信息被分解为每 8 位一个段，每个段都作为十进制数(用字符串表示)传送。段之间用逗号分隔，一个 PORT 命令像下面这样:

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

h1 是因特网主机地址的高 8 位。

被动 (PASV)

此命令请求服务器 DTP 在一个数据端口(不是它的默认端口)上“监听”并等待连接而不是在收到传输命令后主动发起连接。这个命令的响应包括服务器监听的地址和端口号。

表示类型（TYPE）

这个命令的参数指定在数据表示和存储部分介绍的表示类型。某些类型需要第二个参数。第一个参数用单个 Telnet 字符表示，对于 ASCII 和 EBCDIC 的第二个格式化参数也是如此;本地字节的第二个参数是一个表示字节长度的十进制整数。参数之间用<SP>(空格，ASCII 码的 32)分开。

下面的编码用来表示类型:

```
                \       /
  A - ASCII |       | N - 非打印
                |-><-| T - Telnet 格式
E - EBCDIC|       | C - Carriage Control (ASA)
                /       \
  I - 图像

  L <字节长度> - 本地字节长度
```

默认的表示类型是 ASCII 非打印。如果格式化参数首先被更改，然后单独更改第一个参数，格式化参数会变回默认的非打印。

文件结构（STRU）

这个命令的参数是单个 Telnet 字符，用来指定在数据表示和存储部分描述的文件结构。

下面编码用来表示文件结构：

F - 文件 (没有记录的结构)

R - 记录结构

P - 页结构

默认的结构是文件。

传输模式（MODE）

这个命令的参数是单个 Telnet 字符，用来指定在传输模式部分描述的数据传送传输模式。

下面的编码用来表示传送模式：

S - 流

B - 块

C - 压缩

默认的传送模式是流。

4.1.3. FTP 服务命令

FTP 服务命令定义了用户请求传送文件或者文件系统的功能。FTP 服务命令的参数一般是一个路径。路径的语法必须符合服务器站点的惯例（尽量用默认标准）和控制连接的语言习惯。建议的默认参数是使用最后一个设备，目录或文件名，或者本地用户的默认标准。除"rename from"命令后面必须紧跟"rename to"命令以及 restart 命令必须紧跟随中断服务命令(例如 STOR 或 RETR)之外，其他命令可以使用任意的顺序。服务器应当总是使用数据连接来发送服务命令响应，只有少数特定的信息响应除外。下面为 FTP 服务请求命令：

获得 (RETR)

这个命令引起服务器 DTP 传送一个由路径指定的文件拷贝到数据连接另一端的服务器或用户 DTP。服务器文件的状态和内容应该不受影响。

保存 (STOR)

这个命令引起服务器 DTP 接受经过数据连接传送的数据并将这些数据存储为服务器端的一个文件。如果在路径参数里指定的文件在服务器端已经存在，那么这个文件会被传送过来的数据覆盖。如果指定的文件不存在则会在服务器端新建一个文件。

唯一保存 (STOU)

这个命令类似于 STOR 命令，但是它会在在当前目录下创建一个名字唯一的文件。在 250 号标准响应里必须包含创建出的文件名。

追加（包括创建) (APPE)

这个命令引起服务 DTP 接受从数据连接传送过来的数据并存储在服务器端的一个文件里。如果指定的文件在服务器端已经存在，则这个数据会附加到文件的后面；否则服务器端会创建这个文件。

分配 (ALLO)

一些服务器可能要求用这个命令来保留足够的空间来容纳新文件。其参数是一个十进制整数，用来指定保留给文件存储用的字节数（用逻辑字节长度)。对于用记录或者而结构传送的文件而言，还需要有最大结构或页的大小（使用逻辑字节），这个值在这个命令的第二个参数域用十进制整数指定。第二个参数是可选的，但当它存在的时候应该用三个 Telnet 字符<SP>R<SP>和第一个参数分开。这个命令之后应该是 STOR 或者 APPE 命令。在那些不需要预先知道文件最大值的服务器上，这个命令应该被作为 NOOP(无操作)对待，在那些只对记录或页最大值感兴趣的服务器上应该忽略第一个参数。

重新开始 (REST)

这个命令的参数域指定了需要重新开始传输的文件的位置标记。这个命令不会引起文件的传输，只是忽略文件中指定标记点前的数据。

重命名开始 （RNFR）

这个命令指定了需要重新命名的文件的原始路径名。后面必须马上接着“重命名为”命令，来指定新的文件路径

重命名为 （RNTO）

这个命令为在“重命名开始”命令中指定的文件指定新的路径。这两个命令一起为文件重新命名。

放弃（ABOR）

该命令告诉服务器放弃先前的 FTP 服务命令和相关的传输的数据。放弃命令也许需要引起服务器的“特别注意”（参见 FTP 命令部分），使服务器强制识别。当前一个命令（包括数据传输）完成时，将不会产生动作。服务器不会关闭控制连接，但是数据连接必须关闭。

服务器接收这个命令时可能处在两种状态：(1)FTP 服务命令已经完成，或者(2)FTP 服务命令还在执行中。

第一种情况，服务器关闭数据连接（如果数据连接是打开的）回应 226 代码，表示放弃命令已经成功处理。

第二种情况，服务器放弃正在进行的 FTP 服务，关闭数据连接，返回 426 响应代码，表示请求服务请求异常终止。然后服务器发送 226 响应代码，表示放弃命令成功处理。

删除 (DELE)

这个命令在服务器端删除指定的文件。如果需要额外的保护（比如讯问“你丫的真的想删除么？”），应该由用户 FTP 进程提供。

删除目录（RMD）

这个命令移除指定路径下的目录(如果是绝对路径)，或者是当前工作目录的子目录(如果是相对路径）。参看附录 II

新建目录（MKD）

该命令在指定的路径下新建一个目录（如果是绝对路径），或者在当前工作目录下建子目录（如果路径是相对的）。参看附录 II

打印工作目录（PWD）

该命令返回一个当前的工作目录名。参看附录 II

列表（LIST）

该命令从服务器端发送一个列表到被动的 DTP。如果路径名指定了目录或者别的文件组，服务器应该传送指定目录下的文件列表。如果路径名指定了文件，服务器应当传送这个文件的信息。没有参数，意味着用户的当前工作目录或者缺省目录。数据通过数据连接以 ASCII 或 EBCDIC 类型传输。（用户必须确定类型是 ASCII 或者 EBCDIC）。因为不同系统间的文件信息差别很大，这个信息可能不易被程序自动使用，但可能对于用户来说是有用处的。

名字列表 （NLST）

该命令从服务器端传送目录列表到用户端。路径名应该指定一个目录名或者其他系统文件组描述符；无参数意味着当前目录。服务器只返回文件的名字组成的字节流，不包括其他的信息。数据将通过数据连接以 ASCII 或者 EBCDIC 类型传输，每个路径名字符串由<CRLF>或<NL>分割。（用户仍必须保证类型使用正确）。这个命令的响应信息将可能被用于程序对文件的自动处理。例如，多线程下载的实现。

站点参数（SITE）

服务器使用这个命令，提供本系统可能对文件传输有帮助的特殊服务。在协议中它的用处不是很普遍。服务的种类和语法规约可以在 HELP SITE 命令的响应中确定。

系统（SYST）

该命令来得到服务器端操作系统的类型。响应的第一个词应该是 Assigned Numbers 文档[4]中表示操作系统名字的一个词。

状态 （STAT）

该命令应该通过控制连接以响应码的形式返回状态信息。此命令可能在文件传输过程中发出（与 Telnet IP 和同步信号一起，参见 FTP 命令道听部分），此时服务器将返回正在传输的状态。或者这个命令也可能在两个文件传输过程之间发出，这种情况下，命令可能将有一个参数域。如果参数指定了一个路径名，则命令将与列表命令类似，只是数据由控制连接传输。如果给出了部分路径，服务器可能响应指定的路径下的文件名列表或者相关属性。如果没有提供参数，将返回服务器 FTP 进程一般的状态信息，其中应该包括所有传输参数的当前值和连接的状态。

帮助（HELP）

该命令使服务器通过控制连接传送关于具体实现状态的帮助信息给用户。该命令可以有参数（例如，命令的名字）返回更加具体的信息。回应代码是 211 或者 214。建议在输入 USER 命令前允许 HELP。服务器可以用这个响应指定站点特定的参数，例如，在 HELP SITE 响应中指定。

空操作（NOOP）

该命令不应影响任何参数或者之前发送的命令。该命令不指定任何动作，只是要求服务器返回 OK 响应。

文件传输协议在控制连接上的所有通信都遵守 Telnet 协议。因为 Telnet 传输使用的语言可能是一个可协商的选项，下两部分提到的所有参考信息将使用“Telnet 语言”和相应的“Telnet 行尾符”。当然可以将这些转换成 NVT-ASCII 和<CRLF>。没有其它的 Telnet 协议规范被引用。

FTP 命令是以"Telnet 行末符"结尾的"Telnet 字符串"。命令如果带有参数，那么命令代码本身是以<SP>（空格）结尾的文字字符，或者当没有参数时以 Telnet 行末符结尾。命令代码和命令的语义在本章描述；详细的命令语法在命令一章描述，响应序列在命令和响应一章中描述，命令用法的情景演示说明在典型 FTP 情景一章中给出。

FTP 命令分为访问控制命令、数据传输参数命令、FTP 服务请求命令三种。某些命令（例如，ABOR，STAT，QUIT）可以在数据传输过程中，通过控制连接发送。一些服务器可能

不能同时监控控制连接和数据连接，此时就要发出一些特殊的动作来引起服务器的注意。下面的指令格式是试验性建议：

1.用户系统在 Telnet 流中插入 Telnet"中断过程"（Interrupt Process-IP）信号

2.用户系统发出 Telnet “同步”（Synch）信号

3.用户系统在 Telnet 流中插入命令（例如，ABOR）

4.服务器 PI，在接收到"IP"后，扫描 telnet 流，寻找 FTP 命令

（对于其它服务器，这些操作可能没有必要，但并不会引起意外的后果。）

4.2. FTP 响应

文件传输协议命令的响应，用来确保在文件传输过程中的请求和正在执行的动作保持一致，保证用户程序总是可以得到服务器的状态信息。每一个命令必须产生至少一个响应，也可能产生多个响应；多重的响应必须是可以简单区分的。另外，一些命令是有一定顺序的组合。比如 USER、PASS 和 ACCT，或者 RNFR 和 RNTO。此时的响应表示一种中间状态，说明前面的命令是成功的。顺序组合中出现任何错误都会导致需要从头开始整个命令序列。

命令-响应序列的细节，将由下面一组状态图表明确表示。

FTP 响应由 3 位数字组成（以 3 个数字字符传递）后面跟着一些文本。数字用来自动的判断当前的状态，文本内容提供给人类用户。三位数字应该包含足够的信息，使用户 PI 不需要检查文本内容，而将其忽略或返回给用户。文本内容可能是与特定服务器相关的，所以每一个响应的文本内容很可能不同。

响应包含的 3 位数字，后面跟着空格<SP>，然后是一行文本（已指定一行最大的长度），以 Telnet 行末符结尾。有可能出现文本长度大于一行的情况。在这种情况下，文本全文必须在两端加以标识，使用户进程知道什么时候应该停止读取响应（也就是，停止从控制连接读取输入），去做别的事情。这要求第一行文本需要一种特殊的格式，来标识传来的文本内容有多行，并在文本最后一行指明这是最后。必需要包含适当的响应代码，以指明当前文本的状态。为了满足这些功能，第一行和最后一行的代码应该是一样的。

因此，多行回应的格式是：第一行以正常的响应代码开始，后接连字符“-”（也就是那个减号）后面跟着文本。最后一行需要以相同的代码开始，后面跟空格<SP>分隔的可选文本，然后是 Telnet 行末符

例如:

123-第一行

第二行

234 以数字开始的一行

123 最后一行

用户进程只需要简单地寻找一行开始时后面跟随（空格）的相同响应代码，并忽略掉中间的文本。如果中间文本的某一行首出现了 3 位数字，服务器必须在前面填充，以避免混淆。

添加“人工的”第一行和最后一行标志的这种方案，允许使用标准系统例行程序产生响应信息（例如，产生 STAT 响应）。少数情况下，如果例行程序必须在某一行行首生成 3 位数字后跟空格，文本的每一行行首应该填充一些空文本，例如空格。

这个方案假定多行的响应不能被嵌套。

3 位数字的每一位都有特定的意义。允许用户进程将复杂的响应简化。第一位数字标识了响应是好，坏或者未完成。（参见状态图），简单的用户进程可以通过检查第一位数字，决定它的下一个动作（依计划处理，重试，放弃等等）。用户进程如果希望知道大概是发生了什么错误（比如，文件系统错误，语法错误），可以通过检查第二位数字来完成。第三位数字指示信息顺序是否有误（例如，RNTO 前没有 RNFR 命令）。

响应的第一位数字可能有以下五个值：

1yz，预备状态

请求的动作已经启动；在下一个新命令之前，期望一个回应。（用户进程在接收到完成响应前就发送另一条命令是违返协议的。但服务器 FTP 进程在处理前面命令的过程中应该将后续收到的命令存入队列。）这种类型的响应用来表明命令已被接受，对于不能同时监视数据和控制连接的用户进程来说，它可能要开始关注数据的连接了。服务器 FTP 进程最多每个命令发送一个 1yz 代码。

2yz，完成状态

请求动作被成功的完成。一个新的请求可以开始。

3yz，中间状态

命令被接受，但是请求动作暂时没有被执行，等待收到进一步的信息。用户应该发送另一个命令来指定这个信息。这个回应用在命令组合中。

4yz，暂时拒绝状态

命令没有被接受，请求动作没有发生，但是这个错误状态是暂时的，动作可以被再次请求。用户应该重新回到命令队列的开始。说明“暂时”的具体意思是很困难的，尤其在两个截然不同的站点（服务器和用户进程）间要达成解释的一致更是不易。每个 4yz 号响应可能都有一个稍不同的时间值，但总体思想都是鼓励用户进程再一次重试。判断一个响应应该属于 4yz 号还是 5yz 号的一个规则是看这个命令是否可以不加修改并在相同的用户、服务器状

态下（比如，命令使用同样的拼写使用同样的参数；用户不改变文件访问权限；服务器不产生新的实现。）再次重复。

5yz,永久拒绝状态

命令不被接受，请求动作不会发生。用户进程不能重复同样的请求（包括同样的命令顺序）。一些“永久的”错误状态可以被修正，因此人类用户也许希望控制用户进程在将来的某点上重新开始命令队列。（比如在拼写改变之后，或目录状态改变之后。）

下面为第二位数字的功能：

x0z 语法 - 这种响应指出了语法错误。给出的命令不存在、没有被实现、或多余。

x1z 信息 - 对于请求信息的响应，比如对状态或帮助的请求。

x2z 连接 - 关于控制连接和数据连接的响应。

x3z 身份验证和帐户 - 对登陆过程和帐户处理的响应。

x4z 目前还未使用。

x5z 文件系统 - 请求传输时服务器文件系统的状态或其他文件系统动作状态。

第三位数字为第二位数字指定的状态提供了更详细的意义。下面的响应列表会说明这一点。注意，每一个响应的对应文本只是推荐的，而非强制性的，可依照相应的命令而更改。另一方面，响应代码，必须严格的遵守最后部分的规范，也就是说，服务器实现不应该为与上面所描述的只有微小区别的状态发明新的代码，而应该使用已经定义的代码。

类似 TYPE 或 ALLO 这样的成功执行也不会给用户进程新信息的命令将产生 200 号响应。如果命令因为与本计算机系统无关而不必被服务器 FTP 进程支持的，例如 ALLO 在 TOPS20 站点上，应该回复一个完成状态的响应，来通知用户进程可以继续它的动作请求。202 号响应用来处理这种情况，例如，响应文本为“No storage allocation necessary”（无需分配存储）。另外，如果，如果请求了一个并没有被实现的命令，将返回 502。504，表明实现了此命令，但是请求的参数并未被实现。

4.2.1. 按功能分组的响应代码

200 Command okay. （命令 OK）

500 Syntax error, command unrecognized. （语法错误，命令不能被识别）可能包含因为命令行太长的错误。

501 Syntax error in parameters or arguments. （参数语法错误）

202 Command not implemented, superfluous at this site. （命令没有实现，对本站点冗余）

502 Command not implemented. （命令没有实现）

503 Bad sequence of commands. （命令顺序错误）

504 Command not implemented for that parameter. （没有实现这个命令参数）

110 Restart marker reply. （重新开始标记响应）对于这种情况，文本应该是明确的，无需进行特殊实现；必须形如：MARK yyyy = mmmm ； yyyy 是用户进程数据流标记，mmmm 服务器的等效标记（注意，标记间的空格和“=“）

211 System status, or system help reply. （系统状态，或者系统帮助响应。）

212 Directory status. （目录状态）

213 File status. （文件状态）

214 Help message. （帮助信息）

关于如何使用服务器，或者特殊的非标准的命令的意义。只对人类用户有用。

215 NAME system type. （系统类型名称）这里的 NAME 指在 Assigned Numbers 文档中列出的正式名称。

120 Service ready in nnn minutes. （服务将在 nnn 分钟后准备完成）

220 Service ready for new user. （接受新用户服务准备完成）

221 Service closing control connection. （服务关闭控制连接）已注消

421 Service not available, closing control connection. （服务不可用，关闭控制连接）如果服务器知道它必须关闭，应该以 421 作为任何命令的响应代码。

125 Data connection already open; transfer starting. （数据连接已打开，传输开始）

225 Data connection open; no transfer in progress. （数据连接打开，没有传输）

425 Can't open data connection. （不能打开数据连接）

226 Closing data connection. （关闭数据连接）请求文件动作成功（例如，文件传输或者放弃）

426 Connection closed; transfer aborted. （连接关闭，放弃传输）

227 Entering Passive Mode (h1,h2,h3,h4,p1,p2). （进入被动模式）

230 User logged in, proceed. (用户成功登录，继续）

530 Not logged in. （没有登录成功）

331 User name okay, need password. （用户名 OK，需要密码）

332 Need account for login. （需要帐户才能登录）

532 Need account for storing files. （需要帐户来存储文件）

150 File status okay; about to open data connection. （文件状态 OK，将打开数据连接）

250 Requested file action okay, completed. （请求文件动作 OK，完成）

257 "PATHNAME" created. （创建了“PATHNAME”）

350 Requested file action pending further information. （请求文件动作需要进一步的信息）

450 Requested file action not taken. （请求文件动作没有执行）文件不可使用（例如，文件忙）

550 Requested action not taken. (请求的动作没有执行）文件不可用（例如，没有找到文件，没有访问权限）

451 Requested action aborted. Local error in processing. （请求动作放弃，处理中发生本地错误）

551 Requested action aborted. Page type unknown. （请求动作放弃，未知的页面类型）

452 Requested action not taken. （请求动作未执行）系统存储空间不足。

552 Requested file action aborted. (请求文件动作被放弃)超出存储分配空间（当前的路径或者数据集）

553 Requested action not taken. （请求动作未获得）文件名不允许。

4.2.2. 按数字顺序排列的响应代码

110 Restart marker reply. （重新开始标记响应）对于这种情况，文本应该是明确的，无需进行特殊实现；必须形如：MARK yyyy = mmmm ； yyyy 是用户进程数据流标记，mmmm 服务器的等效标记（注意，标记间的空格和“= “）

120 Service ready in nnn minutes. （服务将在 nnn 分钟后准备完成）

125 Data connection already open; transfer starting. （数据连接已打开，传输开始）

150 File status okay; about to open data connection. （文件状态 OK，将打开数据连接）

200 Command okay. （命令 OK）

202 Command not implemented, superfluous at this site. （命令没有实现，对本站点冗余）

211 System status, or system help reply. （系统状态，或者系统帮助响应）

212 Directory status. （目录状态）

213 File status. （文件状态）

214 Help message. （帮助信息）关于如何如使用服务器，或者特殊的非标准的命令的意义。只对人类用户有用。

215 NAME system type. （系统类型名称）这里的 NAME 指在 Assigned Numbers 文档中列出的正式名称。

220 Service ready for new user. （接受新用户服务准备完成）

221 Service closing control connection. （服务关闭控制连接）已注消

225 Data connection open; no transfer in progress. （数据连接打开，没有传输）

226 Closing data connection. （关闭数据连接）请求文件动作成功（例如，文件传输或者放弃）

227 Entering Passive Mode (h1,h2,h3,h4,p1,p2). （进入被动模式）

230 User logged in, proceed. (用户成功登录，继续）

250 Requested file action okay, completed. （请求文件动作 OK，完成）

257 "PATHNAME" created. （创建了“PATHNAME”）

331 User name okay, need password. （用户名 OK，需要密码）

332 Need account for login. （需要帐户才能登录）

350 Requested file action pending further information. （请求文件动作需要进一步的信息）

421 Service not available, closing control connection. （服务不可用，关闭控制连接）如果服务器知道它必须关闭，应该以 421 作为任何命令的响应代码。

425 Can't open data connection. （不能打开数据连接）

426 Connection closed; transfer aborted. （连接关闭，放弃传输）

450 Requested file action not taken. （请求文件动作没有执行）文件不可使用（例如，文件忙）

451 Requested action aborted. Local error in processing. （请求动作放弃，处理中发生本地错误）

452 Requested action not taken. （请求动作未执行）系统存储空间不足

550 Requested action not taken. (请求的动作没有执行) 文件不可用 (例如, 没有找到文件, 没有访问权限)

501 Syntax error in parameters or arguments. （参数语法错误）

502 Command not implemented. （命令没有实现）

503 Bad sequence of commands. （命令顺序错误）

504 Command not implemented for that parameter. （没有实现这个命令参数）

530 Not logged in. （没有登录成功）

532 Need account for storing files. （需要帐户来存储文件）

550 Requested action not taken. (请求的动作没有执行) 文件不可用 (例如, 没有找到文件, 没有访问权限)

551 Requested action aborted. Page type unknown. （请求动作放弃，未知的页面类型）

552 Requested file action aborted. (请求文件动作被放弃)超出存储分配空间(当前的路径或者数据集)

553 Requested action not taken. （请求动作未获得）文件名不允许。

5. 公布的规范

5.1. 最小实现

为了让 FTP 能够不出错误的工作，服务器必需具备以下最小实现:
类型 - ASCII 非打印
模式 - 流模式
结构 - 文件结构，记录结构
命令 - USER, QUIT, PORT,TYPE, MODE, STRU，相应的默认值，RETR, STOR，NOOP.
传输参数的默认值为:
类型 - ASCII 非打印
模式 - 流模式

结构 - 文件结构
所有主机必需接受上面这些标准的默认值。

5.2. 连接

服务器 PI 在端口 L 上“监听”。用户或用户 PI 发起一个双向的控制连接。服务端和用户端进程都应该遵守 ARPA-Internet Protocol Handbook[1]中描述的 Telnet 协议。服务器没有必要一定要提供命令行编辑，但用户主机可能需要。当所有的传输和响应全部完成的时候，控制连接应该在用户的请求下由服务器关闭。

用户 DTP 必须在指定的数据端口上“监听”；这个端口可能是默认的用户端口(U)或者由 PORT 命令指定的端口。服务器端将从默认的数据端口(L-1)发起一个到用户指定数据端口的数据连接。传输方向和使用的端口由 FTP 服务命令指定。

注意，所有的 FTP 实现必须支持使用默认的端口进行数据传送，并且只有用户 PI 可以发起使用非默认端口。

当数据在两个服务器 A 和 B（参考图 2）间传输时,由用户 PI，C，建立到两个服务器的控制连接。然后，C 发送 PASV 命令给服务器 A,让服务器 A 在数据端口上“监听”,而不是在收到传输服务命令后主动发起连接。服务器 A 对 PASV 命令的响应它包括了正在监听的主机 IP 和端口号，用户 PI 通过 PORT 命令发送服务器 A 的端口 a 到服务器 B。B 响应后，用户 PI 可能会发送相关的服务命令给服务器 A 和服务器 B。服务器 B 发起连接并开始传输。下面列出了命令和响应的顺序，其中消息是竖直方向同步，水平方向不同步的。

```
用户-PI - 服务器 A                    用户-PI - 服务器 B
-----------------                     -----------------

C->A : Connect                        C->B : Connect
C->A : PASV
A->C : 227 Entering Passive Mode. 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 : Connect to HOST-A, PORT-a
```

图 3

数据连接应该由哪个服务器关闭遵守建立数据连接一章中的描述。数据连接在数据传输完成后必须马上关闭。等到下一个传输命令是不允许的，因为用户进程要检查数据连接来确定是否要开始“监听”;(用户必须在发送传输请求前“监听”在一个已经关闭的数据端口上)。为了防止混乱，关闭数据连接后服务器发送响应 226（或者如果连接保持打开状态，发送一个"文件传送完成"的应答(250) 并且在发出一个新的传输命令之前用户 PI 应该等待其中的一个响应)。

任何时候，当用户端或者服务器端发现连接将要被另一端关闭时，应该立即读取还在连接队列中的剩余数据，然后再执行关闭。

5.3. 命令

如 FTP 命令一章描述，通过控制连接传输的命令是 Telnet 字符串。命令的功能和语义在访问控制命令，传输参数命令，FTP 服务命令，和杂项命令部分描述。命令语法在这说明。

命令由一个命令代码开头，紧跟一个参数域。命令代码由四个或少于四个的字母组成。大写或小写字符被认为等同。因此，下面这些都可以描述 RETRIEVE 命令。

RETR Retr retr ReTr rETr

这个规定也适用于参数符号，比如 A 或者 a 都表示 ASCII 类型。命令代码和参数域用一个或多个空格进行分隔。

对于 NVT-ASCII 表示，参数域由不定长度的字符串组成，并以字符序列<CRLF>结束(回车，换行)；对于其它协商的语言，可能会使用不同的行结束符。必须注意的是服务器在接收到行结束符之前不会做任何动作。

下面是基于 NVT-ASCII 的语法说明。参数域的所有字符都是 ASCII 字符，包括十进制整数也是 ASCII 字符。用方括号括起来的表示可选参数。如果这些参数没有被指定，则隐含使用默认值。

5.3.1. FTP 命令

下面是 FTP 的命令：

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 命令参数

上面参数域的语法(在可用的地方使用 BNF 表示法)是:

<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 命令和响应顺序

用户和服务器之间的通讯为交互式的对话。基于这样，用户端每发出一个 FTP 命令，服务器都要回复响应。用户端在发送下一个命令之前应该等待上一个命令的响应，以判断是

否成功。

一些命令需要用户等待第二个响应。例如，这些响应可能报告文件传输正在进行或已经完成也可能是数据连接的关闭。它们是文件传输命令的第二个响应。举例来说，这些回复可能报告一个文件传送的进度，或者文件传送结束，或者关闭数据连接。他们是文件传送命令的二级次要回复。

一组很重要的信息响应是连接问候。一般情况下，当连接完成的时候，服务器会发送220"等待输入"的响应。用户应该在发送任何命令之前等待这个问候消息。如果服务器不能立刻接受输入，会立即发送一个120"期待推迟"，当准备好后会发送220号响应。这样用户才不会因为延迟而停滞。

自然响应

有时,"系统"会自发地发送一条消息给用户(经常是所有用户)。比如，"系统将在15分钟之内关闭"。像这种从服务器发送到用户的自发信息是无法预期的。建议这类信息放在服务器具PI的队列中并在下一次响应时发送给用户IP（可能是多行回复)。

下面的表格列出来每个命令可能收到的成功或失败的响应。它们必须严格按照下面的说明：服务器可以修改响应的文字说明，但是命令响应代码的意义不能改变。

命令响应序列

这部分列出了命令响应序列；命令按功能分组。初步响应列在最前（后跟它们的成功响应)，然后是成功或失败的完成响应，最后是为后续命令准备的中间响应。列表是状态图的基础，状态图将在之后介绍。

建立连接
120
220
220
421
Login
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

Logout

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

文件动作命令

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

信息命令

SYST

215

500, 501, 502, 421

STAT

211, 212, 213

450

500, 501, 502, 421, 530

HELP

211, 214

500, 501, 502, 421

杂项命令

SITE

200

202

500, 501, 530

NOOP

200

500 421

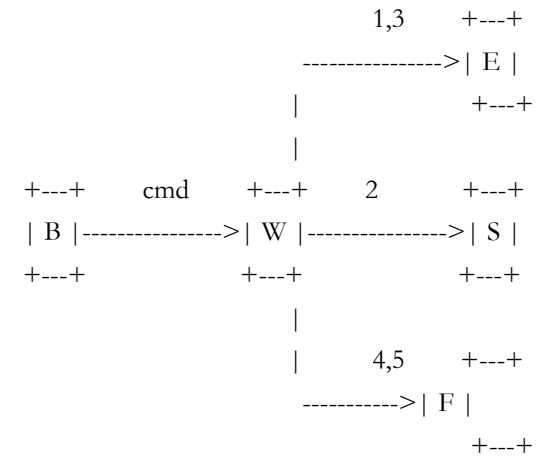
6. 状态图

在这我们介绍一个简单 FTP 实现的状态图。只使用了响应代码的第一位。每一组 FTP 命令或命令序列对应一个状态图。

命令分组是通过构造每个命令的模型，然后把结构相似的命令组合在一起分组得到的。

每个命令或者命令序列都有三种可能的结果：成功(S)，失败(F)和出错(E)。在下面的状态图中我们使用符号 B 表示"开始"，符号 W 表示“等待一个响应”。

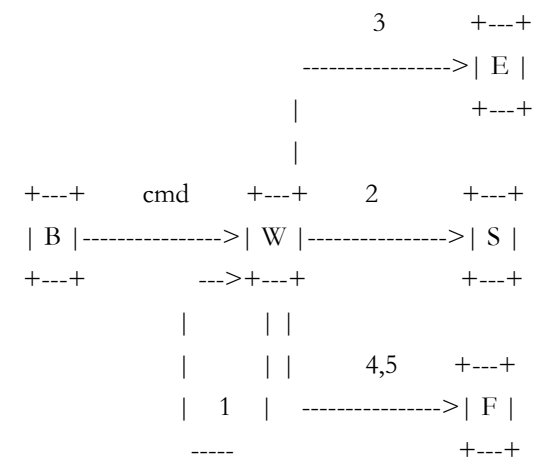
首先我们介绍这个能表现最大 FTP 命令组的图：



这个模型对应下列命令：

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

另一个很大的命令组用这个比较相似的图表示：

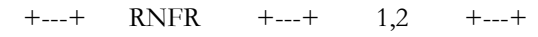


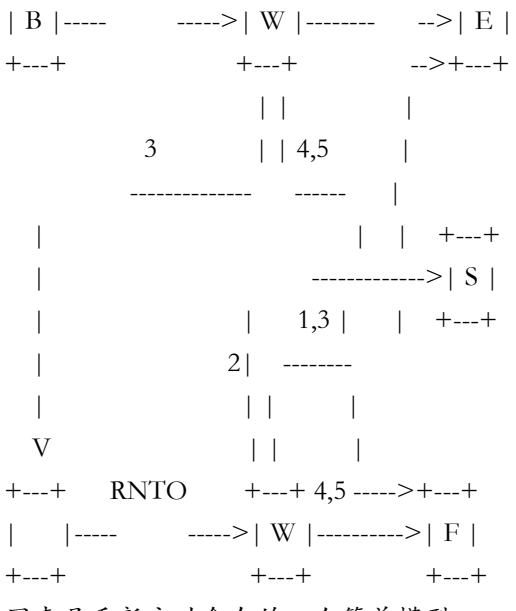
这个模型对应下列命令：

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

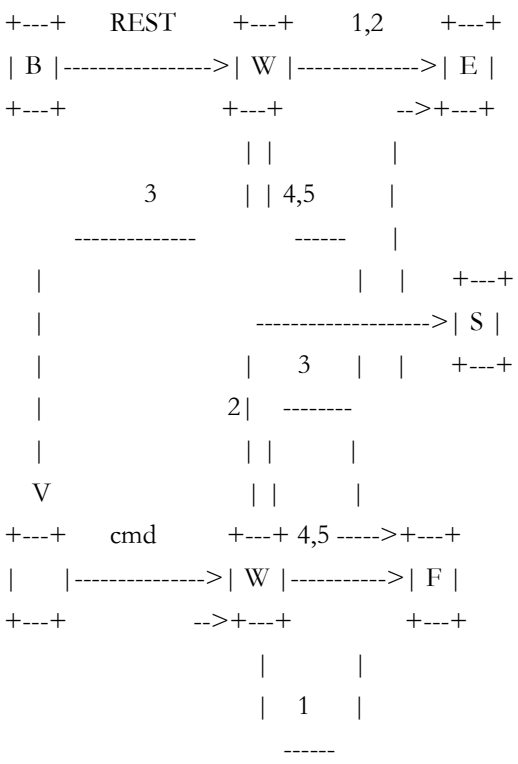
注意第二个模型也可以用来表现第一组命令，它们的区别仅仅在于第一组把 100 系列那些未预期的响应作为错误，而第二组期望(某一些可能是必须)100 系列的响应。每个命令最多允许一个 100 系列响应。

剩余的图模拟了命令序列，在这之中最简单的可能是重命名序列：





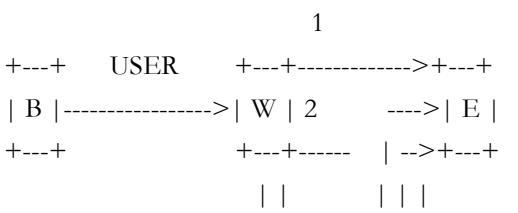
下一个图表是重新启动命令的一个简单模型：

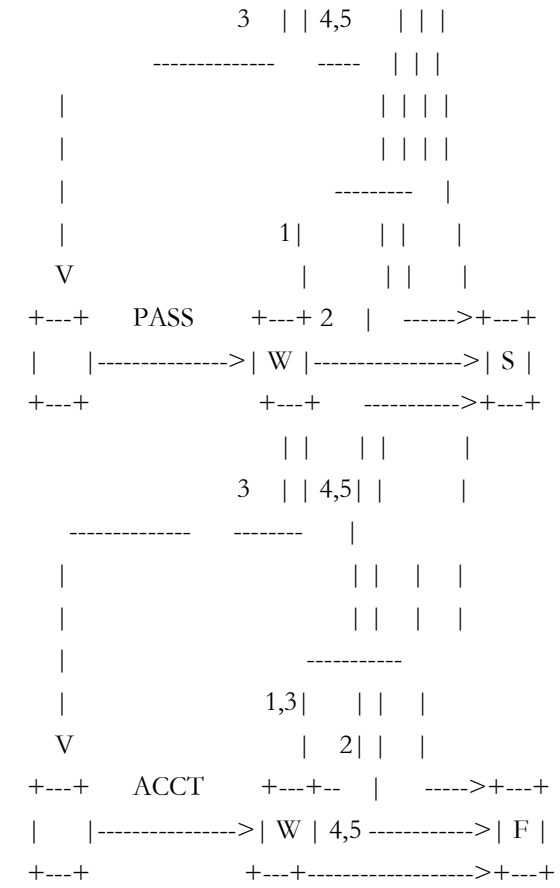


这里的“cmd”表示 APPE,STOR 或者 RETR

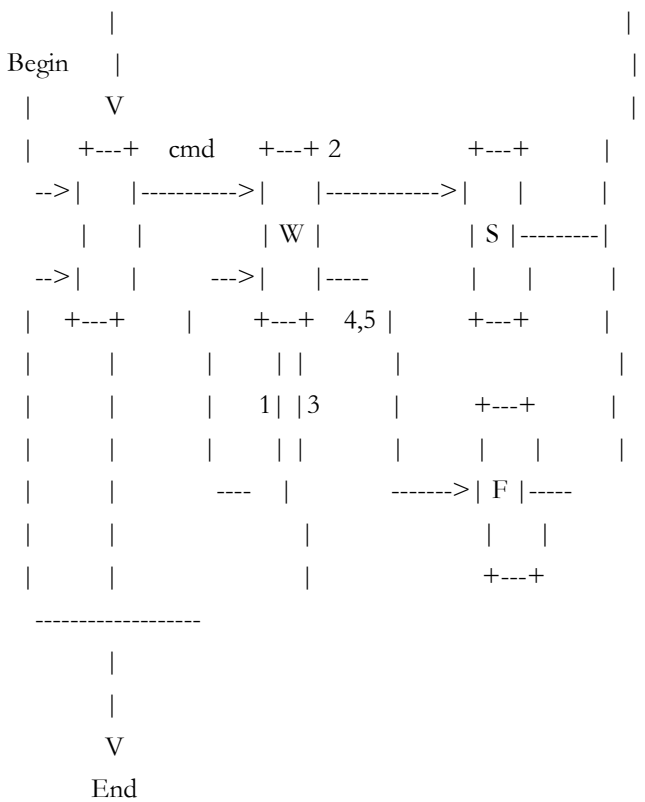
我们注意到上面这三个模型比较相似。Restart 和 Rename 的不同仅仅在第二阶段对 100 系列响应的处理，第二组期望（某一些可能必须)100 系列响应。每个命令最多允许一个 100 系列的响应。

最复杂的图表是这个登录序列：





最后，我们介绍一个用来描述命令和响应交互的一般化的图表：



7. 典型 FTP 情景

在主机 U 上的用户想要与主机 S 互传文件:

一般情况下，用户和服务器的通信会通过一个用户 FTP 进程进行中介。下面可能是一个典型的情景。用户 FTP 的提示信息在圆括号中，'---->'表示从主机 U 发送到主机 S 的命令，'<----'表示从主机 S 发送到主机 U 的响应。

LOCAL COMMANDS BY USER ACTION INVOLVED

用户输入的本地命令　相关动作

用户本地命令	涉及的动作
ftp (host) multics<CR>	连接主机 S，端口 L， 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>	用户 FTP 以 ASCII 方式打开本地文件.
(for. pathname) test.pl1<CR>	RETR test.pl1<CRLF> ---->
	<---- 150 File status okay; about to open data connection<CRLF>.
	服务器与端口 U 建立数据连接
	<---- 226 Closing data connection, file transfer successful<CRLF>.
type Image<CR>	TYPE I<CRLF> ---->
	<---- 200 Command OK<CRLF>
store (local type) image<CR>	
(local pathname) file dump<CR>	用户 FTP 以图像方式打开本地文件
(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. 连接确立

FTP 控制连接在用户进程的端口 U 服务器进程的端口 L 上通过 TCP 建立。此协议分配的服务端口是 21(八进制 25)，也就是 L=21。

附录 I - 页结构

FTP 需要支持页面结构，源自于 FTP 需要在 TOPS-20 系统间支持有效的文件传输，特

别的是 NLS 使用的文件。

TOPS-20 的文件系统以页的概念为基础。操作系统以页形式使操作文件更加有效。操作系统为文件系统提供了接口，使得很多应用程序可以将文件看作字符序列流。然而，还是有一小挫的应用程序，直接使用底层的页面结构，其中的一些产生了空穴文件(holey files)。

TOPS-20 磁盘文件由四部分组成：路径名，页表，一组页面（可能为空），一组属性。

路径名由 RETR 或者 STOR 命令指定。包括目录名，文件名，文件扩展名和代名（族名）。

页表包括相当于 2**18 个项目。每一个项目都可能为 EMPTY，或者指向一个页。如果不为空，会有指定页的存取位；不是一个文件的所有页都需要同样的存取保护。

一个页是由 512 个连续字组成，每一个字由 36 位组成。

文件的属性，在文件描述块（Descriptor Block (FDB)）内，包含创建时间，写入时间，读取时间，作者（或者是写入器?）的字节长度，文件结束指针，读写的计数，备份系统的磁带号等。

页表的项目不需要连续。也许会有空的页表槽在两个项目间（我领会的意思是两个项目间会有空的）。文件结束指针也只是个简单的数字。不一定需要实际指向文件的最后一个数据。TOPS-20 通常使用顺序 I/O，使文件结束指针指向最后写入的数据，但是其他的操作也可能会使它不这么做，如果特定的程序系统这么要求的话。

事实上，在这些特殊的情况下，在 NLS 数据文件中，就会出现空穴文件和文件结束指针不在文件的最后的现象。

TOPS-20 页面结构文件可以在如下 FTP 参数下传输：TYPE L 36, STRU P 和 MODE S（事实上，可以用任何模式）

每一个页信息有一个头字段。每个头字段为是一个逻辑字节，指定 TYPE L 36 的话就是一个 TOPS-20 字了

头字段：

字 0：头长度

头长度是 5。

字 1：页索引。

如果数据是磁盘文件页，这个字是页在文件页映射中的编号。文件里的空页，会被传输。注意，空白页不是写满零的页。

字 2：数据长度。

页面上头字段后的数据字节数量。因此传输的总长度是头部分的长度加上数据的长度。

字 3：页类型。

页类型的代码。数据页类型是 3，FDB 页类型是 2。

字 4：页访问控制。

由与页相关的访问控制位组成。（全字的数量，通过程序从网络读取到磁盘，将 SPACS 转成 AC2）（括号内的有点问题）(RE:俺也不知道是啥意思...)(This full word quantity is put into AC2 of an SPACS by the program reading from net to disk.)

头字段后，是数据长度。数据长度对数据页来说是 512 或者对 FDB 是 31。磁盘文件页结尾的零可能被删除。这样，数据长度就可能小于 512。

附录 II - 目录命令

由于 UNIX 有树形的目录结构，操作目录可以像操作普通文件那样简单。因此在这些主机的 FTP 服务器上包括目录操作命令是很有必要的。在 ARPA-Internet 上也有采用树形目录结构的（包括 TOPS-20 和 Multics），因此对这些命令的需求就是很普遍的了。

FTP 的四个目录命令：

MKD pathname

创建名为"pathname"的目录。

RMD pathname

删除名为"pathname"的目录。

PWD

显示当前工作的目录名。

CDUP

切换到父目录。

"pathname"参数应该作为当前工作目录的子目录被创建（删除），除非"pathname"字符串包含了足够的信息，为服务器指定其他的目录。举例来说，"pathname"是一个绝对的路径（UNIX 和 Multics），或者在 TOPS-20 中，路径是像"<abso.lute.path>"这样的东西。

响应代码

CDUP 命令只是 CWD 命令的一种特殊情况，使程序可以在有着不同的命名父目录方法的操作系统间，简化目录树的传输。CDUP 的响应代码是等同于 CWD 的响应代码。

RMD 的响应代码等同于删除文件的操作-DELE 的响应代码。

但对于 MKD 的响应代码则有些复杂。一个刚建好的目录，很可能是下一个 CWD 命令的对象。不幸的是，MKD 的参数不总是 CWD 的适当的参数。例如，当 TOPS-20 子目录仅仅由给出的子目录的名字创建。就是说，对于 TOPS-20 主机上的 FTP 服务器，下面的命令将失败：

MKD MYDIR

CWD MYDIR

新的目录可能仅可以由它的绝对路径来访问。例如，如果执行上面的 MKD 命令时工作在<DFRANKLIN>目录，则新的子目录只能由<DFRANKLIN.MYDIR>来访问。

甚至在 UNIX 和 Multics 主机上，MKD 的参数也不一定是适合的。如果是相对的路径名（如，相对于当前路径的路径名），用户要在相同的当前路径，以能够到达子目录。依赖于应用程序，这可能是不方便的。这样，程序不一定在任何情况下都是健壮的。

为了解决这些问题，在成功完成 MKD 命令后，服务器应当返回如下形式的一行信息：

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

也就是服务器告知用户访问创建的目录应该用什么字符串。目录的名字可以是任意的字符；里面的双引号应该使用两个双引号转意（"quote-doubling"约定）

例如，用户连接到目录 /usr/dm，创建子目录，命名路径：

CWD /usr/dm

200 directory changed to /usr/dm

MKD pathname

257 "/usr/dm/pathname" directory created

内嵌的双引号例子：

MKD foo"bar

257 "/usr/dm/foo""bar" directory created

CWD /usr/dm/foo"bar

200 directory changed to /usr/dm/foo"bar

创建已经存在的目录，将会产生错误，这种情况，服务器必须返回"禁止访问"错误响应。

CWD /usr/dm

200 directory changed to /usr/dm

MKD pathname

521-"/usr/dm/pathname" directory already exists;

521 taking no action.

MKD 的失败返回代码和它的创建代码 STOR 很类似。同样的，如果在创建子目录时，同已经存在的文件的名字相同，也会引起创建子目录失败，返回“存取失败”（这在 UNIX 上是个问题，不应该发生在 TOPS-20）

本质上，因为 PWD 命令返回的信息和 MKD 命令成功返回的信息一样，成功地 PWD 命令使用 257 号响应代码。

细节

因为这些命令对机器间传输子树很有用处，要当心 MKD 的参数，除非包含了足够多的信息来指明是其他位置，否则参数将被解释为当前目录的子目录。以 TOPS-20 为例：

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，第一个例子将在当前目录下建立子目录。相反的，第二个例子的参数，包含了足够的信息，告知 TOPS-20<unambiguous>目录是一个顶层的目录。同样要注意在第一个例子里，用户企图用名字来访问刚建立的目录，而不是 TOPS-20 返回的信息，这是违反协议的。如果已经有一个目录，这样就可能产生问题；这在一些 TOPS-20 主机的实现是不确定的。使用 RMD 命令时也有同样的问题。要点是：除非主机的路径名规则可以区分出是绝对路径还是相对路径，否则 MKD 和 RMD 命令的参数看做子目录名。MKD 命令的 257 响应必须总是包括创建目录的绝对路径。

附录 III - FTP 相关的 RFC

Bhushan, Abhay, "A File Transfer Protocol", RFC 114 (NIC 5823), MIT-Project MAC, 16 April 1971.

Harslem, Eric, and John Heafner, "Comments on RFC 114 (A File Transfer Protocol)", RFC 141

(NIC 6726), RAND, 29 April 1971.
Bhushan, Abhay, et al, "The File Transfer Protocol", RFC 172 (NIC 6794), MIT-Project MAC, 23 June 1971.
Braden, Bob, "Comments on DTP and FTP Proposals", RFC 238 (NIC 7663), UCLA/CCN, 29 September 1971.
Bhushan, Abhay, et al, "The File Transfer Protocol", RFC 265 (NIC 7813), MIT-Project MAC, 17 November 1971.
McKenzie, Alex, "A Suggested Addition to File Transfer Protocol", RFC 281 (NIC 8163), BBN, 8 December 1971.
Bhushan, Abhay, "The Use of "Set Data Type" Transaction in File Transfer Protocol", RFC 294 (NIC 8304), MIT-Project MAC, 25 January 1972.
Bhushan, Abhay, "The File Transfer Protocol", RFC 354 (NIC 10596), MIT-Project MAC, 8 July 1972.
Bhushan, Abhay, "Comments on the File Transfer Protocol (RFC 354)", RFC 385 (NIC 11357), MIT-Project MAC, 18 August 1972.
Hicks, Greg, "User FTP Documentation", RFC 412 (NIC 12404), Utah, 27 November 1972.
Bhushan, Abhay, "File Transfer Protocol (FTP) Status and Further Comments", RFC 414 (NIC 12406), MIT-Project MAC, 20 November 1972.
Braden, Bob, "Comments on File Transfer Protocol", RFC 430 (NIC 13299), UCLA/CCN, 7 February 1973.
Thomas, Bob, and Bob Clements, "FTP Server-Server Interaction", RFC 438 (NIC 13770), BBN, 15 January 1973.
Braden, Bob, "Print Files in FTP", RFC 448 (NIC 13299), UCLA/CCN, 27 February 1973.
McKenzie, Alex, "File Transfer Protocol", RFC 454 (NIC 14333), BBN, 16 February 1973.
Bressler, Bob, and Bob Thomas, "Mail Retrieval via FTP", RFC 458 (NIC 14378), BBN-NET and BBN-TENEX, 20 February 1973.
Neigus, Nancy, "File Transfer Protocol", RFC 542 (NIC 17759), BBN, 12 July 1973.
Krilanovich, Mark, and George Gregg, "Comments on the File Transfer Protocol", RFC 607 (NIC 21255), UCSB, 7 January 1974.
Pogran, Ken, and Nancy Neigus, "Response to RFC 607 - Comments on the File Transfer Protocol", RFC 614 (NIC 21530), BBN, 28 January 1974.
Krilanovich, Mark, George Gregg, Wayne Hathaway, and Jim White, "Comments on the File Transfer Protocol", RFC 624 (NIC 22054), UCSB, Ames Research Center, SRI-ARC, 28 February 1974.
Bhushan, Abhay, "FTP Comments and Response to RFC 430", RFC 463 (NIC 14573), MIT-DMCG, 21 February 1973.
Braden, Bob, "FTP Data Compression", RFC 468 (NIC 14742), UCLA/CCN, 8 March 1973.
Bhushan, Abhay, "FTP and Network Mail System", RFC 475 (NIC 14919), MIT-DMCG, 6 March 1973.
Bressler, Bob, and Bob Thomas "FTP Server-Server Interaction - II", RFC 478 (NIC 14947), BBN-NET and BBN-TENEX, 26 March 1973.
White, Jim, "Use of FTP by the NIC Journal", RFC 479 (NIC 14948), SRI-ARC, 8 March 1973.
White, Jim, "Host-Dependent FTP Parameters", RFC 480 (NIC 14949), SRI-ARC, 8 March 1973.
Padlipsky, Mike, "An FTP Command-Naming Problem", RFC 506 (NIC 16157), MIT-Multics, 26

June 1973.
Day, John, "Memo to FTP Group (Proposal for File Access Protocol)", RFC 520 (NIC 16819), Illinois, 25 June 1973.
Merryman, Robert, "The UCSD-CC Server-FTP Facility", RFC 532 (NIC 17451), UCSD-CC, 22 June 1973.
Braden, Bob, "TENEX FTP Problem", RFC 571 (NIC 18974), UCLA/CCN, 15 November 1973.
McKenzie, Alex, and Jon Postel, "Telnet and FTP Implementation - Schedule Change", RFC 593 (NIC 20615), BBN and MITRE, 29 November 1973.
Sussman, Julie, "FTP Error Code Usage for More Reliable Mail Service", RFC 630 (NIC 30237), BBN, 10 April 1974.
Postel, Jon, "Revised FTP Reply Codes", RFC 640 (NIC 30843), UCLA/NMC, 5 June 1974.
Harvey, Brian, "Leaving Well Enough Alone", RFC 686 (NIC 32481), SU-AI, 10 May 1975.
Harvey, Brian, "One More Try on the FTP", RFC 691 (NIC 32700), SU-AI, 28 May 1975.
Lieb, J., "CWD Command of FTP", RFC 697 (NIC 32963), 14 July 1975.
Harrenstien, Ken, "FTP Extension: XSEN", RFC 737 (NIC 42217), SRI-KL, 31 October 1977.
Harrenstien, Ken, "FTP Extension: XRSQ/XRCP", RFC 743 (NIC 42758), SRI-KL, 30 December 1977.
Lebling, P. David, "Survey of FTP Mail and MLFL", RFC 751, MIT, 10 December 1978.
Postel, Jon, "File Transfer Protocol Specification", RFC 765, ISI, June 1980.
Mankins, David, Dan Franklin, and Buzz Owen, "Directory Oriented FTP Commands", RFC 776, BBN, December 1980.
Padlipsky, Michael, "FTP Unique-Named Store Command", RFC 949, MITRE, July 1985.

参考文献

[1] Feinler, Elizabeth, "Internet Protocol Transition Workbook", Network Information Center, SRI International, March 1982.
[2] Postel, Jon, "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC 793, DARPA, September 1981.
[3] Postel, Jon, and Joyce Reynolds, "Telnet Protocol Specification", RFC 854, ISI, May 1983.
[4] Reynolds, Joyce, and Jon Postel, "Assigned Numbers", RFC 943, ISI, April 1985.

Network Working Group	J. Postel
RFC 编号: 959	J. Reynolds
	ISI
过时的 RFC: 765 (IEN 149)	October 1985

文件传输协议 (FTP)

备忘录状态

本备忘录描述了文件传输协议（FTP）的官方规范。对本备忘录的发布没有限制。

本规范新包括了如下可选命令:

CDUP (返回父目录), SMNT (结构装备), STOU(唯一保存),RMD (删除目录), MKD (新建目录), PWD(打印目录), and SYST (系统)。

本规范与前一个版本兼容