第十一章 網路終端機系統 – telnet

11-1 網路終端機簡介

每談到『網路終端機』(Network Terminal)，都會想到 Telnet 命令程式，使用過網路的人對它大多不會陌生，最普遍的應用是上 BBS 的 Telnet 連線。其實 Telnet (Telecommunication Network Protocol)是一種通訊協定，它的發展可追溯到 1969 年，幾乎是與 ARPANET 同時誕生，也是 ARPANET 開始時最基本的應用程式。當時 Berkeley Unix 也有發展自己的網路終端機系統，稱之為 Rlogin，但 Rlogin 功能較少，一般都使用在 Unix 系統之間的連線。近年來不同系統之間連線漸漸普遍，Rlogin 的應用也大多移植到功能較強的 Telnet 上，這也是本章以 Telnet 來介紹網路終端機最主要的原因，但首先我們來瞭解何謂『終端機』(Terminal)。

11-1-1 何謂終端機？

在『多人/多工』(Multi-user/Multi-task) 系統下，一般使用者必須透過終端機連線，才可以使用到系統的資源，它的架構如圖 11-1 所示。主機電腦利用串列 (Series)『多工器』(Multiplex) 的連線和終端機銜接，其連線方式依照各家廠商製作有所不同，但大部份都是以 RS-232C 連線方式。一般來講，一部主機可連接數十部到數百部的終端機，它的連線架構如圖 11-1 (a) 所示。經過連線之後，終端機由鍵盤輸入各種命令給主機電腦處理，主機電腦處理後將結果顯示在終端機螢幕上，基本上，終端機只負責使用者和主機電腦之間的交談 (Interactive) 工作，並不負責其它有關資料的處理。圖 11-1 (b) 為一般 Unix/Linux 連線啟動程序，主機電腦啟動後 (執行 Run Level 3)，便執行 gettty 程式來掃描監視多工器上是否提出連線要求，如有連線進來再啟動使用者 Shell Script，處理使用者的登入工作 (Login Script)。使用者在終端機所下的命令，就宛如在主機的主控台所下命令一樣，所執行的命令也是在自己的 Login Script 下所產生的。

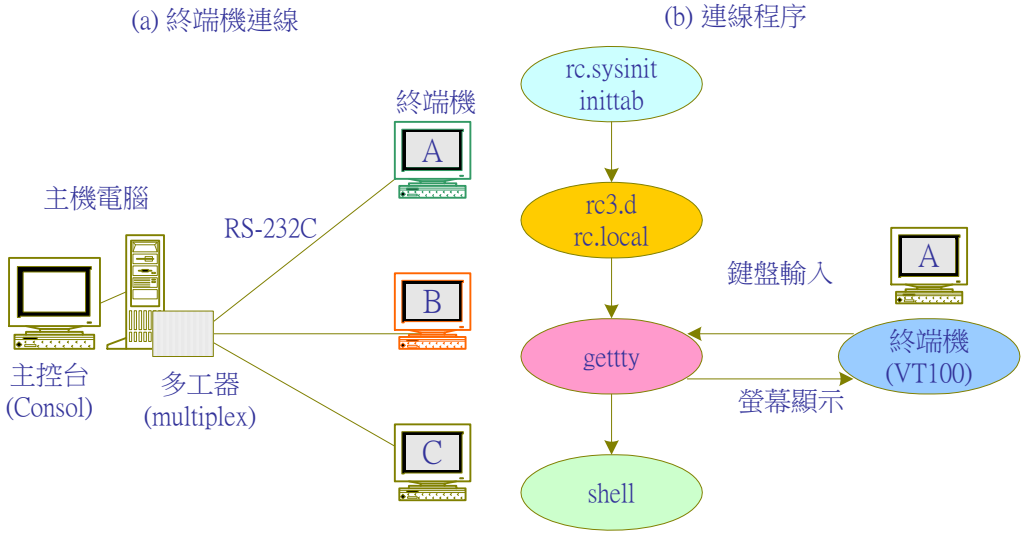

圖 11-1 主機電腦之終端機系統

早期各家電腦廠商都有自己的終端機系統，主要是配合自己作業系統的功能而開發的。隨著廠商愈多而自行開發的終端機不能配合其它電腦系統使用，大部份廠商的終端機系統漸漸地模擬（Emulation）成較大型公司型態，以配合不同環境的使用，當時大型公司以 IBM 和 Digital 為最，因此一般終端機也是以 IBM 公司的 IBM 3270、5250 或 Digital 公司的 VT 52、VT100、VT220 為模擬對象。但 IBM 所使用的字元編碼（EBCDIC）並不相容於一般電腦系統的編碼（ASCII），因此一般還是以 VT 系列的終端機系統較多，這也造成目前網路終端機也是以 VT100 系列較多。

11-1-2 何謂網路終端機？

由上述可以瞭解，主機電腦和終端機之間採用固定連線（如 RS-232C），終端機將被固定在某一部主機上，否則就必須佈放另一連線來連接其它主機。隨著個人電腦愈來愈便宜，我們不希望再製作沒有處理計算能力的終端機，而以有處理能力的個人電腦來取代終端機設備，漸漸地有終端機模擬程式的出現。簡單的說，我們可以依照連線需要將個人電腦模擬成各式各樣的終端機（如 IBM 3270、VT 100、VT220 等等），來連接不同主機系統，早期模擬終端機的電腦和主機之間還大多採用類似 RS-232C 的連線方法。隨著網路應用的蓬勃發展，佈放網路連線愈來愈普遍，固定連線的 RS-232C 也漸漸由網路連線來取代，**『網路終端機』（Network Terminal）**的雛形就因而建立而成，其連線架構如圖 11-2 所示。

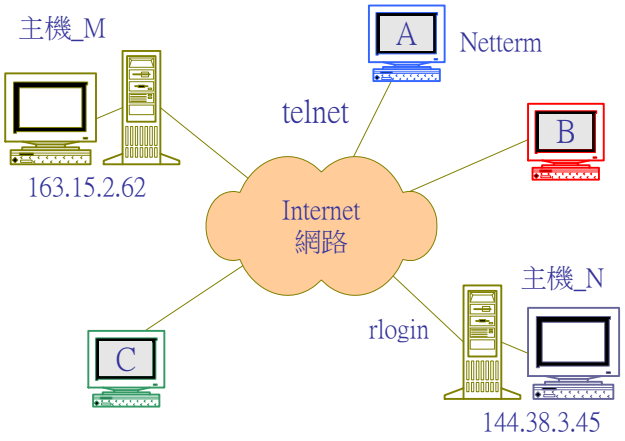

圖 11-2 網路終端機連線

圖 11-2 為網路終端機連線方式，使用者在客戶端電腦上執行終端機模擬程式（如 Netterm 等），將電腦模擬成所需要的終端機型態（如 IBM 3270、VT100、VT220 等），再透過網路通訊協定（如 Telnet 或 Rlogin 等等）和主機連線。此時就將客戶端電腦模擬成終端機，由客戶端鍵盤輸入各種命令給主機電腦，主機電腦執行後將結果顯示在客戶端電腦的螢幕上，原來透過 RS-232C 連線方式，就由網路連線來取代它。另一方面，客戶端（如電腦_A）可依照不同的需求，可分別或同時連線到不同主機上（如主機_M 或主機_N），又不受限於主機和客戶端之間的距離位置，只要網路可以連結到的地方即可。因此，客戶和主機之間便可以在無遠弗界的 Internet 網路上連接。

又從另一個角度來看，我們只要知道對方主機的位址和使用者名稱及密碼，便可登入到 Internet 網路上任何一部主機，並可由鍵盤輸入各種命令請求主機來執行，這將是一件非常嚴重的事情，有心人士只要偷竊到使用者密碼（尤其是系統管理者密碼），便可從事破壞的工作，而不論破壞者位於 Internet 網路上哪一角落，都可以容易達成。也就是說，網路終端機是網路安全上最大的漏洞，因此，必須要做特殊的安全防範（如 TCP Wrappers），甚至有些單位關閉網路終端機功能。

11-2 Telnet 通訊連線

圖 11-3 為 Telnet 通訊連線方式，客戶端透過終端機模擬程式，以 Telnet 通訊協定來連接伺服端主機，模擬程式可能是目前 Windows 系列最常用的 NetTerm 應用程式（或其它程式），也可以是主機內的 NVT（Network Virtual Terminator）程式。在這連線之中有下列重點描述：

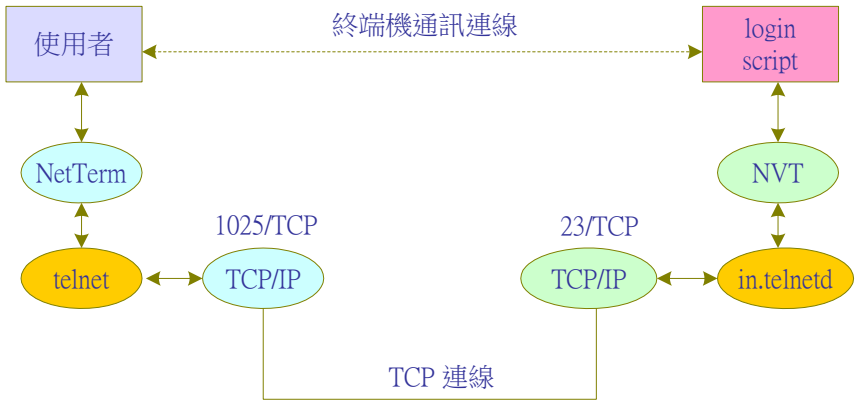

圖 11-3 Telnet 的通訊連線

- Telnet 通訊連線是透過 TCP 傳輸通訊鏈路連接，而 Telnet 伺服器是連接在 TCP/23 著名埠口上。客戶端利用一空閒 TCP 埠口(如 1025/TCP)連結到伺服端 TCP 埠口 23(23/TCP) 上，它們之間也是透過 Socket 通訊端點。

- 在客戶端可以透過一般功能較強的終端機模擬程式 (如 NetTerm 等)，或直接經過系統核心所提供的 NVT 程式，直接呼叫 NVT 連結伺服端。亦是直接在系統下執行 Telnet 命令，如 Windows 系統以 telnet 或 Linux 系統以 # telnet 下達執行命令。

- 伺服端必需隨時監督 TCP 埠口 23 (23/TCP) 是否有連線要求 (一般都由 xinetd 服務程式負責)，當有連線要求時便啟動 Telnet 伺服程式 (in.telnetd)，由 Telnet 伺服程式來負責和 Telnet 程式之間的通訊。

- Telnet 伺服程式必須處理有關**『虛構終端設備』(Pseudo Terminal Device, PST)** 的程序，而在 Internet 網路上都使用 NVT 標準。

- 伺服端透過 NVT 處理程序，此時必需協商雙方使用的終端機型態，並呼叫使用者的登入程序 (Login Script)，由 Login Script 詢問使用者名稱和密碼，如使用者登入成功，便啟動 User Script 來接受與處理使用者所下達的命令。

- 我們可以發現，伺服端主機處理**『虛構終端設備』**，就好像直接連線 (11-1-1 介紹) 的終端機系統中，主機電腦隨時掃描監督連線多工設備一樣，而使用者登入情形也宛如和主機直接連線情況一樣。

如果我們直接由 win98-1 (163.15.2.34) 上 Telnet 登入到 Linux-1 (163.15.2.62)，則執行 C:\> telnet 163.15.2.62，則結果如下：

```
Red Hat Linux release 6.2 (Zoot)
Kernel 2.2.14-5.0 on an i586
login: tsnien
Password:
Last login: Fri Jun 14 09:58:15 from 163.15.2.34
You have mail.
[tsnien@linux-1 tsnien]$
```

可在 Linux-1 上可觀察 Telnet 伺服程式（in.telnetd）的執行情形：(節錄部份內容)

```
[tsnien@linux-1 tsnien]$ ps –ef
….
root        763    550   0 18:36 ?          00:00:00 in.telnetd: 163.15.2.34
root        764    763   0 18:36 pts/0      00:00:00 login -- tsnien
tsnien      765    764   0 18:37 pts/0      00:00:00 -bash
```

由上可以發現 Telnet 伺服程式(in.telnetd)正連接 163.15.2.34 的 Telnet 通訊程式，而 Login Script（login -- tsnien）和使用者程序（User Script）都是建立在同一虛構終端設備（pts/0）。由它們之間 PID（ Process Identifier）也可以發現，是由 in.telnetd 產生 Login Script（login - tsnien），再由使用者登入程序中在產生使用者程序（-bash）。

11-3 Telnet 協定堆疊

圖 11-4 為 Telnet 的通訊協定堆疊，它是建構在 TCP 協定之上，一般伺服端（in.telnetd）都銜接在埠口 23 上，同時具有多工連線的功能（如 12-2-2 介紹），而客戶端可任取 1024 以後的空閒埠口(如 1025)。一般在 Telnet 通訊鏈路中所傳輸的訊息大多是以交談式(Interactive)方式，這就是標準的終端機和主機之間通訊方式，因此，伺服端和客戶端之間的傳輸資料非常的少，照理論來講並不需要使用連接方式(TCP)，但是終端機和主機之間大多希望能即時反映所下達的命令，也希望它們之間能連續且安全性的傳輸，因此，才會採用連接導向的 TCP 連線。

如同圖 11-4 的通訊協定堆疊，圖 11-5 為 Telnet 的封包格式，它是 Internet 網路上透過 Ethernet 網路傳送的訊框方式。在 TCP 封包的資料區中，所存放的訊息就是 Telnet 的封包內容，也就是 Telnet 的命令及所傳遞訊息。至於 Telnet 有哪些命令將會在下一節介紹。

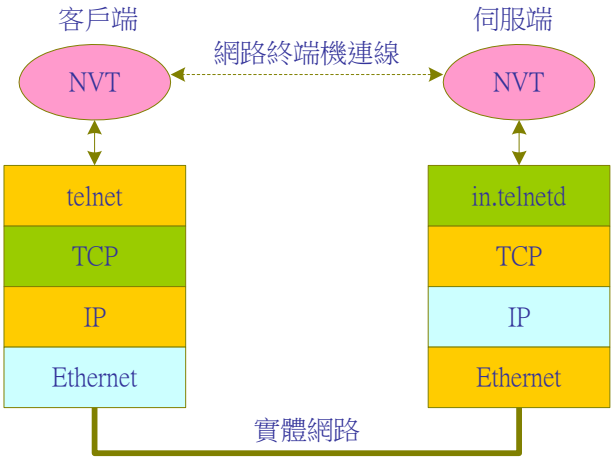

圖 11-4 Telnet 協定堆疊

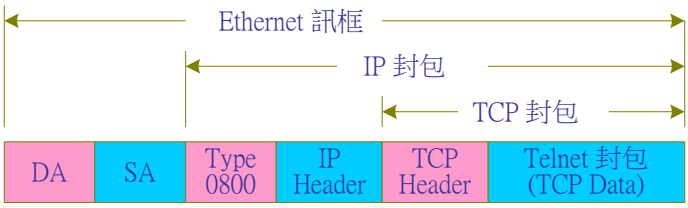

圖 11-5 Telnet 封包格式

11-4 Telnet 通訊協定

早期 Telnet 通訊協定是在 RFC 854 中規範，它是定義最低通用的同質性終端機協定，將客戶端和伺服端之間模擬成一虛擬終端機來互相通訊，稱之為**『網路虛擬終端機』(Network Virtual Terminal, NVT)**。透過虛擬想像的 NVT 裝置，將客戶端和伺服端的真正終端機映射到 NVT 上，也就是說，雙方不論終端機型態為何，都將其映射到 NVT 上，便可以利用 NVT 格式來互相通訊，其中便可省略許多相容性的問題。

如果各種網路應用系統的 Client/Server 雙方都模擬成 NTV 來相互通訊，也就將客戶端所傳送的各種命令，都好像是從主控台的鍵盤輸入一樣，便可省略掉 Internet 網路上各種異質性電腦之間的相容性問題，這也是我們在 12-2-3 節中介紹 Internet 網路上應用系統的通訊方式。也就是說，Telnet 的通訊方式後來都使用在各種應用系統上。有關 NVT ASCII 通訊方式，我們在 12-2-3 節中已介紹過，不再重複說明，以下介紹 Telnet 的通訊命令。

11-4-1 Telnet 通訊命令

在 NVT ASCII 中都是以 8 位元來編碼欲傳輸的資料，但一般 ASCII 碼都為 7 位元，因此，我們將最高位元（第 8 位元）都設定為 0。但一些有關圖形文字碼（如，中文的 Big-5 碼）和影像檔，大多會用到 7 位元以後（第 8 位元為 1）的碼，因此，一般 Internet 應用系統的控制字元大多使用在 8 位元編碼中較後面的，也就是以較接近 255（0xFF）的數碼。有關 Telnet 通訊命令的編碼，在 RFC 856 中有詳細規範，表 11-1 列出 Telnet 通訊命令。

表 11-1 Telnet 通訊命令彙集

名稱	碼（10 進位）	碼（16 進位）	功能描述
EOF	236	0xEC	檔案結束（End of File）。
SUSP	237	0xED	暫停目前處理動作（Suspend）。
ABORT	238	0xEE	中斷放棄（Abort）。
EOR	239	0xEF	紀錄結束（End of Record）。
SE	240	0xF0	子選項結束（End of Subnegotiation）。
NOP	241	0xF1	沒有動作（No Operation）。
DM	242	0xF2	資料標記（Data Mark）。
BRK	243	0xF3	中斷（Break）。
IP	244	0xF4	中斷處理程序（Interrupt Process）。
AO	245	0xF5	放棄輸出（Abort Output）。
AYT	246	0xF6	您還在嗎？（Are You There）。
EC	247	0xF7	刪除字元（Erase Character）。
EL	248	0xF8	刪除字行（Erase Line）。
GA	249	0xF9	前進（Go Ahead）。
SB	250	0xFA	子選項開始（Subnegotiation Begin）。
WILL	251	0xFB	期望（Will）子選項協議。
WONT	252	0xFC	不期望（Won’t）選項協議。
DO	253	0xFD	要求（Do）選項協議。
DONT	254	0xFE	不要求（Don’t）選項協議。
IAC	255	0xFF	直譯命令（Interpret as Command）

主機和終端機雙方所傳送的命令都以 IAC（0xFF）為前導，也就是說，在雙方傳輸當中，IAC 後面緊接著通訊命令。但如果傳輸資料中也有 IAC（0xFF）的位元組，則再插入一個 IAC 來表示資料的意思，其情況就好像網路系統中的字元填塞（Character Stuffer）一樣。

11-4-2 Telnet 選項協議

就我們所瞭解，主機和客戶端之間必須經過協議來決定雙方的通訊方式，譬如，終端機型態、傳輸模式等等，因此雙方隨時都必經過協議來決定某一事項，這些協議的工作就利用選項命令來達成。Telnet 中有下列四種選項命令：

(1) WILL：發送者期望自己選項被啟動（Enable），並詢問對方。

(2) DO：接收者同意對方選項，並回應給發送者。

(3) WONT：發送者想要停用（Disable）自己的選項，並詢問對方。

(4) DONT：接收者不同意對方選項，並回應給發送者。

雙方利用上述四種協議命令，來協調雙方的選項，任何一方都可以拒絕或接受對方的選項，因此，可能發生下列六種情況：(表 11-2)

表 11-2 選項協定運作

	發送者		接收者	說　　明
1.	WILL	→		發送者期望自己選作用。
		←	DO	接收者同意對方選項。
2.	WILL	→		發送者期望自己選作用。
		←	DONT	接收者不同意對方選項。
3.	DO	→		發送者希望對方啟動選項項目。
		←	WILL	接收者回應選項項目。
4.	DO	→		發送者希望對方啟動選項項目。
		←	WONT	接收者回應無法提供。
5.	WONT	→		發送者希望停用自己的選項。
		←	DONT	接收者回應停用該選項（必需同意）。
6.	DON'T	→		發送者要求停用對方的選項。
		←	WONT	接收者回應停用自己選項（必需同意）。

當雙方欲執行上述協議動作時，必需包含三個位元組：一者為 IAC 表示命令的開始；二者為協議命令(如，WILL、DO、WONT、DONT)；三者為欲協議之選項項目的 ID(如，終端機型態、視窗大小、流量控制等等)。每一個位元組表示一種命令，因此，協議命令格式如下：

< IAC, WILL（或 DO、WONT、DONT）, 功能碼_ID >

目前 Internet 網路上針對 Telnet 的選項項目有超過 40 種以上，當然並不是所有的選項都會應用得到，這可以依據使用者的環境而定。這些選項都由各自的 RFC 指定 ID 號碼，並說明其功能，我們在表 11-3 中列出一些較常用的選項。

表 11-3 常用之選項項目

ID	名　稱	說　明	RFC
0	Transmit Binary	設定為 8 位元傳輸模式。	856
1	Echo	Echo 回應輸入位元到螢幕上。	857
3	Supress GA	抑制前進訊號（Go-ahead）。	858
5	Status	詢問對方選項狀態。	859
6	Time Mark	通訊中插入時間標記。	860
24	Terminal Type	終端機型態。	1091
31	Window Size	視窗大小。	1073
32	Terminal Speed	終端機速度。	1079
33	Remote Flow Control	遠端流量控制。	1372
34	Linemode	行輸入模式。	1184
36	Environment Variable	環境變數設定。	1408

11-4-3 Telnet 子選項協議

並非所有選項都只要指定選項項目就可以完成，有些選項必需詳細的指定某些參數，譬如終端機選項，當雙方協議好某一種終端型態之後，也許還必須協議有關終端機的傳輸模式，如 ASCII 字元傳輸模式等等，因此，就有必要在某一選項之下，再協議一些子選項（Sub-negotiation）項目。基本上，每一選項的子選項項目都會在自己的 RFC 規範中制定，但它們的協議方式也都大同小異。一般都以 <IAC, SB, ..> 表示子選項的開始，又以 <.... IAC, SE> 表示子選項的結束，其中 IAC（0xFF）表示交談命令開始；SB（0xFA）表示子選項開始；SE（0xF0）為子選項結束。我們以 RFC 1091 中規範終端機選項，來介紹 Telnet 的終端機型態之協議運作方式，依照圖 11-6 中標示順序號碼說明如下：

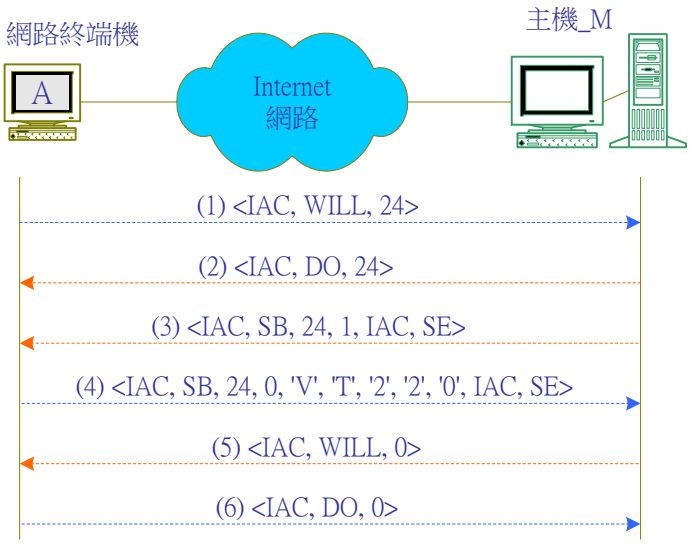

圖 11-6 Telnet 選項協議範例

(1) 首先由客戶端（網路終端機）要求協議終端機型態（<IAC, WILL, 24>），選項 24 表示終端機型態（如表 11-3）。

(2) 伺服端同意對方協議終端機型態，則傳送 <IAC, DO, 24> 給客戶端。

(3) 伺服端欲向客戶端詢問終端機型態，則送出 <IAC, SB, 24, 1, IAC, SE>，其中子選項 1 表示**『請傳送您的終端機型態』（Send）**的意思。

(4) 客戶端收到伺服端的詢問訊號後，便傳送自己的終端機型號給伺服端，發送 <IAC, SB, 24, 0, 'V', 'T', '2', '2', '0', IAC, SE> 給伺服端，其中 0 表示**『我的終端機型態是』（Is）**的意思，而 VT220 是以 ASCII 字元編碼，每一字元以 8 個位元表示。

(5) 此時，伺服端繼續協商希望以 8 位元型態傳輸資料，因此發送 <IAC, WILL, 0> 給客戶端，其中選項 0 表示用 8 位元傳輸模式（如表 11-3 所示）。

(6) 客戶端如同意該選項，則回應 <IAC, DO, 0>；否則回應 <IAC, DONT, 0>。

11-5 Telnet 操作模式

一般來講，Telnet 客戶端和伺服端之間有下列三種操作模式：

(A) 半雙工模式（Half-duplex Mode）

NVT 的預設模式是半雙工方式，此模式現在已非常少使用。半雙工模式表示同一時間內只有單一方面傳送資料，客戶端（或終端機）必需收到伺服端傳來一個 Go-ahead（GA）命令之後，才

可以由鍵盤輸入命令，也只有一行完整命令才會由伺服端收到回應。目前許多網路終端機型態都希望提供全雙工的傳輸模式，因此提供 Echo 選項(RFC 857)和 Suppress Go-ahead 選項(RFC 858) 功能，來提供單一字元模式與行模式。

(B) 單一字元模式（Single Character Mode）

單一字元模式是表示一次一個字元方式，當客戶端鍵入字元時，便將所鍵入的字元一個一個傳送給伺服端，伺服端再回應大多數的字元。也就是說，使用者在網路終端機上的鍵盤輸入時，每鍵入一個字元便傳送給伺服端，當伺服端收到許多字元，也了解使用的命令要求後，再回應處理結果給使用者。早期 Unix 系統上的 Telnet 協定大多採用這個模式。通訊雙方的 NVT 欲進入這個模式，必須抑制前進功能，一般都由客戶端送出 <IAC, DO, Suppress Go-ahead> 給伺服端，再由伺服端回應同意，接著再由伺服端送出 <IAC, WILL, Echo> 來協商採用字元模式。也就是說，雙方 NVT 必需協商同意 Suppress Go-Ahead 和 Echo，才真正進入一次一個字元模式。

(C) 行模式（Line Mode）

行模式表示使用者由鍵盤輸入一行後，再傳送給伺服端處理，一般稱此模式為 Kludge 模式。雙方 NVT 除了必須抑制 Go-Ahead 功能外，也必須協商行模式選項，一般都由客戶端送出子選項協商 <IAC, SB, WILL, 34, IAC, SE> 訊號，再由伺服端同意該子選項。當客戶端輸入一行命令後，希望該命令停止執行，一般都可以用 Ctrl+C 鍵盤命令來中斷它，該命令轉換成 Telnet 命令為 <IAC, IP>，並傳送給伺服端。其中 IP（Interrupt Process）為中斷處理，這就如同在主控台上輸入 Ctrl+C 的功能一樣。

Telnet 連線除了上三種傳輸模式外，還有許多有關傳輸功能必須經過雙方協議而成，以下列出兩種較重要的連線功能：

(A) 同步訊號（Data Mark）

Telnet 以資料標記（DM, 0xF2）命令，來定義它的同步訊號，主要的功能是希望在通訊連線中插入緊急命令時，另一通訊端必需即時處理。當 Telnet 通訊中被插入資料標記時，此時 TCP 連線就將它當成緊急資料處理（Urgent），並會在封包標頭上指引緊急資料的位置（Urgent Point）。

當一端接收到另一端進入緊急模式的通知（DM）時，它就開始讀取資料串列，而捨棄 Telnet 命令以外的所有資料，一直到緊急資料結束（也是以 DM 標示結束），並處理裡面所包含的內容。使用 TCP 緊急模式是為了允許 Telnet 命令能夠跨越連線來傳送。

(B) 客戶端跳脫（Client Escapes）

當客戶端以 Telnet 連線到伺服端之後，也隨時可以跳脫（Escapes）連線，而回到原作業系統底下，但此時該連線並未中斷。一般終端機客戶跳脫都以 Ctrl +]（也都以 ^] 表示）。當連線當中，鍵入客戶跳脫時會回到 telnet> 模式下，便可直接下達 Telnet 的直譯命令。

11-6 Telnet 連線範例

我們在 Linux-1（163.15.2.62）上利用 Telnet 程式，連線到 Linux-2（163.15.2.30），來觀察它們之間協議的情形。首先我們在 Linux-1 主機上執行 Telnet 命令，再設定成顯示所有命令執行情形，執行結果如下：（其中 SEND 表示由客戶端送給伺服端，RCVD 表示接收來自伺服端，隨後括弧內為說明）

```
[tsnien@linux-1 tsnien]$ telnet （執行 telnet 命令，並進入 telnet 工作環境）
telnet> toggle options                （開啟顯示處理情形）
Will show option processing.
telnet> open 163.15.2.30              （連線到 163.15.2.30）
Trying 163.15.2.30...
Connected to 163.15.2.30.
Escape character is '^]'.             （顯示 NVT 跳脫字元）
SENT DO SUPPRESS GO AHEAD （要求對方抑制 Go-Ahead）
SENT WILL TERMINAL TYPE     （要求設定終端機型態）
SENT WILL NAWS        （要求協議視窗大小，Negotiate about window size）
SENT WILL TSPEED              （要求協議終端機速度）
SENT WILL LFLOW         （要求協議本地流量控制，Local Flow Control）
SENT WILL LINEMODE            （要求協議行模式，Line mode）
SENT WILL NEW-ENVIRON       （要求協議傳送環境變數）
SENT DO STATUS                （要求對方的 Telnet 設定狀態）
RCVD DO TERMINAL TYPE       （同意對方設定終端機型態）
RCVD DO TSPEED                （同意對方的終端機速度）
RCVD DO XDISPLOC              （要求設定 X 視窗顯示）
SENT WONT XDISPLOC            （回應不同意 X 視窗顯示）
RCVD DO NEW-ENVIRON           （同意傳送環境變數）
RCVD WILL SUPPRESS GO AHEAD （同意設定抑制 Go-Ahead）
```

```
RCVD DO NAWS                          （同意協商視窗大小）
SENT IAC SB NAWS 0 80 (80) 0 24 (24)  （傳送視窗大小的數值）
RCVD DO LFLOW                         （同意協議本地流量控制）
RCVD DONT LINEMODE                    （不同意協議行模式）
RCVD WILL STATUS                      （同意傳送 Telnet 設定狀態）
RCVD IAC SB TERMINAL-SPEED SEND       （子協議終端機速度）
SENT IAC SB TERMINAL-SPEED IS 9600,9600（子協定終端機速度為 9600）
RCVD IAC SB NEW-ENVIRON SEND          （子協議環境變數）
SENT IAC SB NEW-ENVIRON IS            （同意子協議環境變數）
RCVD IAC SB TERMINAL-TYPE SEND        （子協議終端機型態）
SENT IAC SB TERMINAL-TYPE IS "VT100"  （子協議終端機型態為 VT100）
RCVD DO ECHO                          （協議要求以伺服端的 Echo 模式）
SENT WONT ECHO                        （不同意以伺服端的 Echo 模式）
RCVD WILL ECHO                        （協議以客戶端的 Echo 模式）
SENT DO ECHO                          （同意以客戶端的 Echo 模式）
Red Hat Linux release 7.2 (Enigma)    （伺服端送出顯示文字）
Kernel 2.4.7-10 on an i686
login: tsnien                         （使用者登入）
Password:                             （使用者輸入密碼）
Last login: Mon Jun 17 15:07:24 from linux-1
[tsnien@linux-2 tsnien]$
```

由上述協議中，伺服端傳送 DONT LINEMODE，表示不同意客戶端行模式要求。並且到最後，伺服端要求以本身的 Echo 模式（RCVD DO ECHO），客戶端不同意（SENT WONT ECHO），緊接著，伺服端協議以客戶端的 Echo 模式（RCVD WILL ECHO），客戶端同意以本身 Echo 模式（SENT DO ECHO）。在整個協議當中，已抑制 Go-Ahead 和設定 Echo 模式，因此可以發現雙方已協議以單一字元模式（Single Character Mode）來交談。也可以當使用者登入後，而以跳脫鍵回到 telnet>，再來查詢連線後的狀態，由其中可以觀察到目前所使用的交談模式，查詢範例如下：

```
telnet> status
Connected to 163.15.2.62.
Operating in single character mode    （單一字元模式）
Catching signals locally
Remote character echo                 （客戶端回應 Echo 模式）
Local flow control                    （本地流量控制）
Escape character is '^]'
```

所謂客戶端（或稱遠端）回應（Echo）模式，表示由客戶端鍵盤輸入每一字元，送到伺服器後再回應到客戶端螢幕上顯示，因此，使用者在螢幕上可以看到自己輸入的字元。但是當使用者輸入密碼時，不希望密碼顯示出來，因此，當伺服器送出 "Password:" 時，即關閉 Echo 的功能，直到收到使用 <Enter> 鍵後，再開啟 Echo 的功能，這樣子使用者輸入密碼時就看不到密碼字元。

習題

1. 何謂『**終端機**』（**Terminal**）系統？
2. 請簡述『**網路終端機**』（**Network Terminal**）系統的架構。
3. 請簡述 Telnet 終端機連線架構。
4. 請簡述 Telnet 選項協議的運作程序。
5. 一般 Telnet 連線有哪三種傳操作模式？請分別說明其特性。
6. 請利用您的電腦以 Telnet 連線到主機系統，首先觀察它們之間的操作模式，再將其設定為行模式（Line Mode），並觀察其結果及說明原因。
7. 請利用網路監視器（請參考附錄 A）擷取一個 Telnet 命令的 Ethernet 訊框，並解析訊框標頭的欄位。
8. 同第七題，擷取 IP 封包，並解析 IP 封包標頭欄位的功能。
9. 同第七題，擷取 TCP 封包，並解析 TCP 封包標頭欄位的功能。
10. 同第七題，請擷取終端機和主機協議終端機型態的命令，並分別解析其命令編碼。
11. 同第七題，請擷取主機送出 "Password:" 命令的封包，以及終端機輸入密碼的封包，並觀察是否有 Echo 封包？及說明有關安全性問題。
12. 同第七題，請擷取主機接受使用者名稱及密碼後，回應給終端機訊息的封包。