

组织：中国互动出版网 (<http://www.china-pub.com/>)

RFC 文档中文翻译计划 (<http://www.china-pub.com/compters/emook/aboutemook.htm>)

E-mail: ouyang@china-pub.com

译者：

译文发布时间：2001-10-20

版权：本中文翻译文档版权归中国互动出版网所有。可以用于非商业用途自由转载，但必须保留本文档的翻译及版权信息。

Network Working Group
Request for Comments: 854

J. Postel
J. Reynolds
ISI
May 1983

Obsoletes: NIC 18639

TELNET 协议规范

(RFC874——TELNET PROTOCOL SPECIFICATION)

本 RFC 指定了一个 ARPA 互联网社区的标准。在 ARPA 互联网上的主机应该采纳与实现该标准。

目录

简介.....	1
一般性的考虑.....	1
网络虚终端.....	4
数据的传输.....	4
控制功能的标准表示.....	5
TELNET 中的“同步 (SYNCH)”信号.....	6
NVT 打印机和键盘.....	8
TELNET 命令结构.....	10

简介

TELNET 协议的目的是提供一个相对通用的，双向的，面向八位字节的通信机制。它的主要目标是允许界面终端设备和面向终端的过程能通过一个标准过程进行互相交互。另外，可以预想，该协议可以应用到终端到终端通信(“连接”)和过程到过程通信(分布计算)中。

一般性的考虑

一个 TELNET 连接就是一个用来传输带有 TELNET 控制信息数据的传输控制协议(TCP)的连接。

TELNET 协议的建立基于这样三个主要想法:一, 网络虚终端的概念;二, 可谈判的选项的原理;三, 对终端和过程进行均衡看待的观点。

1. 一旦一个 TELNET 连接建立后, 通信的两端被假设为在一个”网络虚拟终端”, 或者 NVT 上开始和终止操作。一个 NVT 可以被想象为一个能提供标准的, 网络范围的规范终端的中间代表者。这消除了”服务者”和”用户”之间需要保存对方终端和终端处理协定的信息的必要。所有的主机, 包括用户和服务器, 把他们本地的设备属性和协定映射为就象一个在网络上的 NVT, 而且每一方都可以假设对方也有一个类似的映射。NVT 有意地使过度受限(没有提供给主机足够的词汇来映射到他们的本地字符集)和过度包含(使用适当的终止来处罚用户)达到了平衡。

注意:”用户”机通常指那些进行连接的物理终端, ”服务器”提出指的是那些能够提供一些服务的机器。从终端到终端或过程到过程的可应用的平等性来看, ”用户”指的是初始化通信连接的机器。

2. 可谈判的选项的观点基于这样一个事实:许多主机都希望能够在 NVT 之上提供更多的服务, 而许多用户将会拥有一个更复杂的终端, 并且希望能够得到一流的, 而不是极少的一点服务。尽管相互独立, 但建立在 TELNET 协议中的是许许多多的”选项”, 这些选项将被用来认可及同”DO, DON’ T, WILL, WON’ T”结构(下面将会讨论)一起使用去允许用户和服务器同意在他们的 TELNET 连接上使用更精致的(或者可能是完全不同的)协议集合。这些选项包括改变字符集, 回显, 等等。

建立选项使用的基本策略, 是让每一方(或双方)初始化一个使一些选项有效的请求, 另一方可以接受或拒绝该请求。如果该请求被接受了, 选项立即生效;如果该请求被拒绝, 连接的另一端仍然保留 NVT 的特性。很显然, 一方经常可以通过拒绝来使能, 而从来不能通过拒绝来取消一些选项, 因为这些选项是双方为了支持 NVT 而准备的。

我们已经建立了一套谈判选项的规则, 使得双方在同时请求一个相同选项的时候, 每一方都可以把对方的请求当作对自己的请求的肯定回应。

3. 谈判句法的对称性可能会导致无穷尽的应答循环--每一方都把对方发送过来的命令当作必须回答的请求而不是对方的应答。为防止这种循环, 可以应用下面这些规则:

a. 一方只能请求改变选项的状态。也就是一方不能只发送宣布它所使用的模式的请求。

b. 如果一方所接收到的请求是要求它进入当前它所在的状态, 那么该请求将不会被应答。这种不应答对防止无穷尽的循环是非常重要的。对于那些改变模式的请求, 都需要一个应答--尽管该模式不一定改变。

c. 无论何时, 只要一方向第二方发送一个选项命令, 不管该命令是请求还是应答, 而且使用该选项将会对从第一方发送到第二方的数据进行处理时产生影响, 那么必须把该命令插到数据流中它希望开始起作用的点上。(要注意到在传送请求和接收到可能是否定的应答的过程需要一些时间。因此, 一台主机可能在发出请求一个选项的请求后希望缓冲要发送的

数据，直到它知道该请求是被接受还是被拒绝，来隐藏这段对用户来说是“不确定”的时间。)

选项请求在 **TELENT** 连接刚刚建立起的时候要在在连接的两端来来回回传送许多次，每一方都试图从对方获取尽可能好的服务。然而，在另一方面，选项可以用来动态地改变连接的特性，使它与对本地状态的改变相一致。例如，**NVT**(后面将要解释)使用的传输方式比较适合一个用 **BASIC** 语言编的应用，这类应用在传输数据时是每次一行，而对那些每次传输一个字符的应用(比如 **NLS**)就不是很适合。当对本地的处理来说是合适的，一个服务器可能会忍受这种“临时的特征”所需的巨大的处理器开销，并且会谈判一个合适的选项。然而，当不再需要详尽的控制时，处理开销可以(通过谈判)切换回 **NVT** 下的状态。

如果一个过程在收到一个拒绝回应后，仅仅是重新请求该选项，那么由一个过程发起的请求将会导致不停的请求循环。为了防止出现这样的循环，不能重复被拒绝的请求，除非已经改变了某些选项。在运行中，这可能意味着该过程运行一个不同的程序，或者用户已经发出了另外的命令，或者出现了其他所有可以影响一个过程及其选项的上下文的东西。根据经验，重新请求只能是一个连接的另外一端在后来又提交了某些信息，或者本地用户的交互的需要。

选项的设计者不应该拘泥于选项谈判中有限的一些语法。使用简单的语法的本意是希望使得选项易于使用 - 因为要忽略它们是很容易的。如果有一些特殊的选项需要一个比“**DO, DON'T, WILL, WON'T**”更完整的谈判结构，一个比较好的方法是用“**DO, DON'T, WILL, WON'T**”使双方都能理解该选项，一旦这个过程已经完成，就可以自由地使用一个更为特别的语法。比如，一方可以发送一个请求来通知(建立)一行的长度。如果这个请求被另一方所接受，那么可以用另外一个不同的语法来进行实际的对一行的长度的谈判 - 如一个“子谈判”可能包括可以允许的最小值，可以允许的最大值，以及最合适的行的长度等字段。一个较为重要的原理是，这样的扩展谈判只有在前面的一些(标准)谈判已经建立，并且双方都可以解释这些扩展语法的情况下才能开始。

总之，**WILL XXX** 由双方发送出去，表示该方希望(提出)开始对选项 **XXX** 进行处理。**DO XXX** 和 **DON'T XXX** 表示它的肯定和否定回应；类似地，**DO XXX** 发送出去指示(请求)对方(也即 **DO** 的接收者)开始对选项 **XXX** 进行处理，**WILL XXX** 和 **WON'T XXX** 表示肯定和否定回应。

由于在没有使用任何的选项的情况下，**NVT** 通过使用 **DON'T** 和 **WON'T** 回应来保证连接在连接的双方都可以处理的状态中。因此，所有主机都应该这样实现它们的 **TELNET** 进程：在完全不知道一个不支持的选项的情况下，只需要简单地拒绝任何无法了解的该选项请求。

TELNET 协议尽可能地使服务器和用户之间是对称的，以便比较容易和自然地包含用户到用户(连接)和服务器到服务器(协作处理)这两种情况。尽管不是完全需要，但我们也希望选项能够加强这个目的。在任何情况下，我们更倾向于明确承认对称性是一个操作上的原则，而不是一个不变的标准。

请参考相关文档“**TELNET 选项规范**”来得到关于如何建立新的选项的信息。

网络虚终端

网络虚终端（NVT）是一个双向的字符设备。NVT 有一个打印机和一个键盘。打印机负责进来的数据，而键盘负责产生通过 TELNET 连接发送出去的数据，并且在需要“回显”时，同时在 NVT 的打印机上回显这些数据。”回显“并不要求数据一定要经过网络（尽管有一个选项可以控制该操作的”远程“模式，但并不要求主机实现该选项）。

除了在这里说明的外，所有的编码集合都是有八位的，但只使用其中的七位的 USASCII 码。所有的代码转换和时区方面的问题都是本地的事情，而不影响 NVT。

数据的传输

尽管一个通过网络连接的 TELNET 连接本质上是全双工的，但通常把 NVT 看作在线性缓冲模式下的半双工设备。也就是说，除非已经和对方谈判好，以下情形 对应于通过 TELNET 连接进行数据传输。

1) 在本地缓冲空间允许的可用范围内，可以在产生数据的机器上汇集数据，直到完整的一行数据已经准备好传输，或者某些在局部定义的信号明确地要求传输数据。这些信号既可以有进程产生，也可以有用户发出。

定义这个规则的动机是，对于一些主机，处理网络输入中断的代价是很高的，另外，缺省的 NVT 规范指定“回显”操作的数据不经过网络的传输。因此，有理由在产生数据的源上缓冲一些数据。许多系统都会在输入一行结束后进行一些动作（行式打印机或者卡片打孔机经常都是这样子的），因此数据传输可以在一行数据结束时触发。另外，有时候一个用户或者进程会发现有必要或者应该提供一些不在一行的结尾结束的数据；因此实现者要注意，提供的局部信号机制要确保所有的缓冲数据都能够被立即发送出去。

2) 当一个过程已完成向一个 NVT 打印机发送数据，并且输入队列中也没有来自 NVT 键盘，需要进一步进行处理的数据（就是说，当一个在 TELNET 连接的一端的过程无法在另一端没有数据输入的情况下进行处理），该过程必须传输 TELNET 的继续（Go Ahead,GA）命令。

这个规则并不要求在一个连接的两端上的终端都发送 TELNET GA 命令，因为服务器开始进行处理时，一般情况下都不需要一个特别的信号（以及断开连接信号和其他在本地定义的特性）。况且，TELNET GA 被设计来帮助一个具有“可锁定”键盘的本地计算机（如 IBM2741）建立一个物理上的半双工终端。这种终端的一个说明可能对解释 GA 命令的正确用法有帮助。

终端到计算机的连接总是在用户或者计算机的控制之下。任何一方都不能单方面地夺取另一方的控制；而且取得控制的一方必须明确地放弃它地控制。在终端这一方，硬件上就支持在每次一个“连接”终止的时候（也就是在用户按下“新连接”的键时），它就放弃控制。当这种情况发生时，连接的（本地）计算机处理输入的数据，决定是否要产生输出，如果不需要的话，就把控制返回给终端。如果要产生输出，计算机维持控制，直到所有的输出都被

传输完毕。

通过网络使用这种类型的终端，困难是显而易见的。“本地”计算机在看到一个结束连线信号后，无法决定是否要保持控制，这个决定只能由处理这些数据的“远程”计算机作出。因此，TELNET 中的 GA 命令提供了一个机制，使“远程”计算机（服务器）如何给“本地”计算机（用户）发送信号，告诉对方现在是给用户终端传递控制的时间。当用户需要获得对终端的控制时，它应该并且只能在这段时间传递。注意，过早地传递 GA 命令将导致输出阻塞，由此用户可能会认为传输系统已经被暂停，因此将导致用户手工转向连接时失败。

当然，前面所说的这种情况不会在通讯过程中用户到服务器这个方向上出现。在这个方向上，尽管没有必要，但在任何时候都可能发送出 GA。同样，如果 TELNET 连接被应用在过程到过程的通讯中，在两个方向上都不需要发送 GA。最后，对于终端到终端的通讯，在两个方向上可能都需要 GA。如果一个主机打算支持终端到终端的通讯，建议主机在需要通过 TELNET 连接发送 GA 的时候，提供一个手工发信号给用户的方法。然而，在实现 TELNET 过程中，这一点并不是必需的。

注意：由于 TELNET 模型的对称性，从理论上来说，在一个 TELNET 连接的每一端，都必须有一个 NVT。

控制功能的标准表示

就象我们在本文档的简介中所说，TELENT 协议的主要目标是在通过网络连接的终端设备和面向终端的过程之间提供一个标准的接口。早期具有这种互联性质的实验表明，大部分的服务器都实现了某些功能，但调用这些功能的方式却差别很大。对于一个要与多个服务器系统交互的用户来说，这些差别是一个非常大的障碍。因此，TELNET 协议定义了这些功能中的下面 5 种的标准表示。这些标准表示包括标准，含义 --- 尽管不是必需的（除了中断进程（IP）功能，使用 TELENT 协议的其他协议可能需要该功能）。因此，一个没有给本地用户提供某种功能的系统也没有必要给网络上的其他用户提供该功能，并且可以把该功能的标准表示当作 No 操作。在另一方面，如果一个系统已经给本地用户提供了该功能，那么它必须给网络上那些传该功能的标准表示的用户提供同样的功能。

中断进程 - Interrupted Process(IP)

许多系统提供挂起，中断，中止，终止用户进程的操作的功能。当用户确信他的进程已经进入了无穷尽的循环，或者不小心激活了一个并不希望激活的进程时，就要经常使用该功能。IP 就是调用该功能的标准表示。该功能的实现者需要注意，其他使用 TELNET 协议的协议可能要使用 IP，因此实现时要支持这些协议。

中断输出 -- Abort Output (AO)

许多系统提供了允许一个产生输出的进程在不向用户的终端发送输出的情况下完成运行（或者达到在完成运行的过程中将会达到的某一个停止点）的功能。

另外，该功能一般还清除那些已经生成但还没有实际打印（或者显示）到用户的终端上的输出。AO 是调用该功能的标准表示。比如，许多子系统通常会接受一个用户的命令，然

后以一个发送到用户终端的长的字符串作为回应，最后，给用户的终端发送一个“提示”字符（前面跟着<CR><LF>）来表示准备接受下一个命令。如果是在传输字符串的过程中接收到 AO，一个合理的实现应该停止继续传输字符串，而转向发送提示符和跟在前面的<CR><LF>。（这可能同接收到 IP 所进行的动作有一些差别。在接收到 IP 时，将导致停止字符串的传输并且从子系统中退出。

同时还需要注意到，对那些提供这种功能的服务器，可能还需要清除那些存在于系统外的缓冲机制（在网络中或者在用户的本地机器上）中的内容。完成这个过程的一个合适的方法是给用户的系统发送“同步”信号（将在下面描述）。

你在那里吗？ -- Are You There (AYT)

许多系统提供了给用户提供一个系统仍然在运行的一些可见的（如可打印的）迹象。这个功能可以在系统在一个想象不到的很长一段时间里都没有动静时（可能是由于用户没有想象到的计算时间，或者不正常的巨大系统负荷等导致。）由用户调用。 AYT 是调用该功能的标准表示。

消除一个字符 -- Erase Character (EC)

许多系统提供了删除在未删除字符前面或者用户提供的数据流中的“打印位置”最后面的一个字符的功能。该功能通常在键盘输入时输入了错误的字符时使用。 EC 是调用该功能的标准表示。

*注意：一个“打印位置”可能包含相互覆盖的几个字符，或者象下面的字符系列：
<char1> BS <char2>...

消除一行 -- Erase Line (EL)

许多系统提供了删除输出设备上的当前一行的全部数据的功能。该功能经常在用键盘进行输入编辑时使用。 EL 是调用该功能的标准表示。

TELNET 中的“同步 (SYNCH) ”信号

许多系统提供了一种机制，可以允许一个终端的用户对一个“失控”的进程重新获得控制权。上面描述的 IP 和 AO 功能就是这种机制的例子。当在本地使用时，这样的系统可以访问由用户提供的所有信号，而不管这些信号是一些普通字符或者是由电传打字机中的“BREAK”键或 IBM 2741 中的“ATTN”键发送的“带外”信号。然而当通过网络把系统联结起来时，这可能是错误的。网络的流程控制机制可能导致把这些信号缓冲到其他地方，比如用户的机器中。

为了解决这个问题，提出了 TELNET 中的“同步”机制。

一个同步信号包含一个同 TELNET 命令 DATA MARK 结合在一起的 TCP 紧急通知。该紧急通知与 TELNET 连接中的流程控制没有关系，接收它的进程用它来调用数据流的特殊处理过程。在这种模式中，立即对数据流进行扫描，查找下面定义的一些“有趣”的信号，

而把那些干涉的数据丢弃。

TELNET 命令 DATA MARK (DM)是数据流中的同步标记，表示所有特殊的信号都已经产生，接受者可以继续对数据流进行一般的处理。

同步信号通过 TCP 中的发送操作发送，在发送过程中需要把紧急标志设为“真”，并且把 DM 作为最后（或者唯一的）一个字节。

当许多同步信号快速地连续不断地发送时，可以合并紧急通知。不可能去计算紧急通知的次数，因为接收到的紧急通知的次数可能等于或者少于发送次数。在普通模式中，一个 DM 是没有任何操作的，但在紧急模式中，它表示紧急处理过程的结束。

如果在发现 DM 之前，TCP 已经指示紧急数据的结束，TELNET 应该继续对数据流进行特殊的处理，直到发现 DM。

如果在发现 DM 之后，TCP 指示有更多的紧急数据，它只能是另外同步信号。TELNET 应该继续对数据流进行特殊的处理，直到发现另外一个 DM。

定义的“有趣的”信号为：TELNET 中的 IP，AO，和 AYT (没有 EC 或 EL)的标准表示；与这些标准表示类似的本地表示（如果有的话）；所有的其他 TELNET 命令；其他在不延迟数据流的扫描并且能够起作用的自定义信号。

由于 SYNCH 机制的一个影响是丢弃本来在发送者和接收者之间要传输的所有字符(除了 TELNET 命令)，如果需要，这个机制可以作为清除数据路径的一种标准方式。例如，在一个终端上的用户需要传输一个 AO，接收到该 AO 的服务器应该给该用户返回一个同步信号（如果它提供该功能的话）。

最后，就象在 TELNET 层，需要把一个 TCP 紧急通知当作一个带外信号，因此其他使用 TELNET 的协议可能需要从不同层次来看可以当作带外信号的 TELNET 命令。

通过约定系列 [IP, Synch] 可以把它作为这样的信号。例如，假设有一个使用 TELNET 协议的其他协议定义了一个类似于 TELNET 命令 AO 的字符串 STOP。想象用户使用该协议的目的是希望服务器处理 STOP 字符串，但由于服务器在处理其他的命令，导致连接被阻塞。用户应该引导他的系统：

- a) 发送出 TELNET IP 字符；
- b) 发送出 TELNET SYNC 系列，也就是：在一个紧急模式的 TCP 发送操作中把 Data Mark (DM)作为唯一的字符发送出去。
- c) 发送出字符串 STOP；接着
- d) 如果有的话，把其他协议中类似于 TELNET DM 的命令发送出去。

用户（或者代表该用户的进程）必须传输上面步骤 2 中的 TELNET SYNCH 系列，以确保 TELNET IP 已经到达服务器的 TELNET 解释器。

紧急通知将激活 TELNET 进程，而 IP 将激活随后级别较高的进程。

NVT 打印机和键盘

NVT 打印机有一个没有指定宽度的走纸器，并且每一页的长度也没有指定。NVT 打印机可以产生所有 95 个 USASCII 编码的图形表示（从 32 到 126 的编码）。在 33 个 USASCII 编码（0 到 31 及 127）和未包含的其他 128 个编码（128 到 255）中，下面几个编码对 NVT 打印机有确定意义：

名称	编码	意义
NULL (NUL)	0	没有操作
Line Feed (LF)	10	打印头移到下一个打印行，但不改变打印头的水平位置。
Carriage Return (CR)	13	把打印头移到当前行的左边。

另外，在 NVT 打印机上，尽管不是必需的，同时应该定义下面这些编码。TELNET 连接的双方，都不会假设另一方在接收到或传输下面这些编码时将会，或者已经实施某种特殊动作：

BELL (BEL)	7	产生一个可以看到或可以听到的信号（而不移动打印头。）
Back Space (BS)	8	向左移动打印头一个字符位置。
Horizontal Tab (HT)	9	把打印头移到下一个水平制表符停止的位置。它仍然没有指定每一方如何检测或者设定如何定位这样的制表符的停止位置。
Vertical Tab (VT)	11	把打印头移到下一个垂直制表符停止的位置。它仍然没有指定每一方如何检测或者设定如何定位这样的制表符的停止位置。
Form Feed (FF)	12	把打印头移到下一页的顶部，保持打印头在相同的水平位置上。

剩下的其他编码都不会导致 NVT 打印实施任何动作。

在定义中，系列"CR LF"将导致 NVT 打印头移动到下一行的左边（与系列 "LF CR"的效果是一样的）。然而，许多系统和终端并不独立处理 CR 和 LF，为了模拟它们的效果，需要进行一些处理。（比如，许多终端没有独立于 LF 的 CR，但是在这样的终端上可以用退格键来模拟一个 CR。）因此，必须把系列 CR LF"当作一个单独的“新行”字符看待，并且在需要把它们结合在一起的时候使用它们。必须在只需要一个单独的回车键时使用系列"CR

NUL；在其他的场境中必须避免使用 **CR** 字符。这个规则可以确保系统在发现一个 **TELNET** 流中有一个字符的后面跟有 **CR** 的情况下，可以作出合理的选择：是进行“换行”功能还是进行多次的退格操作。

注意，在两个方向上（在缺省的 **ASCII** 模式下）都需要"**CR LF**"或者"**CR NUL**"，以确保 **NVT** 模式的对称性。

尽管在某种情况下（如当远程回显和禁止超前选项同时起作用时），可以认为字符并不被发送到一个实际的打印机上，然而，为了保证一致，在一个数据流中，如果一个 **CR** 的后面没有跟着一个 **LF**，该协议要求把一个 **NUL** 插到 **CR** 的后面。

相反，在接收方，如果从数据流中接收到一个跟在 **CR** 的后面的 **NUL**（在没有用谈判选项显式指定其他选择的情况下），在把 **NVT** 转换成本地字符集之前，应该把 **NUL** 去掉。

NVT 键盘有键或者键的组合，或者键系列来产生所有 128 格 **USAACII** 编码。要注意尽管一些在 **NVT** 打印机上没有什么用处，**NVT** 键盘还是可以生成。

除了这些编码，**NVT** 键盘还可以生成下面这些附加的编码，除注明外，还定义了这些编码的意义（尽管不是必需的）。

对这些“字符”的实际代码分配在 **TELNET** 命令这一节，因为从某种意义上来讲，我们可以认为这些编码是固有的，甚至在把数据流中的数据都解释为属于另外的一个字符集的时候，都可以使用这些编码。

Synch

这个键允许一个用户清空到另一方的数据通道。激活该键将导致发送一个带有 **TCP** 紧急通知的 **DM**（参看命令这一节）。一对 **DM**-紧急通知具有在前面定义的一些意义。

Break (BRK)

之所以提供这个编码，是因为在当前的许多系统中，它是 **USASCII** 集合之外的一个信号，并且具有本地意义。可以用它来表示 **Break** 键或 **Attention** 键已被按下。然而，需要注意的是，它的目的是给需要它的系统提供第 129 个编码，而不等同于 **IP** 的标准表示。

Interrupt Process (IP)

挂起，中断，中止，终止一个 **NVT** 连接的进程。另外，它也是那些使用 **TELNET** 的其他协议的带外信号的一部分。

Abort Output (AO)

允许当前的进程继续运行直到结束，但不给用户发送它的输出信息。并且把一个同步信号发送给用户。

Are You There (AYT)

给 NVT 发送回一些可见的（也就是可打印的）信息以表明已经收到 AYT。

Erase Character (EC)

接收者将删除数据流中最后一个未被删除的前导字符或者“打印位置”。

Erase Line (EL)

接收方将删除由 TELNET 连接发送的数据流中最后一个“CR LF”系列（但不包括该系列）后面的全部内容。

这些“额外”的键，也就是打印机的格式控制字符的本质是，它们是对从“NVT”到“本地”这个必须进行的映射过程的一个自然的扩展。

就象 NVT 中的字节 68（八进制 104），可以映射为本地中代表“大写 D”的任何一个编码，字符 EC 也可以映射为本地中代表“删除一个字符”功能。

另外，就象在一个没有“垂直线”字符的环境下，对编码 124（八进制 174）的映射是任意的，如果在本地没有“删除一个字符”这种机制，对 EL 的映射也是任意的（甚至不映射）。

类似地，对格式控制字符，如果终端确实有一个“垂直制表键”，那么对 VT 地映射就是显而易见的，只有在终端没有一个垂直制表键的情况下，VT 的作用才是无法预测的。

TELNET 命令结构

所有的 TELNET 命令至少包含一个两个字节的序列:跟在命令的代码的后面,"当作命令来解释(Interpret as Command)"(IAC)的转义字符。处理选项谈判的命令有三个字节系列,第三个字节就成了被选项引用的编码。之所以选择这种格式,是这种格式能够更大范围地使用"数据空间"---当然,是通过基本 NVT 的谈判来进行。数据字节与保留的命令值的冲突被大大减少了,而所有这些冲突都需要复杂,低效的方法来把数据字节转换为流。使用现在的方法,只有在需要把 IAC 当作数据发送时才需要把相同的数据发送两次,其他 255 个代码都可以透明地传输。

下面是所有已定义的 TELNET 命令。需要注意的是,这些代码和代码序列只有在前面跟有一个 IAC 时才有意义。

名称	代码	意义
SE	240	子谈判参数的结束
NOP	241	空操作
Data Mark	242	一个同步信号的数据流部分。该

		命令的后面经常跟着一个 TCP 紧急通知
Break	243	NVT 的 BRK 字符
Interrupt Process	244	IP 功能.
Abort output	245	AO 功能.
Are You There	246	AYT 功能.
Erase character	247	EC 功能.
Erase Line	248	EL 功能.
Go ahead	249	GA 信号.
SB	250	表示后面所跟的是对需要的选项的子谈判
WILL (option code)	251	表示希望开始使用或者确认所使用的是指定的选项。
WON'T (option code)	252	表示拒绝使用或者继续使用指定的选项。
DO (option code)	253	表示一方要求另一方使用, 或者确认你希望另一方使用指定的选项。
DON'T (option code)	254	表示一方要求另一方停止使用, 或者确认你不再希望另一方使用指定的选项。
IAC	255	Data Byte 255.

连接的建立

TELNET TCP 连接是在用户端口 U 和服务器端口 L 之间建立的。服务器在用于这种类型的连接的一个众所周知的端口 L 上监听客户请求。由于一个 TPC 连接是全双工的, 并且通过双方的端口来标识, 服务器可以对不同的用户端口 U 和端口 L 的之间的许多并发连接进行应答。

端口分配

当用来给远程用户提供访问服务主机的服务(也就是远程终端访问), 这个协议分配了服务端口 23(把进制 27)。也就是 L=23。

本 RFC 指定了一个 ARPA 互联网社区的标准。在 ARPA 互联网上的主机应该采纳与实现该标准。

给 TELNET 协议提供一些选项的目的是, 使相互通信的主机在解决不同设备之间的通信问题时获得比由网络虚拟终端 (NVT) 提供的可能框架有更好的方案。它可以让主机自由地创建, 测试或者丢弃某些选项。当然, 可以想象, 那些普遍有用的选项最终大部分的主机都应该支持。因此, 应该仔细地设计这些选项的文档, 并且尽可能地公布它们。另外, 确保不在不同地选项中使用相同的选项代码也是必要的。

本文档指定了一个选项代码的分配和选项的文档标准方面的方法。在进行试验时, 可能只需要选项代码分配而不需要完整的文档, 不过一般来说, 在分配选项代码之前都需要一个文档。我们通过把一个选项的文档作为一个 RFC 文档来发布, 从而发布该选项。当然, 选项的创

建者也可以用其他的方式发布选项。

选项代码由下面人员分配：

Jonathan B. Postel
University of Southern California
Information Sciences Institute (USC-ISI)
4676 Admiralty Way
Marina Del Rey, California 90291
(213) 822-1511

Mailbox = POSTEL@USC-ISIF

选项的文档至少要包含下面几个小节：

第 1 节 -- 命令的名称和选项的代码

第 2 节 -- 命令的意义

应该描述同该选项相关的每一个 TELNET 命令的意义。需要注意的是，对于复杂的选项，“子谈判”是必需的，因此可能有许多相关的命令。“子谈判”的原理在下面有更详细的描述。

第 3 节 - 缺省的规范

对那些没有实现，或者没有使用该选项的主机，必须描述这些选项在这些主机中的缺省假定值。

第 4 节 - 动机

对创建一个特殊的选项，或者对某种选项选择一种特殊的格式的动机进行详细的描述，对那些还没有碰到（或者虽然已经碰到，但没有认识到）该选项设计来解决的问题的人，是非常有用的。

第 5 节 - 描述（或者实现规则）

为了确保一个命令的两个不同实现相互之间能够通讯，仅仅定义命令的意义和对该命令的意图进行说明有时候是远远不够的。因此，在许多情况下，我们需要给一个命令提供一个完整的描述。这个描述可以用文本来表示，也可以是一个示例性的实现，或者是实现的线索等等。

对“子谈判”的解释

在主机之间传递选项时，除了一个选项编码外可能还需要更多其他信息。例如，要求一个参数的那些选项就属于这种情况。在主机之间传递除了选项代码外的其他信息的策略包含两个步骤：双方都同意去”商讨“该参数，第二，对参数进行”商讨“。

在第一步中，同意去讨论参数以一种普通的方式来进行。一方通过发送一个带有选项代码的 **DO**(或 **WILL**)命令来建议使用选项，另一方发送一个带有选项代码的 **DO**(或 **WILL**)命令来表示接受这个建议。一旦双方都同意使用这选项，通过在 **SB** 命令的后面跟上相应的选项代码，参数和命令 **SE** 来开始子谈判。每一方都被假设为能够解析该参数。因为在最初通过交换 **WILL** 和 **DO** 命令，双方都表明可以支持该选项。另外，即使接收方不能解析该参数，接收方也可以通过搜索 **SE** 命令（如字符串 **IAC SE**）来定位参数字符串的开始位置。当然，在任何时候，任何一方都可以给另一方发送 **WON'T** 或 **DON'T** 来拒绝继续进行进一步的子谈判。