

Assignment 002

I .About FTP（File Transfer Protocol）

1.1 使用具有断点续传功能的 FTP 工具（如 CuteFTP、FlashFXP 等）观察 FTP 断点续传中 FTP 命令和响应的交互过程，给出截图。

FTP 断点续传是指 FTP 在上传或下载文件时，人为地将文件划分为几个部分，每个部分采用一个线程进行传输。如果碰到网络故障，重新传输时可以直接从已经上传或下载的部分后面继续传输，而不必让整个文件从头开始传输。这样的处理方式可以大大提高传输的速度，节省时间。

我使用 CuteFTP 软件来观察 FTP 断点续传的功能。下面，将从站点连接开始逐步展示 FTP 命令的响应交互过程。为了获得一个可用的 FTP 空间，我在浦东信息港（<http://www.pdxx.net>）申请了一个免费 FTP 空间。FTP 信息见图 Figure1-1。

产品控制面板

修改FTP信息 [返回控制面板首页]

修改FTP信息	
FTP服务器	114.80.210.113
FTP用户名	ComNet
FTP密码	<input type="password"/>
FTP空间大小	300M
已用空间大小	0M <input type="button" value="重新计算"/>
状态	正常

说明：

- 1、如果看不到FTP密码框里没有显示明文密码,请先修改FTP密码,再进行FTP上传。
- 2、修改FTP密码后,如果无法登陆,请按原来密码登陆,新密码稍后生效。

Figure 1-1 申请的免费 FTP 空间信息

利用申请的免费 FTP 信息在 CuteFTP 中新建一个站点并登录，其属性信息如下：

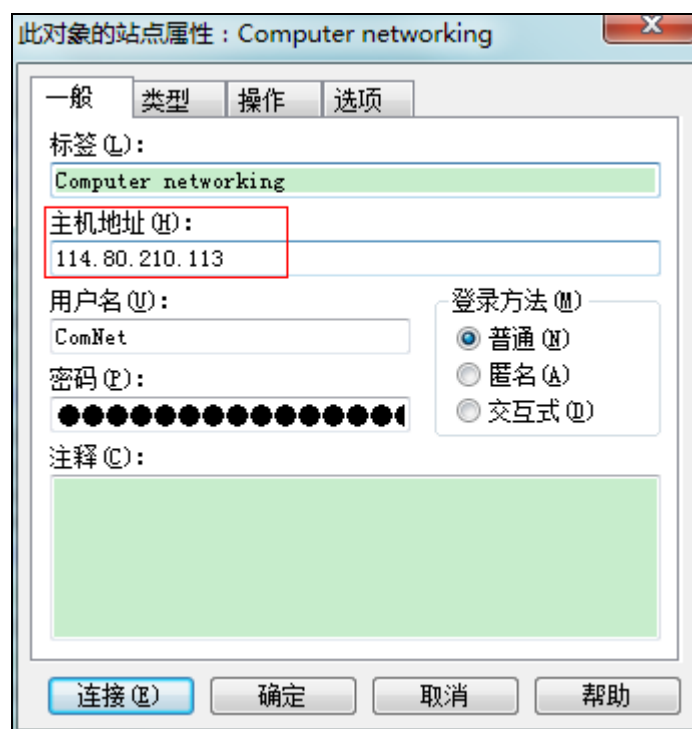


Figure 1-2 CuteFTP 中新建站点属性信息

然后，利用新建的站点演示我们需要的功能如下：

➤ 连接站点：



Figure 1-3 连接站点响应信息

通过上面的响应过程，我们可以看到软件自动以我设置的账号 ComNet 成功登录，并提供了包括主机名、服务器、用户名等在内的响应信息。同时，通过观察软件的日志窗口，我们还可以看到在登录命令响应中还提供了若干参考信息，具体截图如下：

```
命令: > [2014/3/16 13:30:21] FEAT
[2014/3/16 13:30:22] 仅参考性的信息:
211-Extension supported
CLNT
MDTM
MDTM YYYYMMDDHHMMSS[+-TZ];filename
SIZE
SITE PSWD;EXEC;SET;INDEX;ZONE;CHMOD;MSG
REST STREAM
XCRC filename;start;end
MODE Z
MLST Type*;Size*;Create;Modify*;Win32.ea*;
211 End
状态: > [2014/3/16 13:30:22] 此站点支持这些功能。
状态: > [2014/3/16 13:30:22] 此站点支持 XCRC。
状态: > [2014/3/16 13:30:22] 此站点支持 SIZE。
状态: > [2014/3/16 13:30:22] 此站点可以继续曾经中断的下载。
命令: > [2014/3/16 13:30:22] MODE Z
[2014/3/16 13:30:22] 200 MODE Z ok.
命令: > [2014/3/16 13:30:22] REST 0
[2014/3/16 13:30:22] 350 Restarting at 0. Send STORE or RETRIEVE.
命令: > [2014/3/16 13:30:22] PASV
[2014/3/16 13:30:22] 227 Entering Passive Mode (114,80,210,113,5,120)
命令: > [2014/3/16 13:30:22] LIST
状态: > [2014/3/16 13:30:22] 正在连接 FTP 数据套接字... 114.80.210.113:1400...
[2014/3/16 13:30:22] 150 Opening ASCII mode data connection for /bin/ls.
[2014/3/16 13:30:22] 226-Maximum disk quota limited to 307200 kBytes
Used disk quota 5312 kBytes, available 301888 kBytes
226 Transfer complete.
状态: > [2014/3/16 13:30:22] 已完成目录清单。
```

Figure 1-4 连接站点参考信息

可以看到，这部分参考信息主要提示了软件可以实现的功能，注意图中红色标志的部分“此站点可以继续曾经中断的下载”，这显然是我们需要的断点续传功能。

- **文件传输：**由于使用的站点不提供分部分上传，所以我先上传一份文件，然后利用分部分下载来演示需要的多线程下载功能。首先，选择一个文件上传，上传过程中使用断点续传，交互信息如下：

```
命令: > [2014/3/16 13:30:31] TYPE I
[2014/3/16 13:30:31] 200 Type set to I.
命令: > [2014/3/16 13:30:31] MODE S
[2014/3/16 13:30:32] 200 MODE S ok.
命令: > [2014/3/16 13:30:32] SIZE 十五个经典算法研究与总结、目录+索引(by_....rar
[2014/3/16 13:30:32] 213 5439488
命令: > [2014/3/16 13:30:32] MDTM 十五个经典算法研究与总结、目录+索引(by_....rar
[2014/3/16 13:30:32] 213 20140316052814
状态: > [2014/3/16 13:30:32] 远程站点上已存在一个相同的文件。
状态: > [2014/3/16 13:30:32] 正在应用智能覆盖规则：自动继续。
命令: > [2014/3/16 13:30:32] PASV
[2014/3/16 13:30:32] 227 Entering Passive Mode (114,80,210,113,5,124)
命令: > [2014/3/16 13:30:32] REST 5439488
[2014/3/16 13:30:32] 350 Restarting at 5439488. Send STORE or RETRIEVE.
命令: > [2014/3/16 13:30:32] APPE 十五个经典算法研究与总结、目录+索引(by_....rar
状态: > [2014/3/16 13:30:32] 正在连接 FTP 数据套接字... 114.80.210.113:1404...
[2014/3/16 13:30:32] 150 Opening BINARY mode data connection for 十五个经典算法研究与总结、目录+索引(by_....rar
[2014/3/16 13:32:21] 226-Maximum disk quota limited to 307200 kBytes
Used disk quota 10782 kBytes, available 296417 kBytes
226 Transfer complete.
```

Figure 1-5 上传中使用断点续传

在这个断点续传中，我们可以看到客户端首先利用 **SIZE** 命令获取先前已经

上传的大小，服务器返回该值；然后客户端在**传输 REST** 命令中，以该值作为**maker**；随后，发出 **APPE** 命令正式开始断点续传。从这个交互信息中可以清楚地看到断点续传功能的实现。

下面我以刚上传的文件（十五个经典算法研究与总结、目录+索引(by_....rar) 作为下载对象，选择**高级下载—多部分下载—高（2 部分）**展示多线程下载功能：

```
注意: > [2014/3/16 14:04:04] 已为"十五个经典算法研究与总结、目录+索引(by_....rar)" <180> 创建新的传输
状态: > [2014/3/16 14:04:04] 正在传输文件"十五个经典算法研究与总结、目录+索引(by_....rar"...
状态: > [2014/3/16 14:04:04] 正在准备多部分传输...
状态: > [2014/3/16 14:04:04] 正在传输文件"十五个经典算法研究与总结、目录+索引(by_....rar" (0:5520830)...
命令: > [2014/3/16 14:04:04] TYPE I
[2014/3/16 14:04:04] 200 Type set to I.
命令: > [2014/3/16 14:04:04] MODE S
[2014/3/16 14:04:04] 200 MODE S ok.
命令: > [2014/3/16 14:04:04] PASV
[2014/3/16 14:04:04] 227 Entering Passive Mode (114,80,210,113,7,215)
命令: > [2014/3/16 14:04:04] RETR 十五个经典算法研究与总结、目录+索引(by_....rar)
状态: > [2014/3/16 14:04:04] 正在连接FTP 数据套接字... 114.80.210.113:2007...
[2014/3/16 14:04:04] 150 Opening BINARY mode data connection for 十五个经典算法研究与总结、目录+索引(by_....rar (11041661 Bytes).
命令: > [2014/3/16 14:04:27] ABOR
[2014/3/16 14:04:27] 426-Maximum disk quota limited to 307200 kBytes
Used disk quota 10782 kBytes, available 296417 kBytes
426 Data connection closed, file transfer 十五个经典算法研究与总结、目录+索引(by_....rar aborted by client.
[2014/3/16 14:04:27] 226 ABOR command successful.
状态: > [2014/3/16 14:04:27] 传输已中止。
```

Figure 1-6 下载文件命令--响应交互信息

通过上图命令—响应交互信息，我们可以清楚看到一个文件传输命令的执行过程如下：

- 首先，在客户端发出命令后，服务器将文件目录传回本地。可以看到在传输目录时，首先连接数据套接字（socket），然后传输数据，根据服务器响应信息我们可以知道目录信息顺利完成传输；
- 然后，状态显示开始传输目标文件（十五个经典算法研究与总结、目录+索引(by_....rar)，值得注意的是，由于选择的下载方式为多部分下载，可以看到图中红色标志提示状态为“正在准备多部分下载”。此时，设置类型为 1，并使用 PASV 命令监听数据端口、等待连接；
- 接着，客户端正式发出传输文件的命令——RETR。可以看到数据套接字连接完成，开始传输文件。

文件开始传输后，在队列窗口可以看到文件的传输进度图（如图 Figure1-7 所示）。由于前面选择了两部分下载模式，所以可以看到原文件被分为两部分，且这两部分分别由一个线程控制同时下载。

最后，为了验证断点续传功能，我们在文件传输过程中进行中止命令，可以看到客户端发出 ABOR 命令，服务器响应，传输中止。

项目名称	/ 地址	<->	大小	进度	本地	远程	开始时间	完成时间	已用时间	剩余时间	速度	多部分
十五个经典算法研...	114.80...	➡	10.53 ...	12%	C:\Users\Administrator.LBDZ-20120...	/十五个经典算法研究...	2014/3/16 13:5...		0:00:05	0:01:56	647.77...	高
十五个经典算法...	114.80...	➡	5.27 MB	12%	C:\Users\Administrator.LBDZ-20120...	/十五个经典算法研究...	2014/3/16 13:5...		0:00:03	0:02:05	299.90...	高
十五个经典算法...	114.80...	➡	5.27 MB	12%	C:\Users\Administrator.LBDZ-20120...	/十五个经典算法研究...	2014/3/16 13:5...		0:00:02	0:02:19	270.55...	高

Figure 1-7 文件下载时进度图

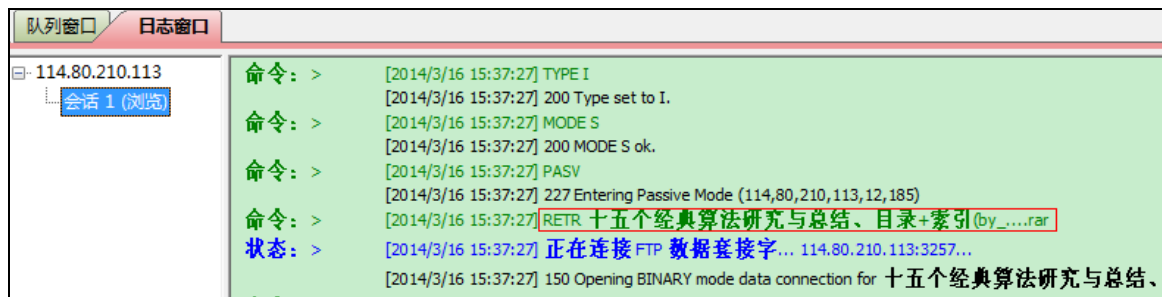
- 断点续传（下载）：

```
命令: > [2014/3/16 14:06:13] TYPE I
[2014/3/16 14:06:14] 200 Type set to I.
命令: > [2014/3/16 14:06:14] PASV
[2014/3/16 14:06:14] 227 Entering Passive Mode (114,80,210,113,8,180)
命令: > [2014/3/16 14:06:14] REST 6227390
[2014/3/16 14:06:14] 350 Restarting at 6227390. Send STORE or RETRIEVE.
命令: > [2014/3/16 14:06:14] RETR 十五个经典算法研究与总结、目录+索引(by_....rar)
状态: > [2014/3/16 14:06:14] 正在连接 FTP 数据套接字... 114.80.210.113:2228...
[2014/3/16 14:06:14] 150 Opening BINARY mode data connection for 十五个经典算法研究与总结、目录+索引(by_....rar (11041661 Bytes)).
```

Figure 1-8 中止后重新传输响应信息

由上面截图信息可以看出，当中止后再次发出文件下载（断点续传）命令时，客户端首先发出的是 REST 命令，即 restart，表示需要服务器重新开始文件传输，服务器返回信息后，客户端再次发出 RETR 命令，正式开始数据的传输的命令，随后连接数据套接字进入数据传输阶段。

另外要特别注意的一点是，由于选择了多部分下载，在下载开始的时候，我们可以看到日志窗口关于下载日志记录有两个会话，其中一个从文件头开始下载，另一个是从文件中间部分开始下载：



```
命令: > [2014/3/16 15:37:27] TYPE I
[2014/3/16 15:37:27] 200 Type set to I.
命令: > [2014/3/16 15:37:27] MODE S
[2014/3/16 15:37:27] 200 MODE S ok.
命令: > [2014/3/16 15:37:27] PASV
[2014/3/16 15:37:27] 227 Entering Passive Mode (114,80,210,113,12,185)
命令: > [2014/3/16 15:37:27] RETR 十五个经典算法研究与总结、目录+索引(by_....rar)
状态: > [2014/3/16 15:37:27] 正在连接 FTP 数据套接字... 114.80.210.113:3257...
[2014/3/16 15:37:27] 150 Opening BINARY mode data connection for 十五个经典算法研究与总结、
```

Figure 1-9 分两部分下载--会话 1（下载）日志



```
状态: > [2014/3/16 15:43:44] 此站点支持这些功能。
状态: > [2014/3/16 15:43:44] 此站点支持 XCRC。
状态: > [2014/3/16 15:43:44] 此站点支持 SIZE。
状态: > [2014/3/16 15:43:44] 此站点可以继续曾经中断的下载。
命令: > [2014/3/16 15:43:44] TYPE I
[2014/3/16 15:43:44] 200 Type set to I.
命令: > [2014/3/16 15:43:44] PASV
[2014/3/16 15:43:44] 227 Entering Passive Mode (114,80,210,113,15,150)
命令: > [2014/3/16 15:43:44] REST 5520830
[2014/3/16 15:43:44] 350 Restarting at 5520830. Send STORE or RETRIEVE.
命令: > [2014/3/16 15:43:44] RETR 十五个经典算法研究与总结、目录+索引(by_....rar)
状态: > [2014/3/16 15:43:44] 正在连接 FTP 数据套接字... 114.80.210.113:3990...
[2014/3/16 15:43:44] 150 Opening BINARY mode data connection for 十五个经典算法研究与总结、目录+索引
```

Figure 1-10 分两部分下载--会话 2（下载）日志

从 Figure1-9 和 Figure1-10 可以看出，分部分下载使用的是多线程下载，各自有一个日志记录。

1.2 通过阅读 RFC959 文档, 解释 FTP 断点续传和多线程下载实现的基本原理。

从第一题通过 CuteFTP 软件展示断点续传和多线程下载的命令-响应交互信息, 我们对其命令和执行流程都很熟悉, 从而对原理也有了一个比较模糊的了解。再来阅读 RFC959 文档就会容易很多。下面笔者分断点续传和多线程下载两部分分别简要阐述其原理。

1.2.1 断点续传基本原理

通过前面对 CuteFTP 的使用可以知道, 断点续传主要用到的一个命令是 REST, 在 RFC959 的 31 页关于 REST 命令的解释如下:

RESTART (REST)

The argument field represents the server marker at which file transfer is to be restarted. This command does not cause file transfer but skips over the file to the specified data checkpoint. This command shall be immediately followed by the appropriate FTP service command which shall cause file transfer to resume.

这段解释共三句话, 传递了两个意思: 一是 REST 所带的参数表示文件传输重新开始的一个标记; 二是 REST 命令本身不会导致文件传输, 而是需要紧接着它的 FTP 服务命令 (如 RETR、STOR、ARRE) 实现, 但是它可以使文件跳过一个特殊的检查点开始传输。

在 RFC959 第 24 页有一个小节标题为: ERROR RECOVERY AND RESTART:

3.5. ERROR RECOVERY AND RESTART

很明显这一小节就是关于断点续传机制的具体介绍。由于篇幅较长, 不再全文展示, 此处仅根据笔者自己的理解加以阐述。

由于网络故障等因素, 文件传输时常为发生中断, 为了在恢复传输时节省时间避免重复传输, FTP 引入了 restart procedure。为此, 需要数据发送者提供一个特殊标识码 (special marker code), 这个标识码应该能够使系统定义一个数据检查点。数据发送者将这个标识码随数据发送给接受者, 接受者在自己的文件系统中标志一个回应的数据点; 一旦出现故障, 在启动 restart procedure 时, 控制进程能够直接根据最后一个标识码重新开始文件传输。简言之, 这个标识码就是所谓的“断点”, 通过这样一个机制也就能实现所谓的断点续传。

下面根据上一题对 CuteFTP 软件的使用, 笔者在此结合一些网上的学习资料, 不失一般性地描述一下 FTP 进行断点续传的基本步骤。

上传:

- 利用 **SIZE 命令** 获取服务器上和本地要上传文件的同名文件大小;
- 向服务器发送 “**REST + marker**” 命令, 通知服务器接下来传送的数据是

-
- 某文件 marker 之后的，其中 maker 是前面 SIZE 命令获得的信息；
- 向服务器发送命令“**APPE + 文件名**”，通知服务器，接下来从数据通道发送给你的数据要附加到这个文件末尾；
 - 定位本地文件指针（和 FTP 上文件大小相同的位置，即 marker）；
 - 从文件指针处读取数据并发送。

下载：

- 向服务器发送“**REST + 本地文件长度**”命令，告诉服务器，客户端要断点下载了，其中本地文件长度是已经下载的长度，接下来从这之后下载；
- 向服务器发送“**RETR + 文件名**”命令，通知服务器要下载的文件名，这时服务器开始定位文件指针读取文件并发送数据。
- 定位本地文件指针（文件末尾）；
- 两端的准备工作都做完了以后，客户端创建 socket，开始传输数据。

1.2.2 多线程下载基本原理

在笔者看来，FTP 的断点续传功能和多线程下载是相辅相成的，也就是说二者的基本原理是类似的。

首先笔者根据 Figure1-9 和 Figure1-10 两个日志记录来分析。在第一题的下载中，笔者设置为**高（2 部分）**下载模式，开始下载以后日志记录有两个，分别记录两个下载的进度，仔细查看两个日志窗口，可以发现，日志会话 1 是从文件头开始下载的，也就是在客户端向服务器发出的命令直接是 **RETR + 文件名**；而日志会话 2 中显示客户端是先向服务器发送了一个 **REST + marker** 命令，然后再是 **RETR + 文件名** 开始文件传输。通过比较 Figure1-8，我们可以看到 marker 的数值大小恰是文件分为两部分的界点，也就是第一部分结束的位置，或者说第二部分开始的位置。

从以上分析并结合 1.2.1 断点续传基本原理，可以看出多线程下载的基本原理。FTP 的多线程下载主要依赖与 REST 和 RETR 两个命令。假设要下载一个大小为 1000bytes 的文件，分两个线程下载，那么 FTP 的实现如下：

第一个线程：直接利用 RETR 命令开始从 0 字节下载，当下载到 500 字节时停止下载；

第二个线程：首先用 REST+500 控制下载的开始位置，然后用 RETR 命令开始从 501 字节下载。

需要注意的是，这两个线程是同时进行传输的，从而大大提升了传输速度。

II .About HTTP (Hyper Text Transfer Protocol)

2.1 阅读 RFC 1945、2068 文档，解释 Persistent 方式的工作机制的基本原理，以及如何使用 HTTP 1.0 client 与 HTTP

1.1 server 之间实现 Persistent connection.

2.1.1 HTTP1.1 关于 persistent 的实现机制

根据 RFC2068 第 43-45 页的小节内容：8.1 Persistent Connections，可以解释 HTTP 的 persistent 工作原理。下面，就按照原文的框架、在尽可能准确的前提下笔者简要阐述其工作机制的基本原理。

首先从 persistent 方式的必要性开始。相比较于持久连接，非持久连接每选择一个 URL 都需要进行一次 TCP 的连接和中断，这无疑增加了服务器的负担，容易造成网络拥堵。再者，考虑到实际中经常出现的情况：一个客户端总是在短时间内会向同一个服务器发出多种不同的请求，采用非持久连接不但增加服务器负担，而且客户端也相对难以获得及时的响应。基于以上问题，HTTP 的持久连接是很有必要的。

相比较 HTTP 其他版本（如 HTTP1.0），HTTP1.1 一个相当重大的变化就是其默认连接方式即为持久连接，而 HTTP 1.0 使用的是非持久连接。下面我们看看 HTTP1.1 关于持久连接的协商的主要内容：

- 如果客户端的连接报头没有包含一个 close 连接字，服务器会假定客户端需要一个持久连接。如果服务器在响应后想立即关闭连接，它应该发送一个包含连接字 close 的连接报头；
- 客户端即使想要保持连接开放，也要根据服务器响应报头是否包含连接字 close 决定。如果客户端不想使用一个持久连接，需要在其发送的请求报头中包含关键字 close；
- 如果客户端和服务器任何一方的报头中包含有连接字 close，那么该连接都会在此次传输后关闭；
- 除非特别说明，否则客户端和服务器都不会考虑和一个版本低于 1.1 的 HTTP 进行连接（这也就是需要解决的第二个问题）；
- 为了保持持久连接，所有该连接上的消息都必须有一个自定义的消息长度。

下面是关于**流水线操作（Pipelining）**，HTTP1.1 持久连接提供带流水线和不带流水线两个操作，默认为带流水线工作。相关协议主要有三点：

- 一个支持持久连接的客户端可能会以流水线的方式向服务器发送请求，这就要求服务器必须以同样的顺序发回其响应。
- 如果客户端在连接建立以后立即流水发送其请求却失败了，那么它必须在第一次失败后再次尝试；但是这次尝试以后，在不能确定持久连接是否已经建立前不能再发送请求。

-
- 同时，如果客户端在流水发送自己的请求时，还没发送完成服务器已经将连接关闭，那么客户端必须再次发送。

而关于**代理服务器**：代理服务器只能分别与客户端或其他服务器（包括原始服务器和代理服务器）建立单线的持久连接，而且代理服务器不能与一个 HTTP1.0 的客户端建立持久连接。

其他还有一些关于持久连接在实际应用需要考虑的问题，在此不再赘述。以上就是 HTTP1.1 关于持久连接的相关协议内容。

简而言之，**HTTP1.1 默认的客户端和服务器之间的连接是持久连接**。如果想要实现非持久连接，可以通过客户端和服务器在相互的请求与响应报头中是否包含连接关键字 **close** 实现。

2. 1. 2 Compatibility with HTTP/1.0 Persistent Connections

下面来讨论 HTTP1.0 client 与 HTTP1.1 server 之间如何实现 Persistent connection。根据前面 HTTP1.1 持久连接协议规定的倒数第二点的提示，查看 RFC2068 的 19.7.1 小节（Compatibility with HTTP/1.0 Persistent Connections）。

通过阅读这一小节我们发现，要实现 HTTP1.0 client 和 HTTP1.1 server 之间的 persistent connection，主要是引入一个新的关键字——**Connection: Keep-Alive**，就好像 HTTP1.1 要实现非持久连接时使用的关键字 **Connection: close** 一样。

所以新的 Connection 机制是：当 HTTP1.1 想要进行非持久连接时，通过关键字 **Connection: close**；当 HTTP1.0 想要实现持久连接时，使用关键字 **Connection: Keep-Alive**。这里有三点主要的说明：

- HTTP1.0 的服务器可以相应 **Connection: Keep-Alive**，客户端也可以通过 **Connection: Keep-Alive** 和 HTTP1.0 建立持久连接；
- HTTP1.1 的服务器可以通过 **Connection: Keep-Alive** 和 HTTP1.0 客户端建立持久连接，但此时客户端不能成块传输，只能使用 **Content-Length** 来标识每条报文的界点；
- 客户端不能发送 **Connection: Keep-Alive** 给代理服务器，因为代理服务器并不遵从 HTTP1.1 的报头协议。

当然，**Connection: Keep-Alive** 在请求与响应中传输时，需要包含一个具体的 **Keep-Alive** 报头。其具体形式如下，相关其他说明不再赘述。

Keep-Alive-header = "Keep-Alive" ":" 0# keepalive-param

keepalive-param = param-name "=" value.

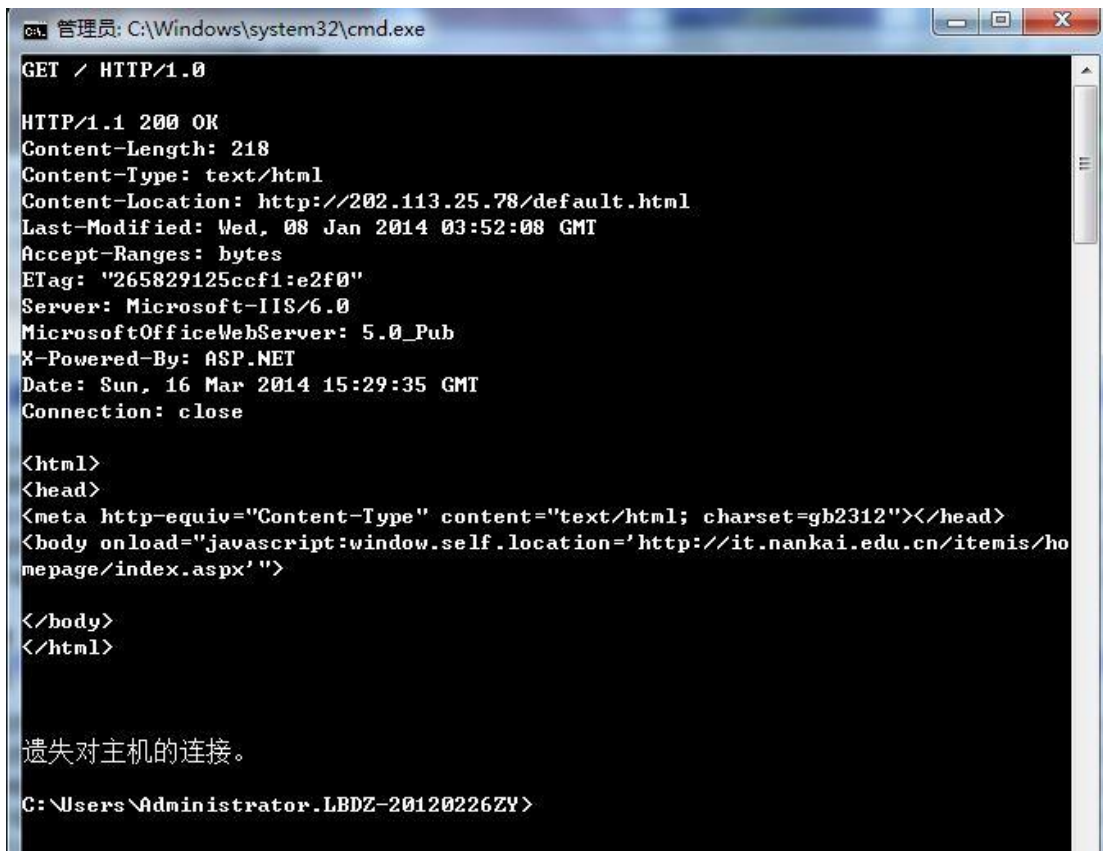
Keep-Alive 报头本身是可选的，只有在传送参数的过程中才使用，而 HTTP1.1 不会定义任何参数。此外，如果发送了 **Keep-Alive** 报头，那么响应连接字就必须一并传输；如果没有连接字，那么 **Keep-Alive** 报头就被忽略。

2.2 通过 Telnet TCP 80 端口，观察 HTTP 命令和响应的交互过程，给出截图。

➤ 利用以前学院网站(现为电光学院网站)首页做目标对象，

输入 telnet it.nankai.edu.cn 80，然后输入如下命令

GET / HTTP/1.0，两次回车，得到响应如下：



```
管理员: C:\Windows\system32\cmd.exe
GET / HTTP/1.0

HTTP/1.1 200 OK
Content-Length: 218
Content-Type: text/html
Content-Location: http://202.113.25.78/default.html
Last-Modified: Wed, 08 Jan 2014 03:52:08 GMT
Accept-Ranges: bytes
ETag: "265829125ccf1:e2f0"
Server: Microsoft-IIS/6.0
MicrosoftOfficeWebServer: 5.0_Pub
X-Powered-By: ASP.NET
Date: Sun, 16 Mar 2014 15:29:35 GMT
Connection: close

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312"></head>
<body onload="javascript:window.self.location='http://it.nankai.edu.cn/itemis/homepage/index.aspx'">

</body>
</html>

遗失对主机的连接。

C:\Users\Administrator.LBDZ-20120226ZY>
```

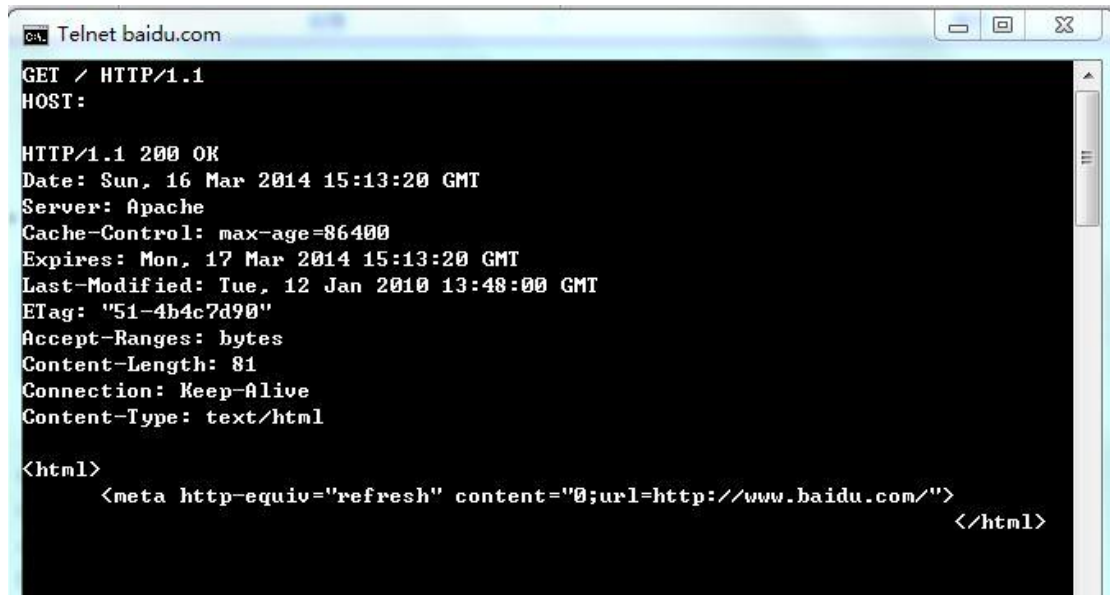
Figure 2-1 连接 it.nankai.edu.cn

➤ 用百度首页为目标，输入

telnet baidu.com 80，然后输入如下命令：

GET / HTTP/1.0

HOST:

A screenshot of a Telnet window titled 'Telnet baidu.com'. The window has a black background with white text. The text shows the commands 'GET / HTTP/1.1' and 'HOST:' followed by the server's response. The response starts with 'HTTP/1.1 200 OK' and lists various headers including Date, Server, Cache-Control, Expires, Last-Modified, ETag, Accept-Ranges, Content-Length, Connection, and Content-Type. It ends with an HTML meta tag for refreshing the page.

```
Telnet baidu.com
GET / HTTP/1.1
HOST:

HTTP/1.1 200 OK
Date: Sun, 16 Mar 2014 15:13:20 GMT
Server: Apache
Cache-Control: max-age=86400
Expires: Mon, 17 Mar 2014 15:13:20 GMT
Last-Modified: Tue, 12 Jan 2010 13:48:00 GMT
ETag: "51-4b4c7d90"
Accept-Ranges: bytes
Content-Length: 81
Connection: Keep-Alive
Content-Type: text/html

<html>
  <meta http-equiv="refresh" content="0;url=http://www.baidu.com/">
</html>
```

Figure 2-2 连接 www.baidu.com