逢　甲　大　學

資訊工程學系專題報告

實作 Telnet Client

學　　生：　黃 龍 欣（四丁）

指導教授：　黃 溪 春

中華民國九十三年五月

目錄

圖目錄..Ⅲ
表目錄..Ⅳ

第一章　導論..1
1.1 前言..1
1.2 目標..1

第二章　Telnet Protocol..2
2.1 簡介..2
2.2 NVT; Network Virtual Terminal..2
2.3 Telnet Commands..3
2.4 Telnet 選項協議..4
2.5 Telnet 子選項協議..6

第三章　ANSI Escape Sequence..7
3.1 簡介..7
3.2 VT100 簡介..8
3.3 Control Sequence 摘要..9
3.3.1 Cursor Control..9
3.3.2 Erasing Text..10
3.3.3 Set Display Attributes..10
3.3.4 特例..11

第四章　開發環境..12
4.1 概述..12
4.2.1 .NET..13
4.2.2 C#..13
4.3 Visual Studio.Net 2003 之介紹..14

第五章　實作..15
5.1 程式流程..15
5.3 建立連線..17
5.3.1 .NET I/O 方法..17
5.3.2 程式連線處理..18
5.4.3 螢幕輸出處理..26
5.5 鍵盤輸入處理..28

第六章　後記 34
6.1 程式尚未解決之問題 34
6.2 感想 37

附錄 A Telnet 相關之 RFC 40
附錄 B RFC 854 43

圖目錄

圖 1.1 之前做到一半的題目 1
圖 2.1 Telnet Client and Server 2
圖 3.1 沒有使用 ANSI Escape Sequence 時，登入 BBS 的慘狀..... 7
圖 3.2 VT100 終端機... 8
圖 3.3 終端機連線 8
圖 4.1 Visual Studio .NET 2003 14
圖 5.1 主要流程圖.. 15
圖 5.2 連線資訊之取得與設定 16
圖 5.3 接收資料的處理流程 19
圖 5.4 Telnet Commands 狀態遷移圖 20
圖 5.5 ANSI 的處理流程..................................... 23
圖 5.6 ESC # + Final Character 的處理流程................ 24
圖 5.7 ESC + [+ Pn + {;Pn} + Final Character 的處理流程 .. 25
圖 5.8 鍵盤輸入的處理流程 28
圖 5.9 輸入連線位址畫面 31
圖 5.10 Timeout 畫面....................................... 32
圖 5.11 登入到 Knight...................................... 32
圖 5.12 登入到逢甲蒼穹的畫面 33
圖 5.13 關閉 ANSI 處理時的畫面 33
圖 6.1 Windows Telnet 35
圖 6.2 這個程式對於雙屬性中文字無法正常顯示 35

表目錄

表 2.1 Telnet Commands 3
表 2.2 六種可能的選項協議 4
表 2.3 常見的 Option ID.................................. 5
表 3.1 Cursor Control 9
表 3.2 Erasing Text 10
表 3.3 標準屬性 10
表 3.4 前景及背景色 11
表 5.1 Code 對照表.. 29

摘要

Telnet 用於 Internet 的遠程登錄。它可以使用戶坐在已上網的電腦鍵盤前通過網絡進入的另一台已上網的電腦，使它們互相連通。這種連通可以發生在同一房間裡面的電腦或是在世界各地已上網的電腦。習慣上來說，為網路上所有用戶提供服務的電腦稱之為伺服器,而自己在使用的電腦稱之為客戶端.一旦連線後，客戶端可以享有伺服器所提供的一切服務，用戶可以進入很多的特殊的伺服器如尋找圖書索引。網上不同的主機提供的各種服務都可以被使用.

現在瀏覽器好用又漂亮，但是小而實用的 Telnet，有時候還是能幫得上大忙，尤其是在校園中，BBS 仍然風行，BBS 站提供了很多的遊戲、虛擬金幣之類的功能，吸引很多年輕學子掛站流連，所以 Telnet 目前看起來還是有存在的必要。

第一章 導論

1.1 前言

剛開始我是打算要做一個像 KKman 一樣結合 WWW 以及 BBS 功能的軟體，只不過當我花了很多時間完成整個程式的主結構以及 WWW 基本功能之後，我才發現要 BBS 部分有困難，最主要困難就是在畫面的輸出，因為在 ANSI 的實作，需要控制文字的前景與背景色彩，除此之外還需要控制游標的位置，我在.net 提供的元件中找不到一個能夠符合我的需求，但是，不使用元件的話，這部分就必須使用 GDI 來實作，我有嘗試要使用 GDI，但是我發現在書店裡根本就沒有專門介紹 GDI 的書籍，部分的.net 書籍中會有提到 GDI+，但是頁數都很少，大概都只是簡單介紹一下而已，所以我就決定放棄原來的題目，然後改用主控台模式來實作 Telnet 協定。

圖 1.1 就是之前的題目的半成品。

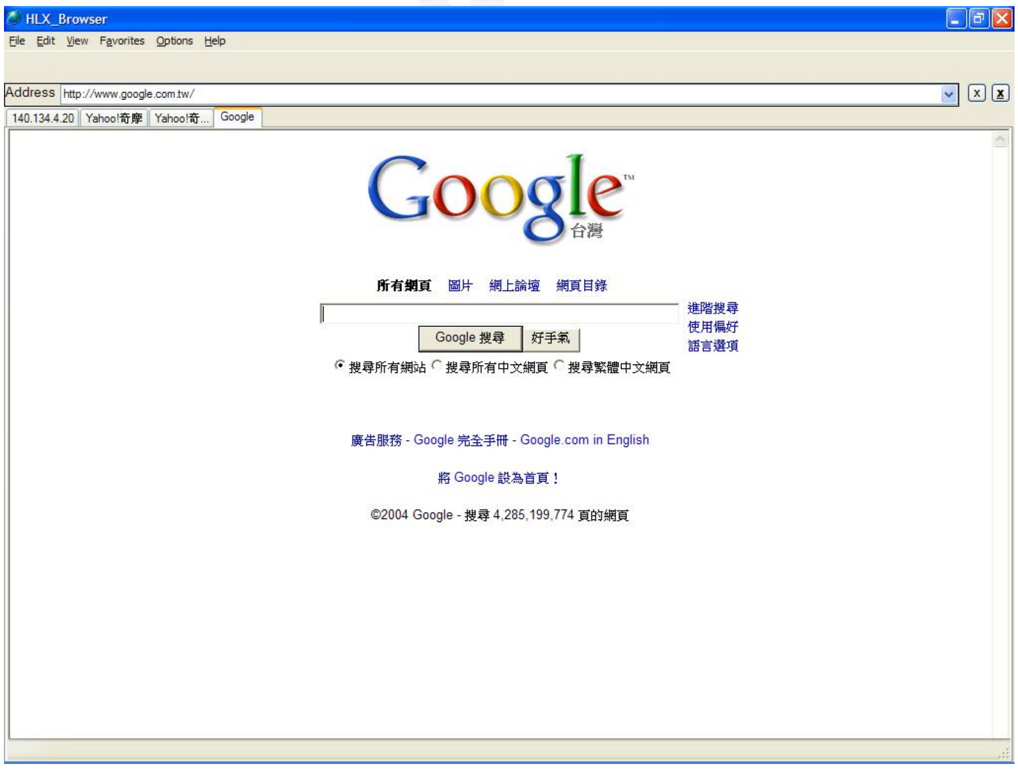

HLX_Browser
File Edit View Favorites Options Help
Address http://www.google.com.tw/
140.134.4.20
Yahoo!奇摩
Yahoo!奇...
Google
Google 台灣
所有網頁 圖片 網上論壇 網頁目錄
進階搜尋
使用偏好
語言選項
Google 搜尋
好手氣
搜尋所有網站 搜尋所有中文網頁 搜尋繁體中文網頁
廣告服務 - Google 完全手冊 - Google.com in English
將 Google 設為首頁！
©2004 Google - 搜尋 4,285,199,774 頁的網頁

圖 1.1 之前做到一半的題目

1.2 目標

首先是要實作 telnet 的協定，亦即實作 RFC 854，然後再處理 ANSI Escape Control Sequence，讓這個程式能夠登入到一般的 BBS 站台。

第二章 Telnet Protocol

2.1 簡介

Telnet 是 TCP/IP的終端機協定，它主要是用來規範不同的電腦系統，經由網路連線到遠端主機進行線上作業的標準。由於網路上連接著各種不同的電腦系統，因此要進入這些不同的電腦使用其提供的資源時，時常會造成連接上的困擾，而Telnet 協定就是為了解決這樣的困擾而被發展出來的。

2.2 NVT; Network Virtual Terminal

一旦一個TELNET連接建立後，連線的兩端被假設為在一個Network Virtual Terminal上開始和終止操作。NVT是一個想像中的標準設備通用設備的代表。這消除了 Server 和 Client 要了解對方機器的特點。所有的主機，包括用戶和伺服器，把他們設備屬性和協定映射為就像一個在網路上的NVT，而且每一方都可以假設對方也有一個類似的映射。圖2.1為連線的示意圖。

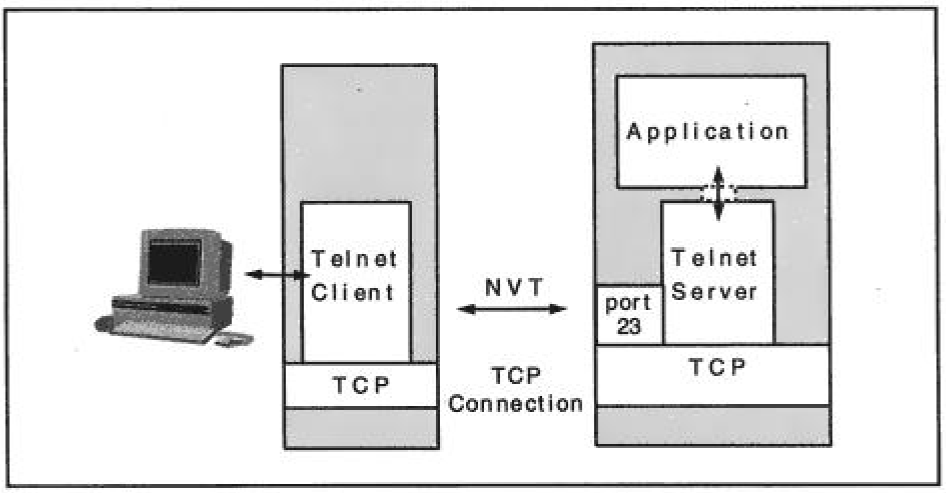


圖2.1 Telnet Client and Server

2.3 Telnet Commands

所有的 Telnet 命令至少包含一個兩個位元組的序列：
所有的命令都必須以IAC(Interpret as Command)開頭，處理選項協議的命令有三個位元組序列，第三個位元組就成了被選項引用的代碼。之所以選擇這種格式，是這種格式能夠更有效地利用"資料空間"---當然，是通過基本NVT的協議來進行。資料位元組與保留的命令值的衝突被大大減少了。表2.1為常見的Command的列表。

➢ Commands 使用格式：

IAC　Command_Code

例：　IAC　NOP　（255　241）

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

表2.1 Telnet Commands

2.4 Telnet 選項協議

- 選項協議使用格式：

  IAC　WILL(DO/WONT/DONT)　OptionID

說明：

- WILL : 發送者期望選項啟動
- DO : 接收者同意對方選項
- WONT : 發送者期望停用自己的選項
- DONT : 接收者不同意對方選項

	發送者		接收者	說　明
1.	WILL	→		發送者希望致能選項
		←	DO	接收者同意
2.	WILL	→		發送者希望致能選項
		←	DONT	接收者不同意
3.	DO	→		發送者希望對方致能選項
		←	WILL	接收者同意
4.	DO	→		發送者希望對方啟動選項
		←	WONT	接收者不同意
5.	WONT	→		發送者希望停用自己的選項
		←	DONT	接收者同意
6.	DONT	→		發送者要求停用對方的選項。
		←	WONT	接收者同意

表2.2 六種可能的選項協議

表2.2列出來的是選項協議中所有可能出現的情況，而表2.3是常見的選項列表

代碼	名稱
1	echo
3	suppress go ahead
5	status
6	timing mark
24	terminal type
31	window size
32	terminal speed
33	remote flow control
34	linemode
36	environment variables

表2.3 常見的Option ID

➢ 使用範例：

當網路忙碌，使用者可能希望由本機來回應輸入的字，這時候就送出：

IAC DONT ECHO

這是要求Server不要做echo，由本機自己做。而Server則會回應：

IAC WONT ECHO

這是代表Server同意關閉Echo的功能

2.5 Telnet 子選項協議

- 選項協議使用格式：

  IAC　SB … … SE

使用範例：

```
(1)Server：IAC WILL  TERMTYPE
(2)Client：IAC DO    TERMTYPE
(3)Server：IAC SB    TERMTYPE 1  IAC   SE
(4)Client：IAC SB    TERMTYPE 0  VT100 IAC   SE
```

說明：

(1)　伺服器要求客戶端告知終端機類型

(2)　客戶端同意要求

(3)　伺服器告訴客戶端把終端機類型傳送過來(1:Send)

(4)　客戶端告訴伺服器端：終端機類型為VT100(0:Is)

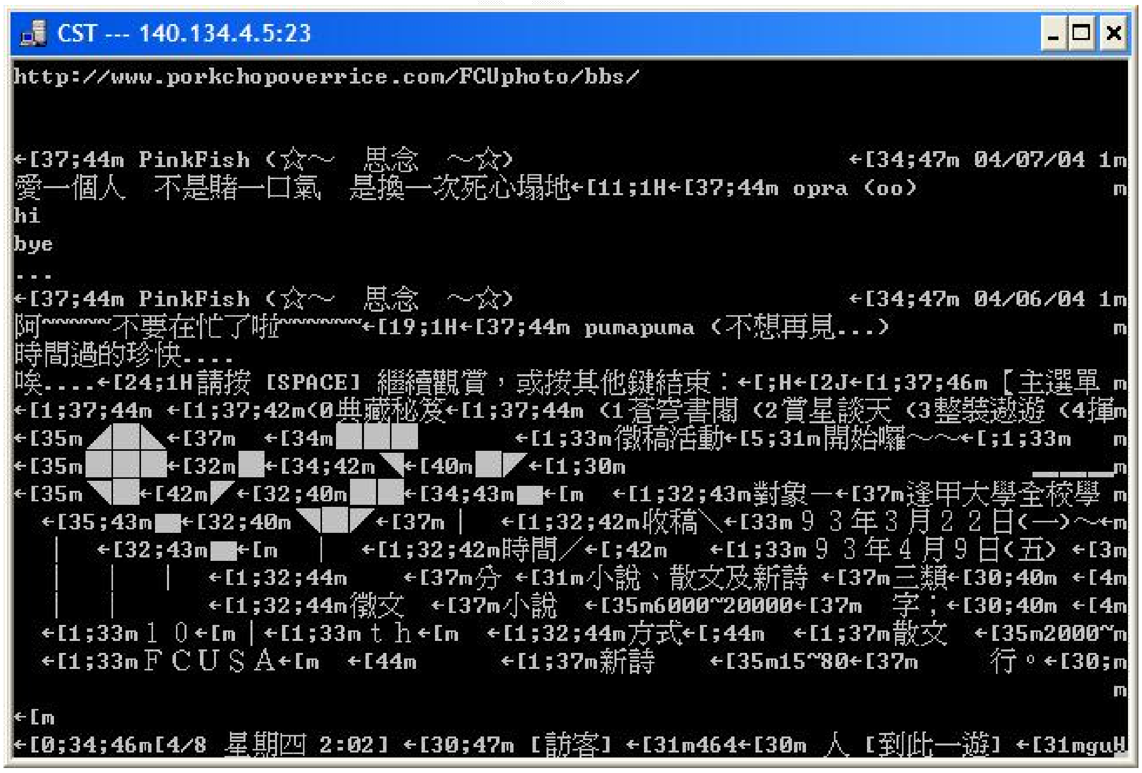
第三章 ANSI Escape Sequence

3.1 簡介

單純的Telnet Protocol(RFC 854)對於要登入一般的Unix主機已經足夠，但是如果要使用Vi/Pico或者是要telnet到BBS站台，還必需配合ANSI Escape Sequence來控制游標、色彩以及清除螢幕，否則就會像圖3.1一樣慘不忍睹。

因為完整的ANSI Escape的標準(ANSI Standard (X3.64) Control Sequences)太過複雜，而VT100的escape sequences是根據ANSI的標準來設計的，所以在程式裡我直接參照VT100的標準來做處理。

圖3.1是沒有處理Escape Sequence時，連線到BBS站台的情形。



```
http://www.porkchopoverrice.com/FCUphoto/bbs/


←[37;44m PinkFish (☆~  思念  ~☆)                            ←[34;47m 04/07/04 1m
愛一個人  不是賭一口氣  是換一次死心塌地←[11;1H←[37;44m opra (oo)                  m
hi
bye
...
←[37;44m PinkFish (☆~  思念  ~☆)                            ←[34;47m 04/06/04 1m
阿~~~~~~~不要在忙了啦~~~~~~~~←[19;1H←[37;44m pumapuma (不想再見...)                m
時間過的珍快....
唉....←[24;1H請按 [SPACE] 繼續觀賞，或按其他鍵結束：←[;H←[2J←[1;37;46m [主選單 m
←[1;37;44m ←[1;37;42m(0典藏秘笈←[1;37;44m (1蒼穹書閣 (2賞星談天 (3整裝遨遊 (4揮m
←[35m ◢██◣←[37m  ←[34m███        ←[1;33m徵稿活動←[5;31m開始囉~~←[;1;33m    m
←[35m███←[32m█←[34;42m ◥←[40m█◤←[1;30m                                      ___m
←[35m ◥█ ←[42m◤←[32;40m██←[34;43m█←[m  ←[1;32;43m對象一←[37m逢甲大學全校學 m
  ←[35;43m█←[32;40m ◥█◤←[37m │  ←[1;32;42m收稿╲←[33m９３年３月２２日(一)~←m
  │  ←[32;43m█←[m    │  ←[1;32;42m時間╱←[;42m   ←[1;33m９３年４月９日(五) ←[3m
  │   │   │   ←[1;32;44m    ←[37m分 ←[31m小說、散文及新詩 ←[37m三類←[30;40m ←[4m
  │   │       ←[1;32;44m徵文  ←[37m小說  ←[35m6000~20000←[37m   字；←[30;40m ←[4m
  ←[1;33m１０←[m│←[1;33mｔｈ←[m  ←[1;32;44m方式←[;44m  ←[1;37m散文   ←[35m2000~m
  ←[1;33mＦＣＵＳＡ←[m  ←[44m      ←[1;37m新詩     ←[35m15~80←[37m      行。←[30;m
                                                                                m
←[m
←[0;34;46m[4/8 星期四 2:02] ←[30;47m [訪客] ←[31m464←[30m 人 [到此一遊] ←[31mguH
```

圖3.1 沒有使用ANSI Escape Sequence時，登入BBS的慘狀

3.2 VT100簡介

當ANSI 的X3委員會出版X3.64 資料(在1977年后期)時， 一些終端機製造商，堅決要求使用分離的邏輯設計，因為標準太過複雜，使他們不能實作。

但是DEC的實作：VT100，實現了使用一個通用微處理器(8 位的Intel 8080)去解譯控制碼的新想法。雖然它並非實現ANSI X3.64標準的第一個終端機，但VT100頗受歡迎並且不久後就成為被廣泛模仿的終端機。圖3.2就是VT100終端機，圖3.3是終端機連線的示意圖。

圖3.2 VT100終端機

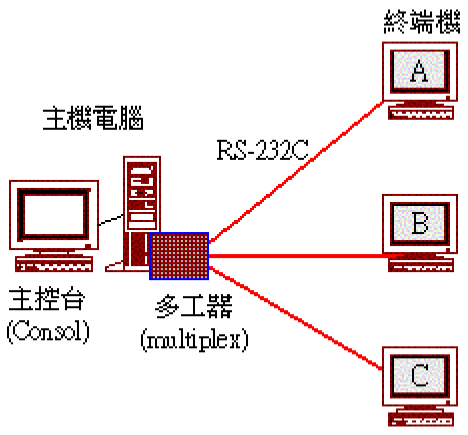


圖3.3 終端機連線

3.3 Control Sequence 摘要

3.3.1 Cursor Control

Cursor up	ESC [Pn A
Cursor down	ESC [Pn B
Cursor forward (right)	ESC [Pn C
Cursor backward (left)	ESC [Pn D
Direct cursor addressing	ESC [Pl ; Pc H ESC [Pl ; Pc f
Index	ESC D
New line	ESC E
Reverse index	ESC M
Save cursor and attributes	ESC 7
Restore cursor and attributes	ESC 8
Cursor up	ESC [Pn A

表3.1 Cursor Control

ESC 為十進位的27，Pn為參數，例：

27 49 65 ➔ ESC 1 A

這一行就是代表要把游標往上移一行

比較需要注意的是Direct Cursor Addressing中的Pl(l:Line)代表要移到第幾列，為y座標，而Pc(c:column)代表要移到第幾行，是為x座標，所以 ESC 1;2 H 是要把游標移到座標(2,1)而不是(1,2)。

表3.1是常見的游標控制代碼。

3.3.2 Erasing Text

From cursor to end of line	ESC [K ESC [0 K
From beginning of line to cursor	ESC [1 K
Entire line containing cursor	ESC [2 K
From cursor to end of screen	ESC [J ESC [0 J
From beginning of screen to cursor	ESC [1 J
Entire screen	ESC [2 J

表3.2 Erasing Text

表3.2為常見的清除螢幕的代碼。

3.3.3 Set Display Attributes

格式：<ESC>[{attr1};...;{attrn}m

例：
ESC 41m → 將背景色設定為紅色
ESC 32;41m → 設定為紅底綠字

所有的設定均是對後來所顯示的文字才有作用，而且屬性一直維持，直到下次設定為其他屬性為止。
表 3.3 及表 3.4 為一般常見之文字前背景的屬性。

0	重設所有屬性
1	高亮度(針對前景)
4	底線(針對前景)
5	閃爍(針對前景)
7	反轉(前景與背景色交換)
8	不顯示

表 3.3 標準屬性

前景色彩		背景色彩	
30	Black	40	Black
31	Red	41	Red
32	Green	42	Green
33	Yellow	43	Yellow
34	Blue	44	Blue
35	Magenta	45	Magenta
36	Cyan	46	Cyan
37	White	47	White

表 3.4 前景及背景色

3.3.4 特例

當處理 Escape Sequence 時，如果遇到 ESC(27)、CAN(24)、SUB(26)時，則立即放棄目前的 Control Sequence。

第四章 開發環境

4.1 概述

C#(C Sharp)是微軟在.NET 平台上新開發的語言，當 C#還未正式推出之前，就有不少評論說 C#都是抄襲 Java、了無新意，所以剛開始我對 C#就沒什麼好印象，但是經過一段時間，C#與.NET 的資訊越來越豐富，我發覺.NET 的概念其實蠻吸引人的，我開始去試著了解.NET 以及 C#。

而大三時的微軟巡迴講座，則使得我決定放棄 Java，投靠 C#陣營。所以這個專題，我之所以會決定使用 C#，倒不是因為.NET 跨平台或者是跨語言的關係，而是因為我慣於使用 C#。而程式的開發環境則是使用 Microsoft Visual Studio.NET 2003，當然，這也只是習慣，畢竟開發主控台模式的程式，IDE 大部份的功能是派不上用場的。

4.2 C# 與 .NET 平台簡介

4.2.1 .NET

.NET 是 Microsoft 為 XML Web 服務所提供的平台。XML Web 服務可讓多個應用程式透過 Internet 彼此通訊並共用資料，不論其作業系統或程式語言為何。

Microsoft 相信目前潮流正朝分散式運算轉型。由於過去幾年人們不斷佈設管線的結果，頻寬的限制已不如以往那般嚴苛。目前已有許多採分散式架構的應用程式，例如 Napster 應用程式就是使用 Rich Client 與分散式目錄服務交談，並將網路上所有連線的電腦當成伺服器來使用。分散式應用程式的另一個例子就是立即訊息，您可以在網路上使用 Rich Client 與分散式朋友清單交談，並與其他 Rich Client（例如 Instant Messenger 及 Windows）進行通訊。.NET 的目的是加速下一代分散式運算的發展。

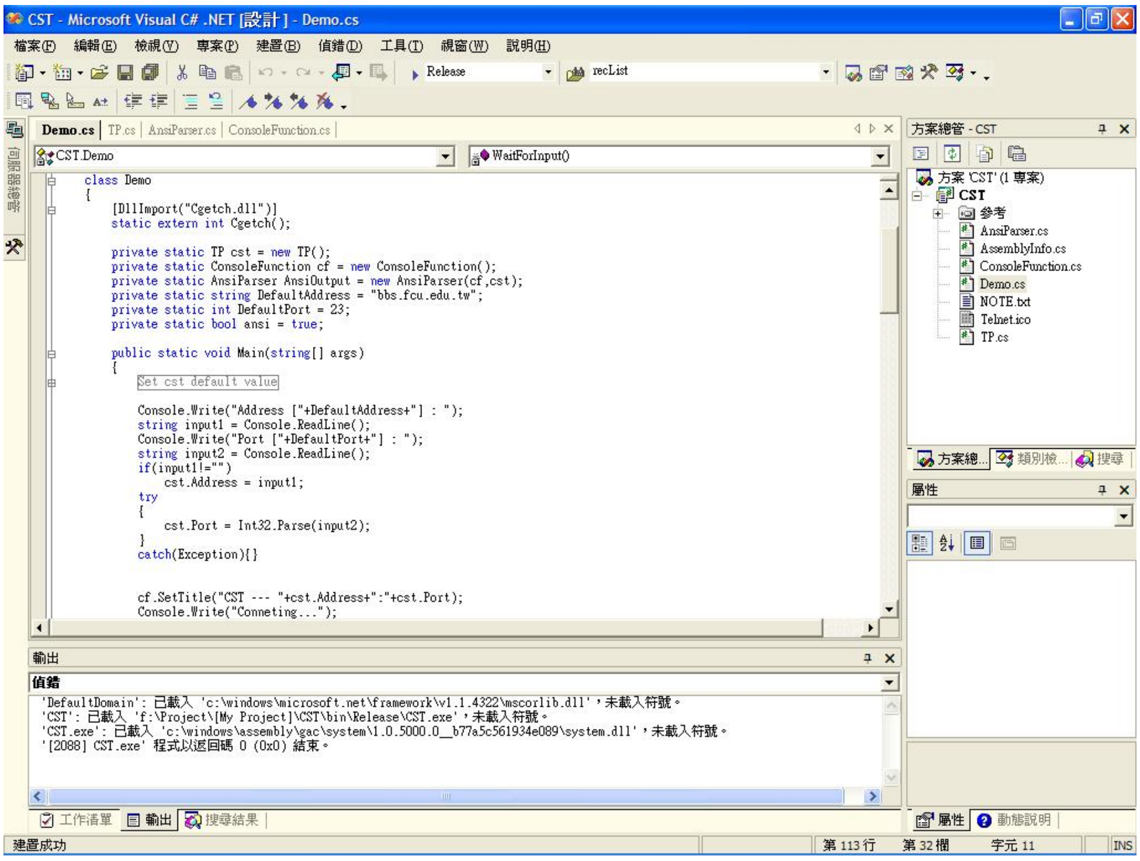
4.2.2 C#

過去二十多年來，C 與 C++ 一直是開發商用與商務軟體最廣為使用的語言。雖然這兩種語言為程式設計人員提供了非常大量的精密控制，然而這樣的彈性卻很耗用生產成本。和像 Visual Basic 這樣的語言相比，相同的 C 與 C++ 應用程式常常要花上更長的時間來開發。因為這些語言的複雜性與很長的循環時間，許多 C 與 C++ 的程式設計人員一直在尋找能在功能強與有生產力之間提供較佳平衡的語言。

Microsoft 對此問題的解決方案是稱為 C#的語言。C# 是一種先進的、物件導向的語言，可讓程式設計人員替新的 Microsoft .NET 平台快速建置各種應用程式，並提供完全利用運算與通訊功能的工具與服務。因為 C# 有精緻的物件導向設計，對於架構各種不同元件而言都是極佳的選擇。使用簡單的 C# 語言結構，這些元件可以轉換成 XML Web Services，並可以透過 Internet 從任何作業系統上執行的任何語言來叫用這些元件。不只這樣，C# 是專門為 C++ 程式設計人員所設計的，讓他們可以不必犧牲 C 與 C++ 長久以來的功能與控制特性，就能擁有快速開發的能力。因為 C# 繼承了這樣的特性，所以它和 C 與 C++ 之間保有高度的相似性。熟悉這些語言的開發人員將可以很快的上手，運用 C# 來提高生產力。

4.3 Visual Studio.Net 2003 之介紹

Visual Studio .NET 是用來快速建置與整合 XML Web services 和應用程式的廣泛開發工具。 Visual Studio .NET 提供一個高效率環境， Visual Studio .NET 使用安全、高效能的 Microsoft .NET Framework Run-Time 環境，可提供強大的工具來設計、建立、測試和部署 XML Web services 與應用程式。圖 4.1 為 Visual Studio.NET 2003 的執行畫面。



```
class Demo
{
    [DllImport("Cgetch.dll")]
    static extern int Cgetch();

    private static TP cst = new TP();
    private static ConsoleFunction cf = new ConsoleFunction();
    private static AnsiParser AnsiOutput = new AnsiParser(cf,cst);
    private static string DefaultAddress = "bbs.fcu.edu.tw";
    private static int DefaultPort = 23;
    private static bool ansi = true;

    public static void Main(string[] args)
    {
        Set cst default value

        Console.Write("Address ["+DefaultAddress+"] : ");
        string input1 = Console.ReadLine();
        Console.Write("Port ["+DefaultPort+"] : ");
        string input2 = Console.ReadLine();
        if(input1!="")
            cst.Address = input1;
        try
        {
            cst.Port = Int32.Parse(input2);
        }
        catch(Exception){}


        cf.SetTitle("CST --- "+cst.Address+":"+cst.Port);
        Console.Write("Conneting...");
```

```
'DefaultDomain': 已載入 'c:\windows\microsoft.net\framework\v1.1.4322\mscorlib.dll'，未載入符號。
'CST': 已載入 'f:\Project\[My Project]\CST\bin\Release\CST.exe'，未載入符號。
'CST.exe': 已載入 'c:\windows\assembly\gac\system\1.0.5000.0__b77a5c561934e089\system.dll'，未載入符號。
'[2088] CST.exe' 程式以返回碼 0 (0x0) 結束。
```

圖 4.1 Visual Studio .NET 2003

第五章 實作

5.1 程式流程

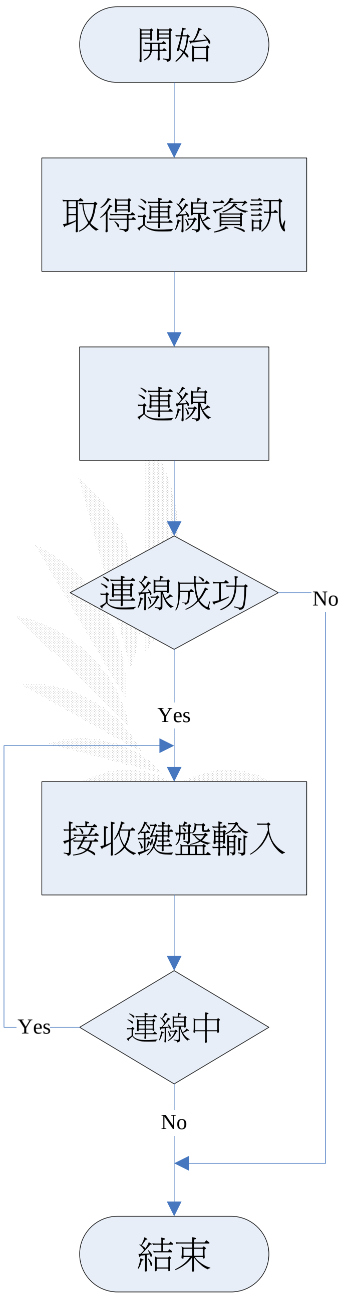


圖 5.1 主要流程圖

程式起始之後，會先要求使用者輸入 address 以及 port，然後做連線處理，接下來就進入迴圈接收並處理使用者的輸入。詳細的實作細節在下面的各個小節會說明。圖 5.1 是程式的主流程。

5.2 連線位址的取得與設定

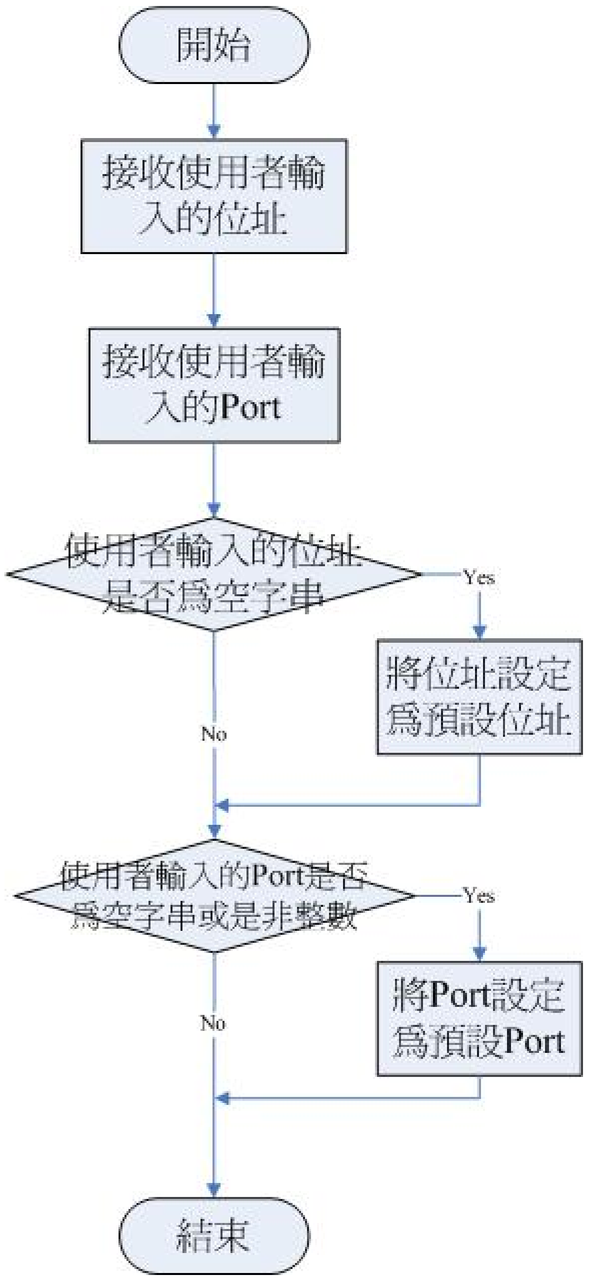


圖 5.2 連線資訊之取得與設定

一開始會在螢幕輸出提示字串告知使用者輸入 address，然後再要求使用者輸入 port，在提示字串後面會有預設值，使用者可直接按 Enter 鍵接受預設值。使用者輸入完，程式就會判斷接收到的值，如果是空字串，就不做設定的動作，直接使用預設值(address 的預設值是:bbs.fcu.edu.tw；port 的預設值是 23)，如果使用者有輸入 address 的話，就把程式裡面變數的值設定為使用者輸入的值，在 port 的部份，會先把使用者輸入的值透過 Int32.Parse 轉成整數，如果程

式產生 exception，代表使用者輸入之資料形態不合法，則繼續使用預設值 23，如果沒產生 exception，就把程式裡的變數設定為輸換成功的整數。圖 5.2 是連線資訊取得與設定的流程。

5.3 建立連線

5.3.1 .NET I/O 方法

.NET Sockets 透過三種基本 I/O 方法來管理 socket 的資料和連線：封鎖式、select(選取)與非同步。

封鎖式 I/O

封鎖式 I/O 是最容易使用的模型。每當呼叫.NET Sockets 的 I/O 繫結方法像是 Receive 時，如果接收的 socket 上沒有擱置的資料，呼叫便只會執行封鎖。假如應用程式必須執行其他作業或另外處理其他 socket 要求，應用程式必須再額外建立執行緒。假如應用程式只是處理單一連線的簡單型用戶端，封鎖式 socket 就是最好的 I/O 模型。

select I/O

select 模型是廣泛運用在 Winsock 中的另一種 I/O 模型，因為集中使用 select 函數來管理 I/O 而被稱為 select 模型。此模型最初是針對以 Unix 為主、實作 Berkeley socket 的電腦而設計的。Select 模型目前已併入 Winsock 1.1，如果應用程式不想在 socket 呼叫中使用封鎖，則可改用此模型以便有組織的管理多重 socket。由於 Winsock 1.1 回溯相容於 Berkeley socket 實作，使用 select 函數的 Berkeley socket 應用程式就技術上而言應該不必修改就可執行。

由於封鎖式 I/O 在管理多重 socket 方面有所限制，.NET Sockets 提供了類似 Winsock 1 Select API 的 Select 方法，可經由單一執行緒管理多重 socket I/O。

非同步 I/O

.NET Sockets 的非同步模型最適合管理來自一個或多個 socket 的 I/O。這是三種模型當中最有效率的一種。當等待網路作業完成時，非同步的用戶端通訊端不會暫止應用程式。反而，當應用程式還繼續在原始執行緒上執行時，它會使用標準 .NET Framework 非同步程式撰寫模型 (Programming Model) 來處理一個執行緒上的網路連接。非同步通訊端很適合大量利用網路或在繼續執行之前無法等待網路作業完成的應用程式。

5.3.2 程式連線處理

這個程式是使用非同步 I/O 模型。

程式首先建立 socket 以及 IPEndPoint 物件，並設定適當之參數，然後利用 AsyncCallback(ConnectCallback)，將回呼委派給 ConnectCallback 這個函式做處理。然後呼叫 BeginConnect 開始連線作業，同時啟動 timer 物件，並呼叫 WaitOne()封鎖目前的執行緒。在下面兩種情況下，執行緒才會解除封鎖：

1. 當 BeginConnect 作業完成時，會呼叫 ConnectCallback 這個函式，如果連線成功，則解除目前執行緒的封鎖，並停止 timer 物件。接下來再一次利用 AsyncCallback 方法把接收資料的事件委派給 OnRecievedData 函式，接下來就呼叫 BeginReceive 開始接收資料。如果連線失敗的話，就結束程式。

2. 如果超過 10 秒，而 BeginConnect 作業還沒完成，timer 就會呼叫 connectTimer_Elapsed 函式，解除目前執行緒的封鎖，並顯示 timeout 的訊息，並結束程式。

5.4 接收資料的處理

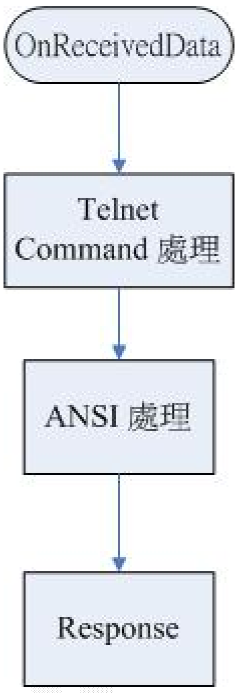


圖 5.3 接收資料的處理流程

圖 5.3 為接收到資料時，程式所進行的處理。

當 socket 物件接收到資料的時候，便會呼叫 OnReceivedData 函式處理接收到的資料。

首先會先呼叫 Telnet Commands 的處理函理針對 command 做適當之處理，然後將剩餘的資料交給 ANSI 處理函式做輸出，接下來再針對 Telnet 的 Commands 做出回應傳給 Server。

相關的細節在下面的各個小節會說明。

5.4.1 Telnet

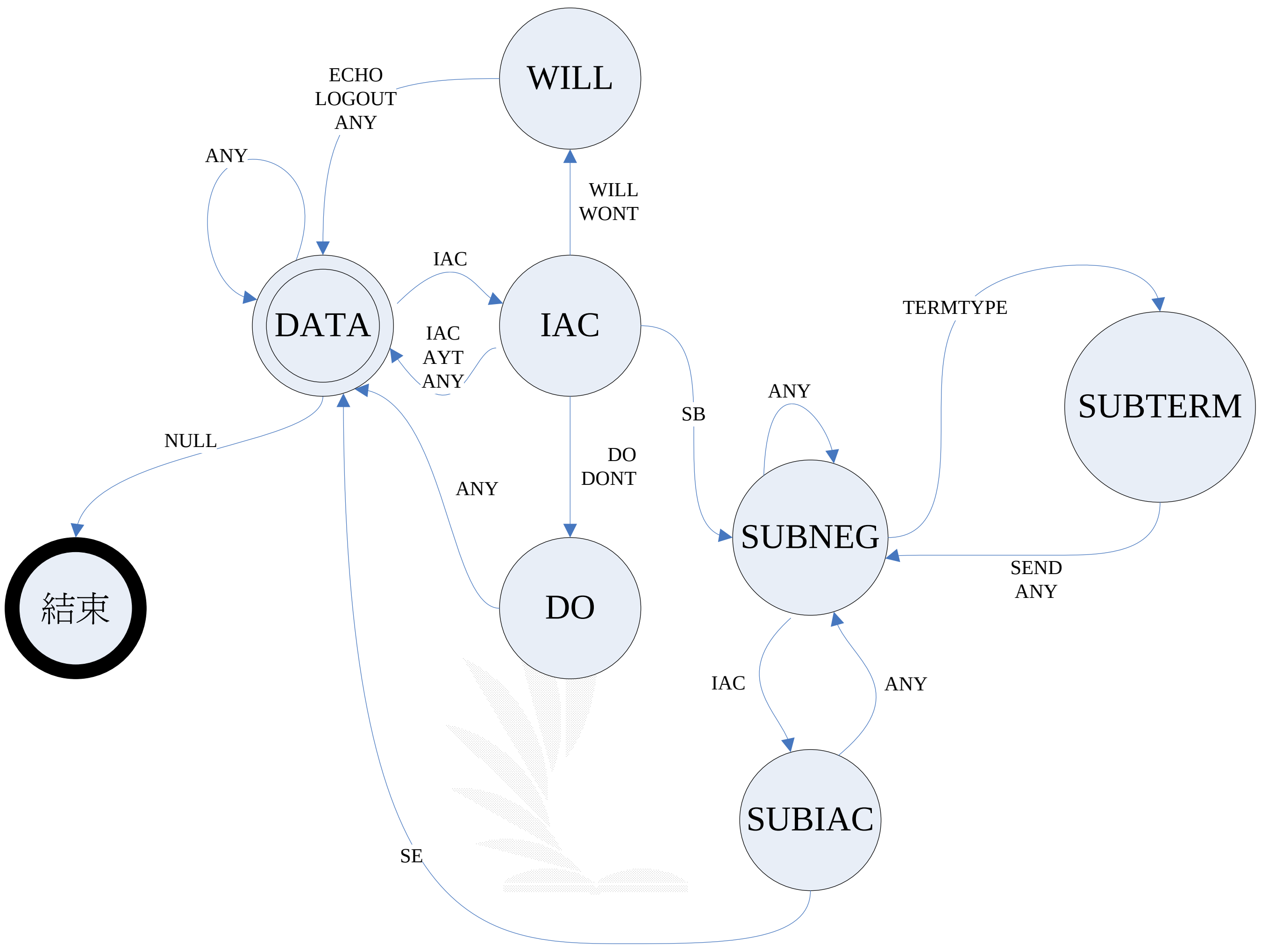


圖 5.4 Telnet Commands 狀態遷移圖

圖 5.4 是接收到 Telent commands 時所進行的處理。

DATA 為初始狀態，ANY 代表其他未指定之字元，以下將針對各個狀態進行詳細分明：

- state DATA：
  從 Buffer 讀取一個字元，如果是：
  IAC(255) → 轉移到 state IAC。
  NULL(Buffer 已經讀取結束)：流程結束。
  ANY → 停留在 state DATA，把字元放進字串變數 output。

- state IAC：
  從 Buffer 讀取一個字元，如果是：

DO(253) or DON’T(254) → 轉移到 state DO，將字元儲存到變數 OptionCmd。
WILL(251) or WONT(252) → 轉移到 state WILL，將字元儲存到變數 OptionCmd。
SB(250) → 轉移到 state SB。
IAC(255) → 轉移到 state DATA，把字元放進字串變數 output。
AYT(246) → 轉移到 state DATA，把'a'(任意可視字元皆可)放進字串變數 SendStr。
ANY →轉移到 state DATA。

- state WILL：
  從 Buffer 讀取一個字元，如果是：
  ECHO(1) →
    a. OptionCmd == WILL：將布林變數 doecho 設為 true，然後在字串變數 SendStr 中加入：IAC DO ECHO，轉移到 state DATA。
    b. OptionCmd == WONT：將變數 doecho 設為 false，然後在字串變數 SendStr 中加入：IAC DON’T ECHO，轉移到 state DATA。

  LOGOUT(18) → 在變數 SendStr 中加入：IAC DON’T LOGOUT，這是避免太久沒動作而被 Server 踢掉。然後轉移到 state DATA。

  ANY →在變數 SendStr 中加入：IAC DON’T ch，ch 為讀取進來的字元，這是意謂著不支援或不認識這個選項。然後轉移 state DATA。

- state DO：
  從 Buffer 讀取一個字元，如果是：
  TERMTYPE(24) →在變數 SendStr 中加入：IAC WILL TERMTYPE，這是同意傳送 termtype 的類型給 Server。接下來轉移到 state DATA。

  ANY →在變數 SendStr 中加入：IAC WONT ch，ch 為讀取進來的字元，這是意謂著不支援或不認識這個選項。然後轉移 state DATA。

- state SUBNEG：
  從 Buffer 讀取一個字元，如果是：
  TERMTYPE(24) → 轉移到 state SUBTERM。
  IAC(255) → 轉移到 state SUBIAC。
  ANY → 狀態不變。

- state SUBIAC：
  從 Buffer 讀取一個字元，如果是：
  SE(240) → 轉移到 state DATA。
  ANY → 轉移到 state SUBNEG。

- state SUBTERM：
  從 Buffer 讀取一個字元，如果是：
  SEND(1)：在變數 SendStr 中加入：
  IAC SB TERMTYPE IS “VT100” SE IAC，
  轉移到 state SUBNEG。

5.4.2 ANSI

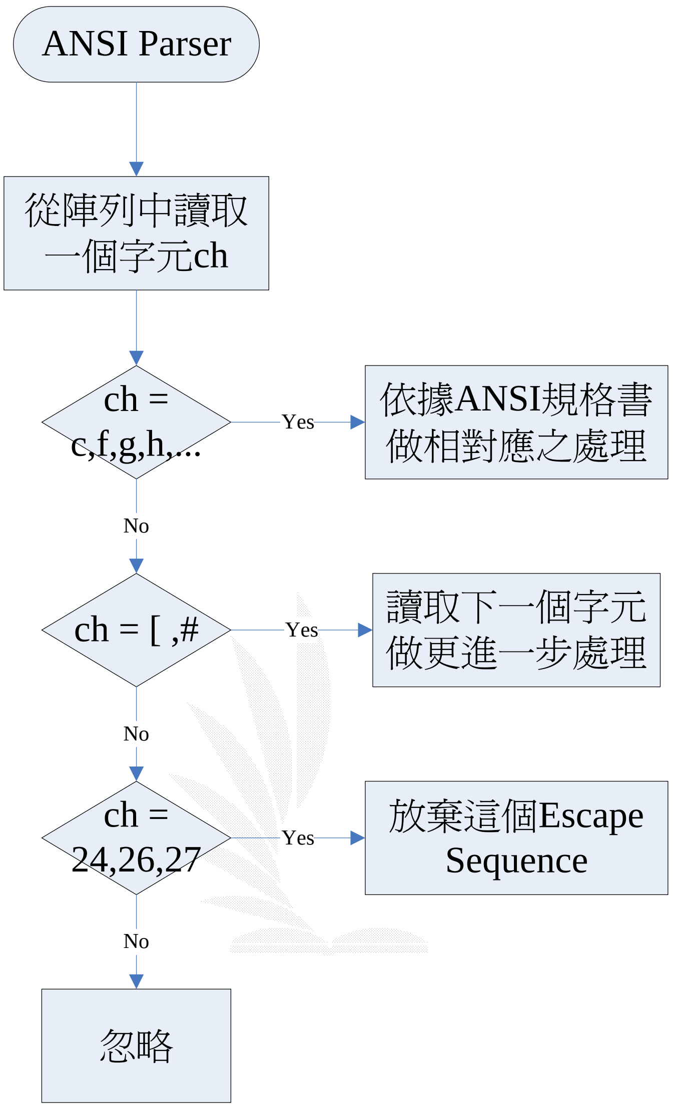


圖 5.5 ANSI 的處理流程

圖 5.5 是處理 Escape Sequence 的流程。

在 ANSI 處理流程中，是先依參數個數將 Escape Sequence 歸成四種類型再做處理：

1. ESC + Final Character：

   例如：ESC A → 將游標上移一行

   ESC B → 將游標下移一行

   這一類的只要依照其給定之 Final Character 在 ANSI 規格書中找出對映之動作並執行即可。

2. ESC + [(or #) + Final Character：

   例如：ESC [K → 清除游標該行，在游標之後的字元

   ESC [J → 清除螢幕中在游標之後的字元

這一類同樣也只要依照其給定之 Final Character 在 ANSI 規格書中找出對映之動作並執行就可以了。
圖 5.6 是針對以 ESC # 啟始的 Sequence 所做的處理。

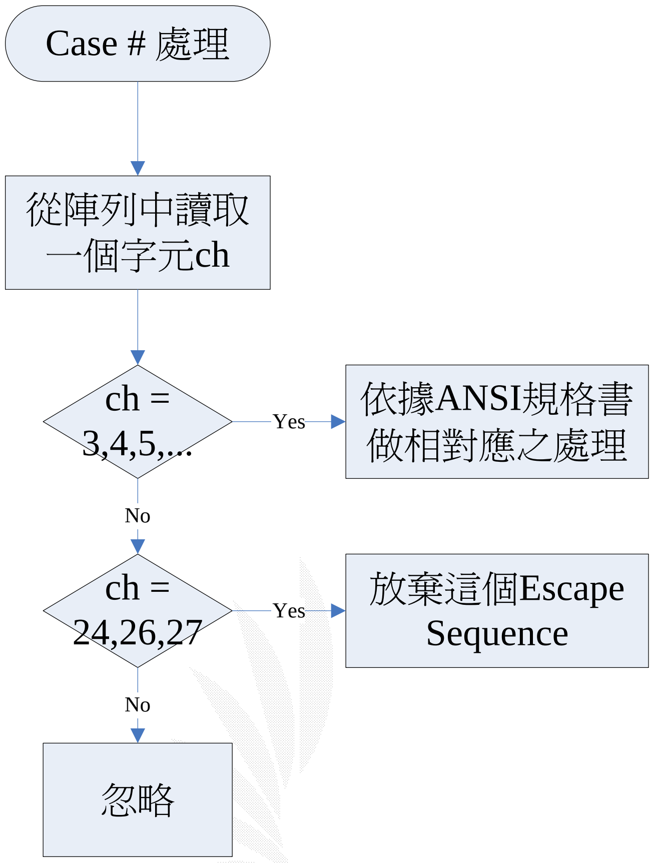


圖 5.6 ESC # + Final Character 的處理流程

3. ESC + [+ Pn + Final Character：
   例如：ESC [1k → 清除從行首到游標的字元
   這一類同樣除了要處理 Final Character，還要考慮到 Pn 所代表的意義。

4. ESC + [+ Pn + {;Pn} + Final Character：
   例如：ESC [1;31m → 設定前景色為高亮度的紅色
   這一類的參數個數未知，所以要把字元讀取逐一讀取並記錄，一直到讀取到 Final Character 為止。然後再從記錄下來的字串依據分號的位置將參數逐一讀取出來，再依照 Fianl Character 做適當之處理。

● 在任何時候只要遇到 CAN(24)、SUB(26)、ESC(27)就立即放棄目前的 Escape Sequence。

圖 5.7 是針對以 ESC [為起始的 Sequence 所做的處理。

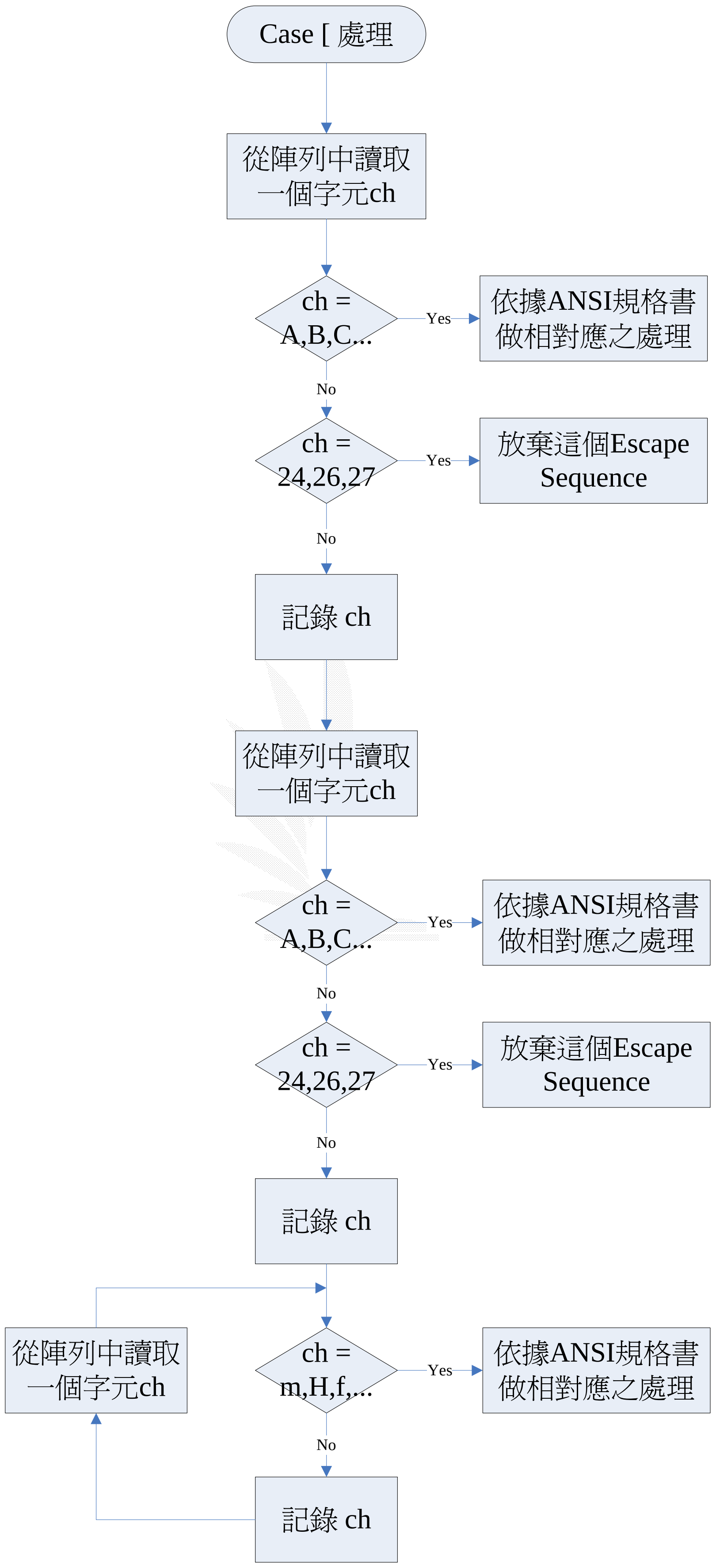


圖 5.7 ESC + [+ Pn + {;Pn} + Final Character 的處理流程

5.4.3 螢幕輸出處理

螢幕輸出的處理，.net framework 本身支援的功能並不多，所以在程式中輸出的功能，大部分都是由 Windows API 所組合而成。

在 C#中引用 Windows API 的方法：

- 首先要先引用 namespace：
  `using System.Runtime.InteropServices;`

- 接下來指定引用的 dll 檔：
  `[DllImport("kernel32.dll")]`

- 最後要指定名稱：
  `extern int GetStdHandle(int nStdHandle);`
  如果不特別指定，預設會把名稱當成進入點，如果要自己設定比較好記的名稱，則必須指定引用的 dll 檔的檔名時順便指定進入點：
  `[DllImport("kernel32.dll", EntryPoint=" GetStdHandle")]`

這個方法不只是能引用 Windows API 而已，當然也可以引用自己開發的 dll 檔。

以下是程式中使用的 API：

GetStdHandle：取得標準輸入、輸出裝置的 Handle 值。
GetConsoleScreenBufferInfo：取得螢幕緩衝區的資訊。
SetConsoleTextAttribute：設定文字前景及背景的屬性。
SetConsoleCursorPosition：設定游標屬性。
FillConsoleOutputAttribute：以特定屬性之字元填滿指定區域
SetConsoleScreenBufferSize：設定螢幕緩衝區的大小。
ExitProcess：結束 Console 視窗。
SetConsoleTitle：設定 Console 視窗的標題列。
SetConsoleMode：設定輸入或輸出緩衝區輸入或輸出的屬性。
GetConsoleMode：取得輸入或輸出緩衝區輸入或輸出的屬性。

所有關於螢幕輸出的函式都位於類別 ConsoleFunction 裡。
以下是函式的簡略說明：

[文字屬性]

TextColor：設定前景顏色。
BackgroundColor：設定背景顏色。
ResetColor：重設文字的屬性。
LightColor：將文字前景設為高亮度。
ReverseVideo：將目前的前景與背景色對調。
UnderScore：讓文字加上底線。

[游標位置]
GotoXY(int x,int y)：將游標移到(x,y)。
WhereX：傳回 x 座標。
WhereY：傳回 y 座標。
GoHome：回到原點。

[清除螢幕]
ClearEOL：自游標清除到行尾。
ClearBOL：自行首清除到游標。
ClearLine：清除游標所在整行。
ClearBOS：清除螢幕開頭到游標。
ClearEOS：清除游標到螢幕結尾。
ClearScreen：清除整個螢幕。

[其他]
Close：關閉 Console 視窗
SetBufferSize：設定 Console 的 Buffer Size
WrapAtEolOutput：設定是否自動換行
SetTitle：設定 Console 視窗標題列文字

5.5 鍵盤輸入處理

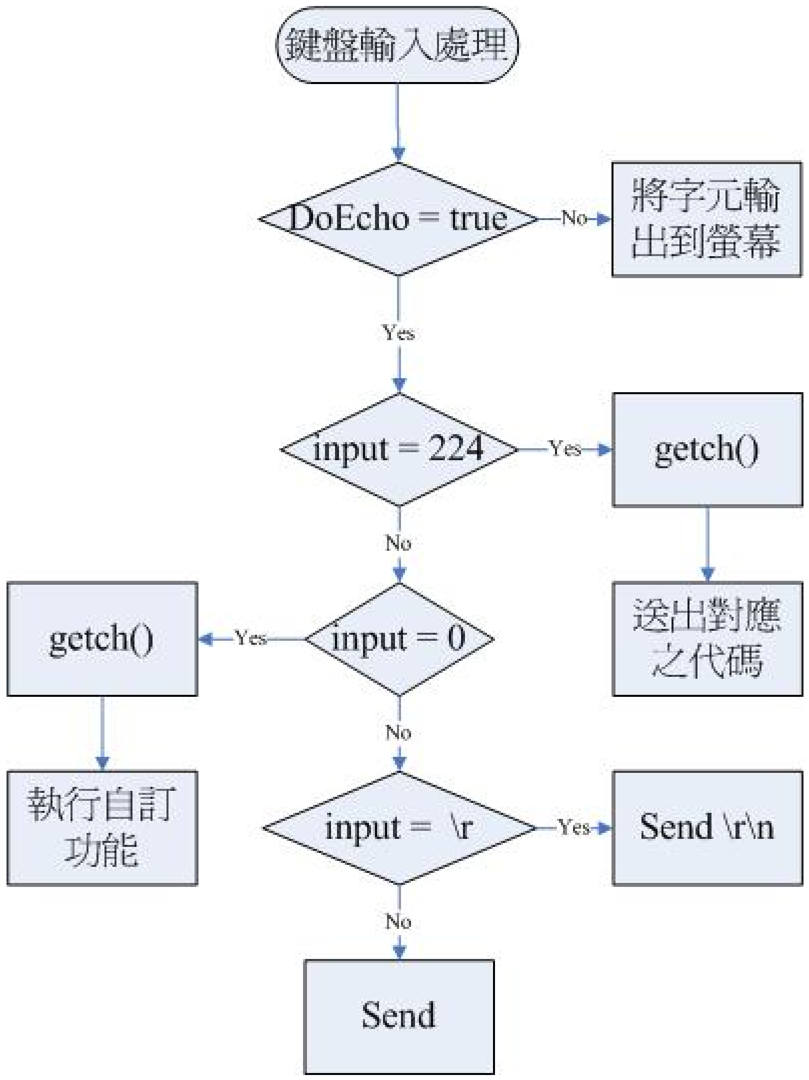


圖 5.8 鍵盤輸入的處理流程

圖 5.8 是使用者按下按鍵時的處理流程圖。

因為.net 中沒有類似 c 語言裡 getch()的函式可以讓按下的按鍵不顯示在螢幕上，所以我利用用 c 寫了一個小函式呼叫 getch()，然後編成 dll 檔供程式叫用。

這是在 c 裡面的寫法：

```
#include <conio.h>

int __declspec(dllexport) Cgetch()
{
    return getch();
}
```

把這個函式編輯成 Cgetch.dll，然後在 C#中就可以利用前面

引用 Windows API 的方法來引用這個 dll 檔。
這是在 C#裡面的寫法：

```
using System.Runtime.InteropServices;
…
[DllImport("Cgetch.dll")]
 int Cgetch();
```

以下是對於 getch()所傳回的 keycode 的處理流程說明：

(1) 方向鍵、Ins、Del、Home、End、PageUp、PageDown 這些鍵利用 getch()取得時，會取得兩個碼，第一個是 224，然後要用第二個碼來做判斷。

Code	KEY	Send
71	Home	ESC [1~
72	Up Arrow	ESC [A
73	Page Up	ESC [5~
75	Left Arrow	ESC [D
77	Right Arrow	ESC [C
79	End	ESC [4
80	Down Arrow	ESC [B
81	Page Down	ESC [6
82	Insert	ESC [2
83	Delete	127

表 5.1 Code 對照表

表 5.1 是 Code 為 224 X 的對映按鍵。

表 5.1 的第一個欄位是取得的第二個碼，第二個按鍵代表在鍵盤上對映的按鍵，第三個欄位代表應該送給 Server 的代碼。

(2) 按鍵 F1~F12 一樣會產生兩個碼，第一個碼是 0，以第二個碼可以判定是那一個按鍵被按下，這些按鍵並不需要傳回給 Server 做處理，可以留給程式自行運用。在這個程式中，使用了 F1~F4 四個按鍵。

59:F1，送出 IAC DO LOGOUT，告訴 Server 將使用者登出。至於是否會登出，則要看 Server 如何處理。

60:F2，自行將 socket 的連線結束掉。
61:F3，啟動/關閉 ANSI 的處理。
62:F4，啟動/關閉 Debug Mode。啟動時會將接收到的資料輸出到文字檔。

(3)　Enter 只會產生\r，但是 Telnet 協定中規定要送出\r\n。而如果是送出\r\0 則代表 Carriage Return。

(4)　其他的可視字元或是 Ctrl+c 等等的按鍵則直接送出。

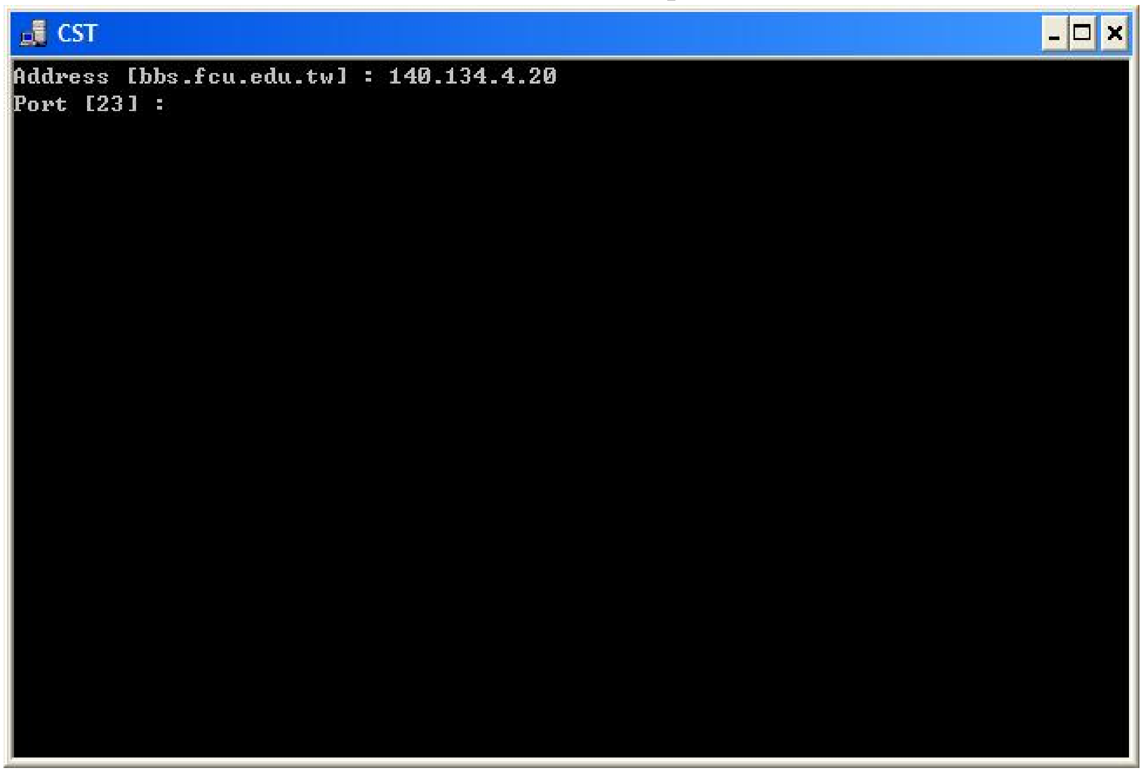
5.6 程式執行

這個程式是以 C#開發，所以必須要在安裝有.net framework 的電腦上才可執行。

.net framework 1.1 下載點：
http://download.microsoft.com/download/8/2/7/827bb1ef-f5e1-4464-9788-40ef682930fd/dotnetfx.exe

圖 5.9 為程式執行時之畫面：

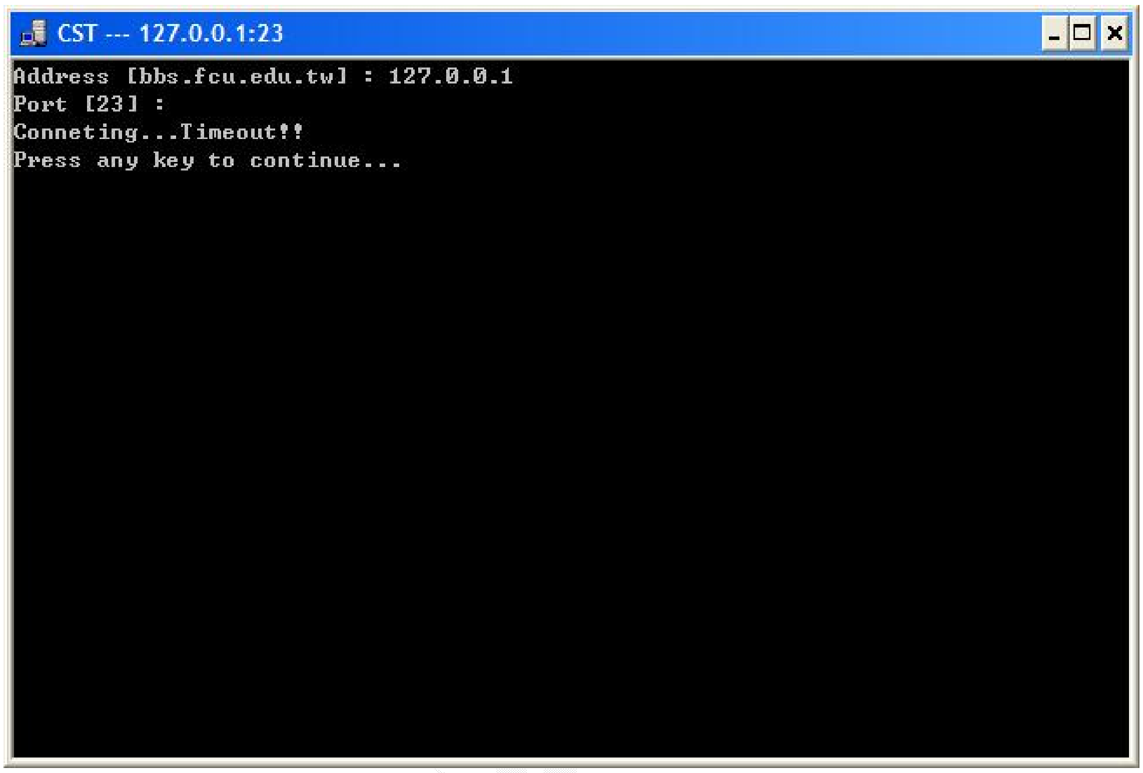
程式起始，要求使用者輸入 address 以及 port：



```
Address [bbs.fcu.edu.tw] : 140.134.4.20
Port [23] :
```

圖 5.9 輸入連線位址畫面

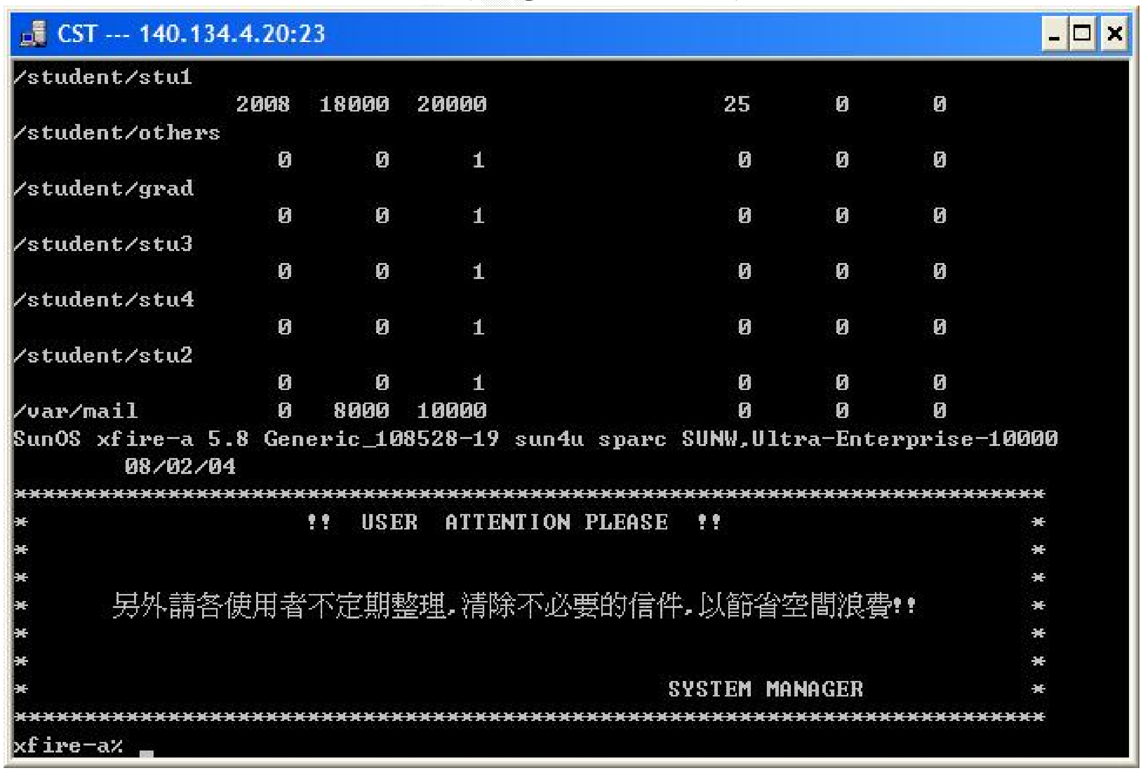
圖 5.10 是超過 10 秒，顯示 timeout 訊息的畫面：



```
Address [bbs.fcu.edu.tw] : 127.0.0.1
Port [23] :
Conneting...Timeout!!
Press any key to continue...
```

圖 5.10 Timeout 畫面

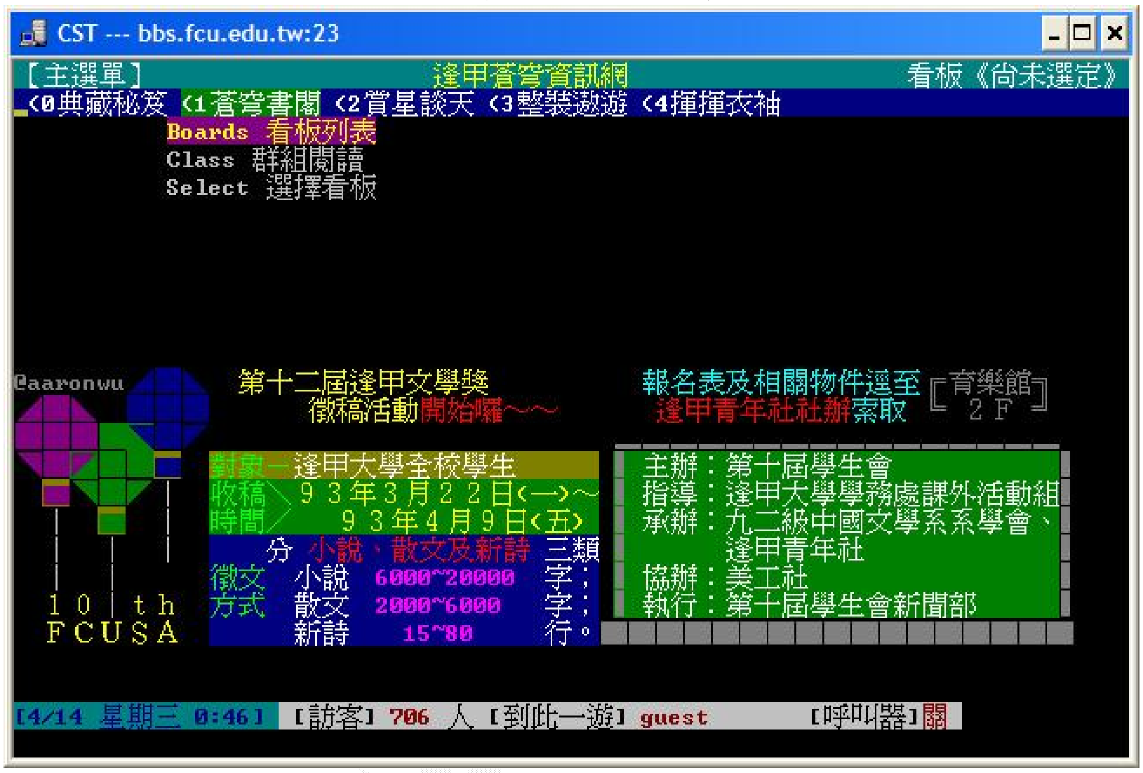
圖 5.11 為登入到 UNIX 主機(knight.fcu.edu.tw)的畫面：



```
/student/stu1
                2008  18000  20000                0      0
/student/others
                   0      0      1                  0      0      0
/student/grad
                   0      0      1                  0      0      0
/student/stu3
                   0      0      1                  0      0      0
/student/stu4
                   0      0      1                  0      0      0
/student/stu2
                   0      0      1                  0      0      0
/var/mail          0   8000  10000                  0      0      0
SunOS xfire-a 5.8 Generic_108528-19 sun4u sparc SUNW,Ultra-Enterprise-10000
        08/02/04
**************************************************************************
*                    !!  USER  ATTENTION PLEASE  !!                      *
*                                                                        *
*                                                                        *
*      另外請各使用者不定期整理,清除不必要的信件,以節省空間浪費!!              *
*                                                                        *
*                                                                        *
*                                              SYSTEM MANAGER            *
**************************************************************************
xfire-a% _
```

圖 5.11 登入到 Knight

圖 5.12 為登入到 BBS 的畫面(bbs.fcu.edu.tw)


CST --- bbs.fcu.edu.tw:23

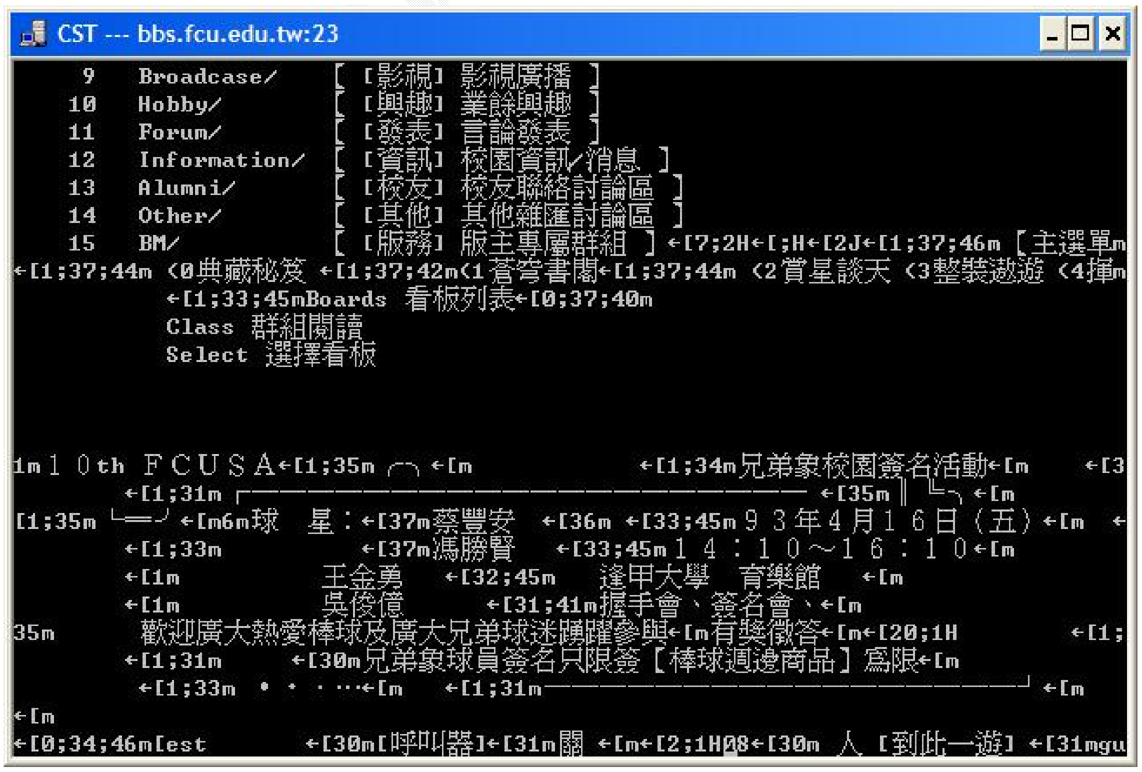
```
【主選單】                    逢甲蒼穹資訊網                    看板《尚未選定》
 <0典藏秘笈 <1蒼穹書閣 <2賞星談天 <3整裝遨遊 <4揮揮衣袖
           Boards  看板列表
           Class  群組閱讀
           Select  選擇看板

@aaronwu      第十二屆逢甲文學獎              報名表及相關物件遞至 [育樂館]
                  徵稿活動開始囉~~             逢甲青年社社辦索取  [ 2 F ]
             對象─逢甲大學全校學生           主辦:第十屆學生會
             收稿\ 9 3 年 3 月 2 2 日(一)~   指導:逢甲大學學務處課外活動組
             時間/    9 3 年 4 月 9 日(五)   承辦:九二級中國文學系系學會、
                分 小說、散文及新詩 三類          逢甲青年社
             徵文  小說   6000~20000   字;   協辦:美工社
   1 0  t h  方式  散文   2000~6000    字;   執行:第十屆學生會新聞部
  F C U S A        新詩     15~80      行。

[4/14 星期三 0:46] [訪客] 706 人 [到此一遊] guest        [呼叫器]關
```

圖 5.12 登入到逢甲蒼穹的畫面

5.13 將 ANSI 處理關閉時的畫面：


CST --- bbs.fcu.edu.tw:23

```
     9    Broadcase/       [ [影視] 影視廣播 ]
    10    Hobby/           [ [興趣] 業餘興趣 ]
    11    Forum/           [ [發表] 言論發表 ]
    12    Information/     [ [資訊] 校園資訊/消息 ]
    13    Alumni/          [ [校友] 校友聯絡討論區 ]
    14    Other/           [ [其他] 其他雜匯討論區 ]
    15    BM/              [ [版務] 版主專屬群組 ] ←[7;2H←[;H←[2J←[1;37;46m [主選單m
←[1;37;44m <0典藏秘笈 ←[1;37;42m<1蒼穹書閣←[1;37;44m <2賞星談天 <3整裝遨遊 <4揮m
           ←[1;33;45mBoards 看板列表←[0;37;40m
           Class 群組閱讀
           Select 選擇看板



1m1 0th F C U S A←[1;35m ╭╮ ←[m          ←[1;34m兄弟象校園簽名活動←[m    ←[3
      ←[1;31m ┌──────────────────────── ←[35m║ └╮ ←[m
[1;35m └═╯ ←[m6m球  星:←[37m蔡豐安  ←[36m ←[33;45m9 3 年4 月1 6 日 (五)←[m  ←
      ←[1;33m         ←[37m馮勝賢   ←[33;45m1 4 : 1 0 ~1 6 : 1 0←[m
      ←[1m           王金勇   ←[32;45m   逢甲大學  育樂館   ←[m
      ←[1m           吳俊億      ←[31;41m握手會、簽名會、←[m
35m       歡迎廣大熱愛棒球及廣大兄弟球迷踴躍參與←[m有獎徵答←[m←[20;1H       ←[1;
      ←[1;31m     ←[30m兄弟象球員簽名只限簽【棒球週邊商品】爲限←[m
       ←[1;33m • • · ···←[m   ←[1;31m──────────────────────────┘ ←[m
←[m
←[0;34;46m[est        ←[30m[呼叫器]←[31m關 ←[m←[2;1H08←[30m 人 [到此一遊] ←[31mgu
```

圖 5.13 關閉 ANSI 處理時的畫面

第六章 後記

6.1 程式尚未解決之問題

目前這個程式還有幾個問題沒有解決：

1. 文字閃爍屬性：

   遇到 Escape Sequence：ESC [5m 的時候，應該要啟動文字閃爍的屬性，但是我在 Windows API 中並沒有找到這功能，我所知道有提供這種功能的大概就只有 Turbo C 以及組合語言，所以遇到這個屬性時，我就把它忽略不處理。

2. 單一中文字雙重屬性：

   在很多的 BBS 都會出現一個中文字有兩種屬性的問題，例如：

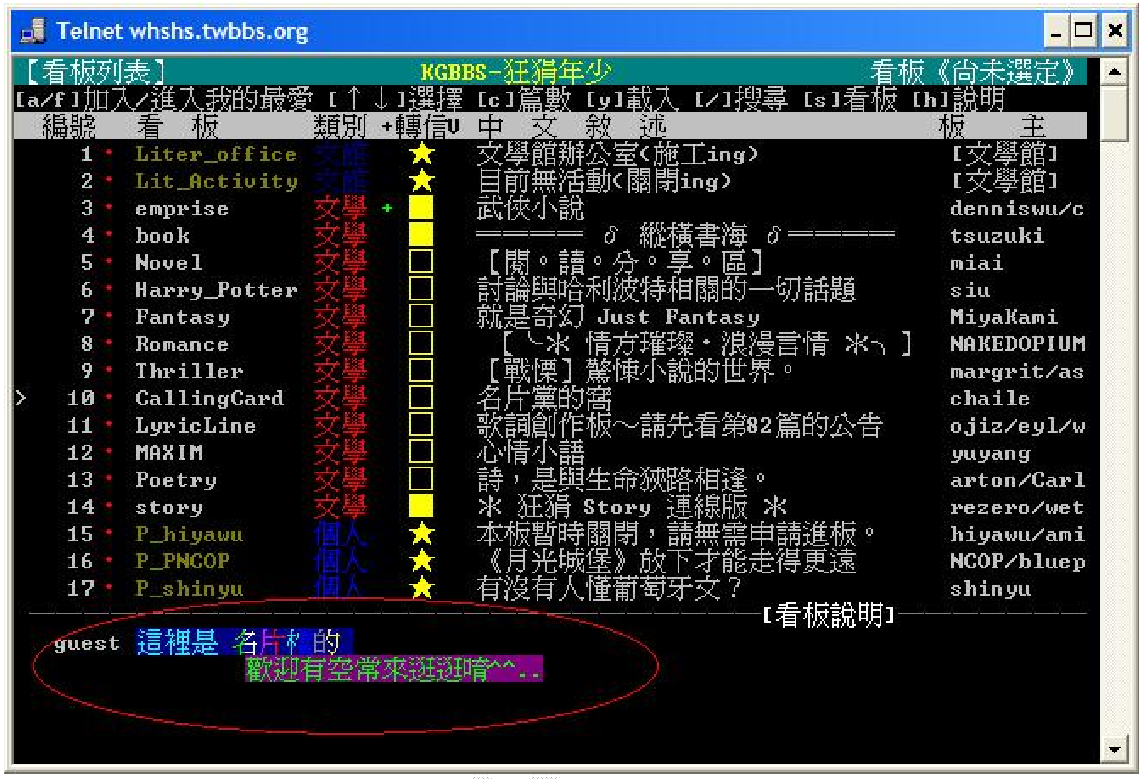
   ESC [31m 179_{10} ESC [32m 123_{10}

   上面的 Escape sequence 輸出的結果是一個前面一半是紅色而後面一半是綠色的”逢”。

   因為在.NET 中中文字也只占一個 char 的空間，沒有辦法分開處理，所以沒有辦法正常顯示，反而會因為.net 本身編碼函式的誤判，而把 179_{10}ESC 當成一個中文字，而造成後面的字元也都無法正常顯示。

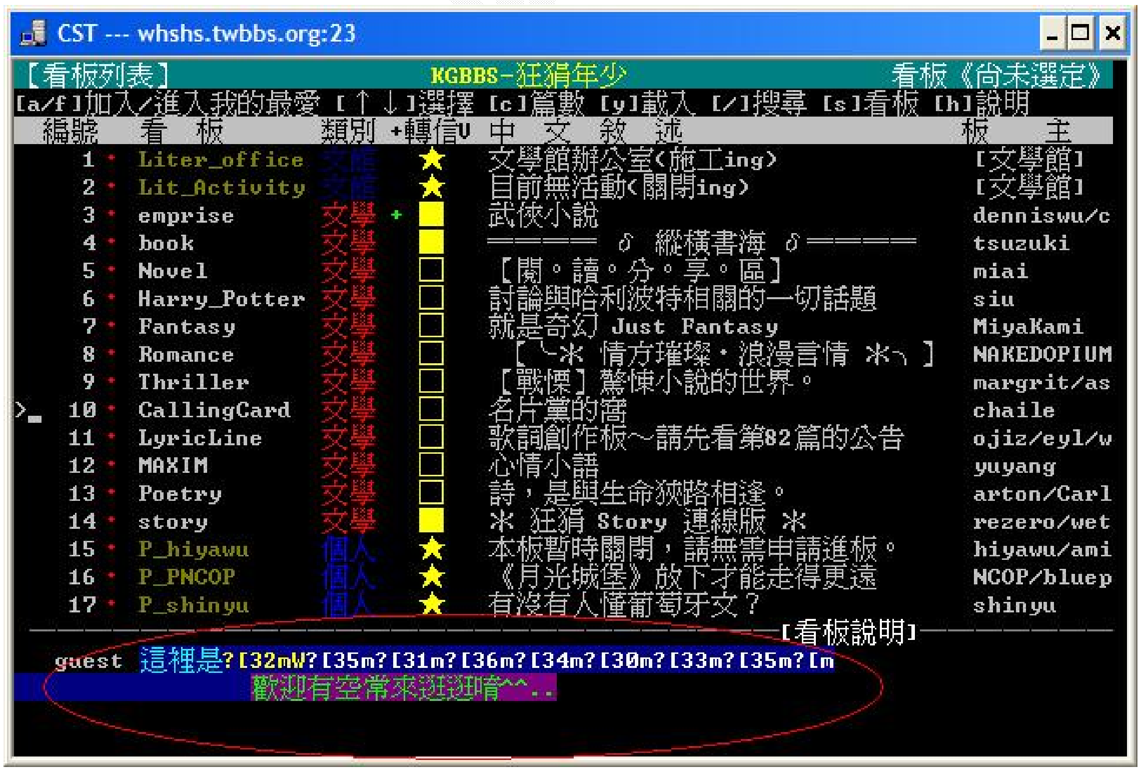
   註：依據 Big-5 的編碼規則，中文字的第一個位元組的範圍是 81-FE(129_{10}-254_{10})，第二個位元組的範圍則是 40-7E(64_{10}-126_{10})或 A1-FE(161_{10}-254_{10})。而 $179_{10}27_{10}$ 並不符合這規則，理論上來說應該會該編碼函式判定為兩個字，可是實際上卻被當成了一個字，實在是很奇怪。

   下面兩張圖是 Windows 本身附的 Telnet 和這個程式對於雙屬性中文字顯示的差異，圖 6.1 是 Windows 的 Telnet，圖 6.2 則是這個程式，在畫面下方有好幾個雙屬性的中文字：



```
Telnet whshs.twbbs.org
【看板列表】                      KGBBS-狂狷年少                    看板《尚未選定》
[a/f]加入/進入我的最愛 [↑↓]選擇 [c]篇數 [y]載入 [/]搜尋 [s]看板 [h]說明
 編號   看  板          類別 +轉信v 中  文  敘  述                        板   主
     1 ‧ Liter_office   文館   ★  文學館辦公室(施工ing)                  [文學館]
     2 ‧ Lit_Activity   文館   ★  目前無活動(關閉ing)                    [文學館]
     3 ‧ emprise        文學 + ■  武俠小說                               denniswu/c
     4 ‧ book           文學   ■  ═════ ♂ 縱橫書海 ♂═════               tsuzuki
     5 ‧ Novel          文學   □  [閱。讀。分。享。區]                   miai
     6 ‧ Harry_Potter   文學   □  討論與哈利波特相關的一切話題           siu
     7 ‧ Fantasy        文學   □  就是奇幻 Just Fantasy                  MiyaKami
     8 ‧ Romance        文學   □   [ ╰米 情方璀璨‧浪漫言情 米╮ ]       NAKEDOPIUM
     9 ‧ Thriller       文學   □  [戰慄] 驚悚小說的世界。                margrit/as
>   10 ‧ CallingCard    文學   □  名片黨的窩                             chaile
    11 ‧ LyricLine      文學   □  歌詞創作板~請先看第82篇的公告          ojiz/eyl/w
    12 ‧ MAXIM          文學   □  心情小語                               yuyang
    13 ‧ Poetry         文學   □  詩，是與生命狹路相逢。                 arton/Carl
    14 ‧ story          文學   ■  米 狂狷 Story 連線版 米                rezero/wet
    15 ‧ P_hiyawu       個人   ★  本板暫時關閉，請無需申請進板。         hiyawu/ami
    16 ‧ P_PNCOP        個人   ★  《月光城堡》放下才能走得更遠           NCOP/bluep
    17 ‧ P_shinyu       個人   ★  有沒有人懂葡萄牙文？                   shinyu
─────────────────────────────────────────[看板說明]──────────────
  guest 這裡是 名片ｷ 的
                   歡迎有空常來逛逛唷^^..
```

圖 6.1 Windows Telnet



```
CST --- whshs.twbbs.org:23
【看板列表】                      KGBBS-狂狷年少                    看板《尚未選定》
[a/f]加入/進入我的最愛 [↑↓]選擇 [c]篇數 [y]載入 [/]搜尋 [s]看板 [h]說明
 編號   看  板          類別 +轉信v 中  文  敘  述                        板   主
     1 ‧ Liter_office   文館   ★  文學館辦公室(施工ing)                  [文學館]
     2 ‧ Lit_Activity   文館   ★  目前無活動(關閉ing)                    [文學館]
     3 ‧ emprise        文學 + ■  武俠小說                               denniswu/c
     4 ‧ book           文學   ■  ═════ ♂ 縱橫書海 ♂═════               tsuzuki
     5 ‧ Novel          文學   □  [閱。讀。分。享。區]                   miai
     6 ‧ Harry_Potter   文學   □  討論與哈利波特相關的一切話題           siu
     7 ‧ Fantasy        文學   □  就是奇幻 Just Fantasy                  MiyaKami
     8 ‧ Romance        文學   □   [ ╰米 情方璀璨‧浪漫言情 米╮ ]       NAKEDOPIUM
     9 ‧ Thriller       文學   □  [戰慄] 驚悚小說的世界。                margrit/as
>_  10 ‧ CallingCard    文學   □  名片黨的窩                             chaile
    11 ‧ LyricLine      文學   □  歌詞創作板~請先看第82篇的公告          ojiz/eyl/w
    12 ‧ MAXIM          文學   □  心情小語                               yuyang
    13 ‧ Poetry         文學   □  詩，是與生命狹路相逢。                 arton/Carl
    14 ‧ story          文學   ■  米 狂狷 Story 連線版 米                rezero/wet
    15 ‧ P_hiyawu       個人   ★  本板暫時關閉，請無需申請進板。         hiyawu/ami
    16 ‧ P_PNCOP        個人   ★  《月光城堡》放下才能走得更遠           NCOP/bluep
    17 ‧ P_shinyu       個人   ★  有沒有人懂葡萄牙文？                   shinyu
─────────────────────────────────────────[看板說明]──────────────
  guest 這裡是?[32mW?[35m?[31m?[36m?[34m?[30m?[33m?[35m?[m
                   歡迎有空常來逛逛唷^^..
```

圖 6.2 這個程式對於雙屬性中文字無法正常顯示

3. 無法使用 vi：

不知道是什麼原因，在 unix 執行 vi 的時候，按鍵都沒辦法正常的運作，我找了很久，也沒找到關於 vi 的特殊規定，而且不只是這個程式，就連 Pcman、KKman、Netterm 也都沒辦法正常的使用 vi，我試過的程式，唯一能夠正常使用 vi 的只有 Windows 本身內建的 telnet。

4. Windows API 的使用：

就理論上而言，.net 的程式可以跨平台，只要在有安裝.net framework 的平台上，不管作業系統為何，.net 的程式都可以直接執行。不過.net framework 才剛起步，目前支援的功能還很少，所有我在程式中使用了許多的 Windows API，使用這些 API，造成了程式執行的不確定性，如果把這個程式拿到舊版的 Windows 上面，可能會有部分的功能不被支援。

這個問題必須要等到下一版的.net framework(1.2)推出才有辦法解決，從微軟釋出的文件可以發現到下一版的版本很大，功能也更加齊全，目前在這個程式中不得已使用的 API，到了下一版，就可以直接由.net framework 支援，自然就沒有呼叫 API 的必要性了。

6.2 感想

對於這次的專題，我覺得讓我最困擾的倒不是 coding 的部份，而是英文，我花了不少的時間去研讀規格書，但是始終無法捉住其重點，幸好還有其他的書籍可以參考，所以才能讓我稍稍有一些概念。

雖然原先的題目並沒有完成，但是我從中學到了很多以前不知道的用法與技巧，例如：.net 的自訂控制項，其功能就是讓程式設計師可以開發屬於自己的元件，加入自訂的屬性、事件以及方法，然後就可以像.net 本身附帶的元件一樣直接拖曳到視窗上就可以使用。我原本不知道這個用法的時候，WWW 和 BBS 都是混在一起寫的，程式真的亂的可以，寫到後來真的不知道怎麼繼續下去，但是後來在書上看到這種用法之後，我把 WWW 和 BBS 分開設計，然後再從主程式呼叫，整個程式看起來就簡潔明瞭許多了。

參考資料

1. Andrew Troelsen 原著，陳玄玲譯，用 C#談.NET 平台，旗標，April 2002
2. Anthony Jones and Jim ohlund 原著，黃嘉光譯，C#.NET 網際網路程式設計-TCP/IP 與 Internet Programming，文魁，May 2003
3. 超維度工作室，Visual Basic .NET Win32 API 大全集，博碩文化，2003
4. Comer,Douglas E. and David L.Stevens，Internetworking with TCP/IP Vol.Ⅲ:Client-Server Programming and Applications BSD Socket Version 2/E，Prentice Hall，1996
5. 陳弘馨，Microsoft Windows 網路程式設計第二版，Microsoft Press，October 2002
6. 陳永昱 and 黃禎福,Windows 2000 程式設計實務,旗標,September 2000
7. J. Postel and J. Reynolds，RFC 854:TELNET PROTOCOL SPECIFICATION，May 1983
8. Mark Crispin，RFC 727: Telnet Logout Option，April 1977
9. J. Postel and J. Reynolds，RFC 855: TELNET OPTION SPECIFICATIONS，May 1983
10. J. Postel and J. Reynolds，RFC 857: TELNET ECHO OPTION，May 1983
11. J. VanBokkelen，RFC 1091: Telnet Terminal-Type Option，February 1989
12. 微軟知識庫文件 KB231866，The TELNET Protocol
13. Paul Williams，VT100 User Guide

14. Paul Bourke，ANSI Standard (X3.64) Control Sequences for Video Terminals and Peripherals in alphabetic order by mnemonic，March 1989

15. http://www.ietf.org/rfc.html，The Internet Engineering Task Force － RFC pages

16. http://www.telnet.org/htm/dev.htm

17. http://vt100.net/

18. http://vt100.net/docs/vt100-ug/

19. http://www.fh-jena.de/~gmueller/Kurs_halle/esc_vt100.html

20. http://www.cmex.org.tw/，中推會

附錄 A Telnet 相關之 RFC

RFC 652	Telnet output carriage-return disposition option
RFC 653	Telnet output horizontal tabstops option
RFC 654	Telnet output horizontal tab disposition option
RFC 655	Telnet output formfeed disposition option
RFC 656	Telnet output vertical tabstops option
RFC 657	Telnet output vertical tab disposition option
RFC 658	Telnet output linefeed disposition
RFC 698	Telnet extended ASCII option
RFC 718	Comments on RCTE from the Tenex implementation experience
RFC 719	Discussion on RCTE
RFC 726	Remote Controlled Transmission and Echoing Telnet option
RFC 727	Telnet logout option
RFC 728	Minor pitfall in the Telnet Protocol
RFC 734	SUPDUP Protocol
RFC 735	Revised Telnet byte macro option
RFC 736	Telnet SUPDUP option
RFC 746	SUPDUP graphics extension
RFC 747	Recent extensions to the SUPDUP Protocol
RFC 748	Telnet randomly-lose option
RFC 749	Telnet SUPDUP-Output option
RFC 779	Telnet send-location option
RFC 782	Virtual Terminal management model
RFC 818	Remote User Telnet service
RFC 854	Telnet Protocol specification

RFC 855	Telnet option specifications
RFC 856	Telnet binary transmission
RFC 857	Telnet echo option
RFC 858	Telnet Suppress Go Ahead option
RFC 859	Telnet status option
RFC 860	Telnet timing mark option
RFC 861	Telnet extended options: List option
RFC 885	Telnet end of record option
RFC 927	TACACS user identification Telnet option
RFC 933	Output marking Telnet option
RFC 946	Telnet terminal location number option
RFC 1041	Telnet 3270 regime option
RFC 1043	Telnet Data Entry Terminal option: DODIIS implementation
RFC 1053	Telnet X.3 PAD option
RFC 1073	Telnet window size option
RFC 1079	Telnet terminal speed option
RFC 1091	Telnet terminal-type option
RFC 1096	Telnet X display location option
RFC 1097	Telnet subliminal-message option
RFC 1143	The Q Method of Implementing TELNET Option Negotiation
RFC 1184	Telnet Linemode option
RFC 1205	5250 Telnet Interface
RFC 1372	Telnet Remote Flow Control Option
RFC 1411	Telnet Authentication: Kerberos Version 4
RFC 1412	Telnet Authentication : SPX
RFC 1416	Telnet Authentication Option

RFC 1571	Telnet Environment Option Interoperability Issues
RFC 1572	Telnet Environment Option
RFC 1576	TN3270 Current Practices
RFC 1646	TN3270 Extensions for LUname and Printer Selection
RFC 1647	TN3270 Enhancements
RFC 1921	TNVIP protocol
RFC 2066	TELNET CHARSET Option
RFC 2217	Telnet Com Port Control Option
RFC 2840	TELNET KERMIT OPTION
RFC 2877	5250 Telnet Enhancements
RFC 2941	Telnet Authentication Option
RFC 2942	Telnet Authentication: Kerberos Version 5
RFC 2943	TELNET Authentication Using DSA
RFC 2944	Telnet Authentication: SRP
RFC 2945	The SRP Authentication and Key Exchange System
RFC 2946	Telnet Data Encryption Option
RFC 2947	Telnet Encryption: DES3 64 bit Cipher Feedback
RFC 2948	Telnet Encryption: DES3 64 bit Output Feedback
RFC 2949	Telnet Encryption: CAST-128 64 bit Output Feedback
RFC 2950	Telnet Encryption: CAST-128 64 bit Cipher Feedback
RFC 2951	TELNET Authentication Using KEA and SKIPJACK
RFC 2952	Telnet Encryption: DES 64 bit Cipher Feedback
RFC 2953	Telnet Encryption: DES 64 bit Output Feedback

附錄 B RFC 854

Network Working Group
Request for Comments: 854

Obsoletes: NIC 18639

May 1983

J. Postel
J. Reynolds
ISI

TELNET PROTOCOL SPECIFICATION

This RFC specifies a standard for the ARPA Internet community. Hosts on the ARPA Internet are expected to adopt and implement this standard.

INTRODUCTION

The purpose of the TELNET Protocol is to provide a fairly general, bi-directional, eight-bit byte oriented communications facility. Its primary goal is to allow a standard method of interfacing terminal devices and terminal-oriented processes to each other. It is envisioned that the protocol may also be used for terminal-terminal communication ("linking") and process-process communication (distributed computation).

GENERAL CONSIDERATIONS

A TELNET connection is a Transmission Control Protocol (TCP) connection used to transmit data with interspersed TELNET control information.

The TELNET Protocol is built upon three main ideas: first, the concept of a "Network Virtual Terminal"; second, the principle of negotiated options; and third, a symmetric view of terminals and processes.

1. When a TELNET connection is first established, each end is assumed to originate and terminate at a "Network Virtual Terminal", or NVT. An NVT is an imaginary device which provides a standard, network-wide, intermediate representation of a canonical terminal. This eliminates the need for "server" and "user" hosts to keep information about the characteristics of each other's terminals and terminal handling conventions. All hosts, both user and server, map their local device characteristics and conventions so as to appear to be dealing with an NVT over the network, and each can assume a similar mapping by the other party. The NVT is intended to strike a balance between being overly restricted (not providing hosts a rich enough vocabulary for mapping into their local character sets), and being overly inclusive (penalizing users with modest terminals).

NOTE: The "user" host is the host to which the physical terminal is normally attached, and the "server" host is the host which is normally providing some service. As an alternate point of view, applicable even in terminal-to-terminal or process-to-process communications, the "user" host is the host which initiated the communication.

2. The principle of negotiated options takes cognizance of the fact that many hosts will wish to provide additional services over and above those available within an NVT, and many users will have sophisticated terminals and would like to have elegant, rather than minimal, services. Independent of, but structured within the TELNET Protocol are various "options" that will be sanctioned and may be used with the "DO, DON'T, WILL, WON'T" structure (discussed below) to allow a user and server to agree to use a more elaborate (or perhaps just different) set of conventions for their TELNET connection. Such options could include changing the character set, the echo mode, etc.

The basic strategy for setting up the use of options is to have either party (or both) initiate a request that some option take effect. The other party may then either accept or reject the request. If the request is accepted the option immediately takes effect; if it is rejected the associated aspect of the connection remains as specified for an NVT. Clearly, a party may always refuse a request to enable, and must never refuse a request to disable some option since all parties must be prepared to support the NVT.

The syntax of option negotiation has been set up so that if both parties request an option simultaneously, each will see the other's request as the positive acknowledgment of its own.

3. The symmetry of the negotiation syntax can potentially lead to nonterminating acknowledgment loops -- each party seeing the incoming commands not as acknowledgments but as new requests which must be acknowledged. To prevent such loops, the following rules prevail:

a. Parties may only request a change in option status; i.e., a party may not send out a "request" merely to announce what mode it is in.

b. If a party receives what appears to be a request to enter some mode it is already in, the request should not be acknowledged. This non-response is essential to prevent endless loops in the negotiation. It is required that a response be sent to requests for a change of mode -- even if the mode is not changed.

c. Whenever one party sends an option command to a second party, whether as a request or an acknowledgment, and use of the option will have any effect on the processing of the data being sent from the first party to the second, then the command must be inserted in the data stream at the point where it is desired that it take effect. (It should be noted that some time will elapse between the transmission of a request and the receipt of an acknowledgment, which may be negative. Thus, a host may wish to buffer data, after requesting an option, until it learns whether the request is accepted or rejected, in order to hide the "uncertainty period" from the user.)

Option requests are likely to flurry back and forth when a TELNET connection is first established, as each party attempts to get the best possible service from the other party. Beyond that, however, options can be used to dynamically modify the characteristics of the connection to suit changing local conditions. For example, the NVT, as will be explained later, uses a transmission discipline well suited to the many "line at a time" applications such as BASIC, but poorly suited to the many "character at a time" applications such as NLS. A server might elect to devote the extra processor overhead required for a "character at a time" discipline when it was suitable for the local process and would negotiate an appropriate option. However, rather than then being permanently burdened with the extra processing overhead, it could switch (i.e., negotiate) back to NVT when the detailed control was no longer necessary.

It is possible for requests initiated by processes to stimulate a nonterminating request loop if the process responds to a rejection by merely re-requesting the option. To prevent such loops from occurring, rejected requests should not be repeated until something changes. Operationally, this can mean the process is running a different program, or the user has given another command, or whatever makes sense in the context of the given process and the given option. A good rule of thumb is that a re-request should only occur as a result of subsequent information from the other end of the connection or when demanded by local human intervention.

Option designers should not feel constrained by the somewhat limited syntax available for option negotiation. The intent of the simple syntax is to make it easy to have options -- since it is correspondingly easy to profess ignorance about them. If some particular option requires a richer negotiation structure than possible within "DO, DON'T, WILL, WON'T", the proper tack is to use "DO, DON'T, WILL, WON'T" to establish that both parties understand the option, and once this is accomplished a more exotic syntax can be used freely. For example, a party might send a request to alter (establish) line length. If it is accepted, then a different syntax can be used for actually negotiating the line length -- such a "sub-negotiation" might include fields for minimum allowable, maximum allowable and desired line lengths. The important concept is that such expanded negotiations should never begin until some prior (standard) negotiation has established that both parties are capable of parsing the expanded syntax.

In summary, WILL XXX is sent, by either party, to indicate that party's desire (offer) to begin performing option XXX, DO XXX and DON'T XXX being its positive and negative acknowledgments; similarly, DO XXX is sent to indicate a desire (request) that the other party (i.e., the recipient of the DO) begin performing option XXX, WILL XXX and WON'T XXX being the positive and negative acknowledgments. Since the NVT is what is left when no options are enabled, the DON'T and WON'T responses are guaranteed to leave the connection in a state which both ends can handle. Thus, all hosts may implement their TELNET processes to be totally unaware of options that are not supported, simply returning a rejection to (i.e., refusing) any option request that cannot be understood.

As much as possible, the TELNET protocol has been made server-user symmetrical so that it easily and naturally covers the user-user (linking) and server-server (cooperating processes) cases. It is hoped, but not absolutely required, that options will further this intent. In any case, it is explicitly acknowledged that symmetry is an operating principle rather than an ironclad rule.

A companion document, "TELNET Option Specifications," should be consulted for information about the procedure for establishing new options.

THE NETWORK VIRTUAL TERMINAL

The Network Virtual Terminal (NVT) is a bi-directional character device. The NVT has a printer and a keyboard. The printer responds to incoming data and the keyboard produces outgoing data which is sent over the TELNET connection and, if "echoes" are desired, to the NVT's printer as well. "Echoes" will not be expected to traverse the network (although options exist to enable a "remote" echoing mode of operation, no host is required to implement this option). The code set is seven-bit USASCII in an eight-bit field, except as modified herein. Any code conversion and timing considerations are local problems and do not affect the NVT.

TRANSMISSION OF DATA

Although a TELNET connection through the network is intrinsically full duplex, the NVT is to be viewed as a half-duplex device operating in a line-buffered mode. That is, unless and until options are negotiated to the contrary, the following default conditions pertain to the transmission of data over the TELNET connection:

1) Insofar as the availability of local buffer space permits, data should be accumulated in the host where it is generated until a complete line of data is ready for transmission, or until some locally-defined explicit signal to transmit occurs. This signal could be generated either by a process or by a human user.

The motivation for this rule is the high cost, to some hosts, of processing network input interrupts, coupled with the default NVT specification that "echoes" do not traverse the network. Thus, it is reasonable to buffer some amount of data at its source. Many systems take some processing action at the end of each input line (even line printers or card punches frequently tend to work this way), so the transmission should be triggered at the end of a line. On the other hand, a user or process may sometimes find it necessary or desirable to provide data which does not terminate at the end of a line; therefore implementers are cautioned to provide methods of locally signaling that all buffered data should be transmitted immediately.

2) When a process has completed sending data to an NVT printer and has no queued input from the NVT keyboard for further processing (i.e., when a process at one end of a TELNET connection cannot proceed without input from the other end), the process must transmit the TELNET Go Ahead (GA) command.

This rule is not intended to require that the TELNET GA command be sent from a terminal at the end of each line, since server hosts do not normally require a special signal (in addition to end-of-line or other locally-defined characters) in order to commence processing. Rather, the TELNET GA is designed to help a user's local host operate a physically half duplex terminal which has a "lockable" keyboard such as the IBM 2741. A description of this type of terminal may help to explain the proper use of the GA command.

The terminal-computer connection is always under control of either the user or the computer. Neither can unilaterally seize control from the other; rather the controlling end must relinguish its control explicitly. At the terminal end, the hardware is constructed so as to relinquish control each time that a "line" is terminated (i.e., when the "New Line" key is typed by the user). When this occurs, the attached (local) computer processes the input data, decides if output should be generated, and if not returns control to the terminal. If output should be generated, control is retained by the computer until all output has been transmitted.

The difficulties of using this type of terminal through the network should be obvious. The "local" computer is no longer able to decide whether to retain control after seeing an end-of-line signal or not; this decision can only be made by the "remote" computer which is processing the data. Therefore, the TELNET GA command provides a mechanism whereby the "remote" (server) computer can signal the "local" (user) computer that it is time to pass control to the user of the terminal. It should be transmitted at those times, and only at those times, when the user should be given control of the terminal. Note that premature transmission of the GA command may result in the blocking of output, since the user is likely to assume that the transmitting system has paused, and therefore he will fail to turn the line around manually.

The foregoing, of course, does not apply to the user-to-server direction of communication. In this direction, GAs may be sent at any time, but need not ever be sent. Also, if the TELNET connection is being used for process-to-process communication, GAs need not be sent in either direction. Finally, for terminal-to-terminal communication, GAs may be required in neither, one, or both directions. If a host plans to support terminal-to-terminal communication it is suggested that the host provide the user with a means of manually signaling that it is time for a GA to be sent over the TELNET connection; this, however, is not a requirement on the implementer of a TELNET process.

Note that the symmetry of the TELNET model requires that there is an NVT at each end of the TELNET connection, at least conceptually.

STANDARD REPRESENTATION OF CONTROL FUNCTIONS

As stated in the Introduction to this document, the primary goal of the TELNET protocol is the provision of a standard interfacing of terminal devices and terminal-oriented processes through the network. Early experiences with this type of interconnection have shown that certain functions are implemented by most servers, but that the methods of invoking these functions differ widely. For a human user who interacts with several server systems, these differences are highly frustrating. TELNET, therefore, defines a standard representation for five of these functions, as described below. These standard representations have standard, but not required, meanings (with the exception that the Interrupt Process (IP) function may be required by other protocols which use TELNET); that is, a system which does not provide the function to local users need not provide it to network users and may treat the standard representation for the function as a No-operation. On the other hand, a system which does provide the function to a local user is obliged to provide the same function to a network user who transmits the standard representation for the function.

Interrupt Process (IP)

Many systems provide a function which suspends, interrupts, aborts, or terminates the operation of a user process. This function is frequently used when a user believes his process is in an unending loop, or when an unwanted process has been inadvertently activated. IP is the standard representation for invoking this function. It should be noted by implementers that IP may be required by other protocols which use TELNET, and therefore should be implemented if these other protocols are to be supported.

Abort Output (AO)

Many systems provide a function which allows a process, which is generating output, to run to completion (or to reach the same stopping point it would reach if running to completion) but without sending the output to the user's terminal. Further, this function typically clears any output already produced but not yet actually printed (or displayed) on the user's terminal. AO is the standard representation for invoking this function. For example, some subsystem might normally accept a user's command, send a long text string to the user's terminal in response, and finally signal readiness to accept the next command by sending a "prompt" character (preceded by <CR><LF>) to the user's terminal. If the AO were received during the transmission of the text string, a reasonable implementation would be to suppress the remainder of the text string, but transmit the prompt character and the preceding <CR><LF>. (This is possibly in distinction to the action which might be taken if an IP were received; the IP might cause suppression of the text string and an exit from the subsystem.)

It should be noted, by server systems which provide this function, that there may be buffers external to the system (in the network and the user's local host) which should be cleared; the appropriate way to do this is to transmit the "Synch" signal (described below) to the user system.

Are You There (AYT)

Many systems provide a function which provides the user with some visible (e.g., printable) evidence that the system is still up and running. This function may be invoked by the user when the system is unexpectedly "silent" for a long time, because of the unanticipated (by the user) length of a computation, an unusually heavy system load, etc. AYT is the standard representation for invoking this function.

Erase Character (EC)

Many systems provide a function which deletes the last preceding undeleted character or "print position"* from the stream of data being supplied by the user. This function is typically used to edit keyboard input when typing mistakes are made. EC is the standard representation for invoking this function.

*NOTE: A "print position" may contain several characters which are the result of overstrikes, or of sequences such as <char1> BS <char2>...

Erase Line (EL)

Many systems provide a function which deletes all the data in the current "line" of input. This function is typically used to edit keyboard input. EL is the standard representation for invoking this function.

THE TELNET "SYNCH" SIGNAL

Most time-sharing systems provide mechanisms which allow a terminal user to regain control of a "runaway" process; the IP and AO functions described above are examples of these mechanisms. Such systems, when used locally, have access to all of the signals supplied by the user, whether these are normal characters or special "out of band" signals such as those supplied by the teletype "BREAK" key or the IBM 2741 "ATTN" key. This is not necessarily true when terminals are connected to the system through the network; the network's flow control mechanisms may cause such a signal to be buffered elsewhere, for example in the user's host.

To counter this problem, the TELNET "Synch" mechanism is introduced. A Synch signal consists of a TCP Urgent notification, coupled with the TELNET command DATA MARK. The Urgent notification, which is not subject to the flow control pertaining to the TELNET connection, is used to invoke special handling of the data stream by the process which receives it. In this mode, the data stream is immediately scanned for "interesting" signals as defined below, discarding intervening data. The TELNET command DATA MARK (DM) is the synchronizing mark in the data stream which indicates that any special signal has already occurred and the recipient can return to normal processing of the data stream.

The Synch is sent via the TCP send operation with the Urgent flag set and the DM as the last (or only) data octet.

When several Synchs are sent in rapid succession, the Urgent notifications may be merged. It is not possible to count Urgents since the number received will be less than or equal the number sent. When in normal mode, a DM is a no operation; when in urgent mode, it signals the end of the urgent processing.

If TCP indicates the end of Urgent data before the DM is found, TELNET should continue the special handling of the data stream until the DM is found.

If TCP indicates more Urgent data after the DM is found, it can only be because of a subsequent Synch. TELNET should continue the special handling of the data stream until another DM is found.

"Interesting" signals are defined to be: the TELNET standard representations of IP, AO, and AYT (but not EC or EL); the local analogs of these standard representations (if any); all other TELNET commands; other site-defined signals which can be acted on without delaying the scan of the data stream.

Since one effect of the SYNCH mechanism is the discarding of essentially all characters (except TELNET commands) between the sender of the Synch and its recipient, this mechanism is specified as the standard way to clear the data path when that is desired. For example, if a user at a terminal causes an AO to be transmitted, the server which receives the AO (if it provides that function at all) should return a Synch to the user.

Finally, just as the TCP Urgent notification is needed at the TELNET level as an out-of-band signal, so other protocols which make use of TELNET may require a TELNET command which can be viewed as an out-of-band signal at a different level.

By convention the sequence [IP, Synch] is to be used as such a signal. For example, suppose that some other protocol, which uses TELNET, defines the character string STOP analogously to the TELNET command AO. Imagine that a user of this protocol wishes a server to process the STOP string, but the connection is blocked because the server is processing other commands. The user should instruct his system to:

1. Send the TELNET IP character;

2. Send the TELNET SYNC sequence, that is:

   Send the Data Mark (DM) as the only character
   in a TCP urgent mode send operation.

3. Send the character string STOP; and

4. Send the other protocol's analog of the TELNET DM, if any.

The user (or process acting on his behalf) must transmit the TELNET SYNCH sequence of step 2 above to ensure that the TELNET IP gets through to the server's TELNET interpreter.

The Urgent should wake up the TELNET process; the IP should wake up the next higher level process.

THE NVT PRINTER AND KEYBOARD

The NVT printer has an unspecified carriage width and page length and can produce representations of all 95 USASCII graphics (codes 32 through 126). Of the 33 USASCII control codes (0 through 31 and 127), and the 128 uncovered codes (128 through 255), the following have specified meaning to the NVT printer:

NAME	CODE	MEANING
NULL (NUL)	0	No Operation
Line Feed (LF)	10	Moves the printer to the next print line, keeping the same horizontal position.
Carriage Return (CR)	13	Moves the printer to the left margin of the current line.

In addition, the following codes shall have defined, but not required, effects on the NVT printer. Neither end of a TELNET connection may assume that the other party will take, or will have taken, any particular action upon receipt or transmission of these:

BELL (BEL)	7	Produces an audible or visible signal (which does NOT move the print head).
Back Space (BS)	8	Moves the print head one character position towards the left margin.
Horizontal Tab (HT)	9	Moves the printer to the next horizontal tab stop. It remains unspecified how either party determines or establishes where such tab stops are located.
Vertical Tab (VT)	11	Moves the printer to the next vertical tab stop. It remains unspecified how either party determines or establishes where such tab stops are located.
Form Feed (FF)	12	Moves the printer to the top of the next page, keeping the same horizontal position.

All remaining codes do not cause the NVT printer to take any action.

The sequence "CR LF", as defined, will cause the NVT to be positioned at the left margin of the next print line (as would, for example, the sequence "LF CR"). However, many systems and terminals do not treat CR and LF independently, and will have to go to some effort to simulate their effect. (For example, some terminals do not have a CR independent of the LF, but on such terminals it may be possible to simulate a CR by backspacing.) Therefore, the sequence "CR LF" must be treated as a single "new line" character and used whenever their combined action is intended; the sequence "CR NUL" must be used where a carriage return alone is actually desired; and the CR character must be avoided in other contexts. This rule gives assurance to systems which must decide whether to perform a "new line" function or a multiple-backspace that the TELNET stream contains a character following a CR that will allow a rational decision.

Note that "CR LF" or "CR NUL" is required in both directions (in the default ASCII mode), to preserve the symmetry of the NVT model. Even though it may be known in some situations (e.g., with remote echo and suppress go ahead options in effect) that characters are not being sent to an actual printer, nonetheless, for the sake of consistency, the protocol requires that a NUL be inserted following a CR not followed by a LF in the data stream. The converse of this is that a NUL received in the data stream after a CR (in the absence of options negotiations which explicitly specify otherwise) should be stripped out prior to applying the NVT to local character set mapping.

The NVT keyboard has keys, or key combinations, or key sequences, for generating all 128 USASCII codes. Note that although many have no effect on the NVT printer, the NVT keyboard is capable of generating them.

In addition to these codes, the NVT keyboard shall be capable of generating the following additional codes which, except as noted, have defined, but not reguired, meanings. The actual code assignments for these "characters" are in the TELNET Command section, because they are viewed as being, in some sense, generic and should be available even when the data stream is interpreted as being some other character set.

Synch

This key allows the user to clear his data path to the other party. The activation of this key causes a DM (see command section) to be sent in the data stream and a TCP Urgent notification is associated with it. The pair DM-Urgent is to have required meaning as defined previously.

Break (BRK)

This code is provided because it is a signal outside the USASCII set which is currently given local meaning within many systems. It is intended to indicate that the Break Key or the Attention Key was hit. Note, however, that this is intended to provide a 129th code for systems which require it, not as a synonym for the IP standard representation.

Interrupt Process (IP)

Suspend, interrupt, abort or terminate the process to which the NVT is connected. Also, part of the out-of-band signal for other protocols which use TELNET.

Abort Output (AO)

Allow the current process to (appear to) run to completion, but do not send its output to the user. Also, send a Synch to the user.

Are You There (AYT)

Send back to the NVT some visible (i.e., printable) evidence that the AYT was received.

Erase Character (EC)

The recipient should delete the last preceding undeleted character or "print position" from the data stream.

Erase Line (EL)

The recipient should delete characters from the data stream back to, but not including, the last "CR LF" sequence sent over the TELNET connection.

The spirit of these "extra" keys, and also the printer format effectors, is that they should represent a natural extension of the mapping that already must be done from "NVT" into "local". Just as the NVT data byte 68 (104 octal) should be mapped into whatever the local code for "uppercase D" is, so the EC character should be mapped into whatever the local "Erase Character" function is. Further, just as the mapping for 124 (174 octal) is somewhat arbitrary in an environment that has no "vertical bar" character, the EL character may have a somewhat arbitrary mapping (or none at all) if there is no local "Erase Line" facility. Similarly for format effectors: if the terminal actually does have a "Vertical Tab", then the mapping for VT is obvious, and only when the terminal does not have a vertical tab should the effect of VT be unpredictable.

TELNET COMMAND STRUCTURE

All TELNET commands consist of at least a two byte sequence: the "Interpret as Command" (IAC) escape character followed by the code for the command. The commands dealing with option negotiation are three byte sequences, the third byte being the code for the option referenced. This format was chosen so that as more comprehensive use of the "data space" is made -- by negotiations from the basic NVT, of course -- collisions of data bytes with reserved command values will be minimized, all such collisions requiring the inconvenience, and inefficiency, of "escaping" the data bytes into the stream. With the current set-up, only the IAC need be doubled to be sent as data, and the other 255 codes may be passed transparently.

The following are the defined TELNET commands. Note that these codes and code sequences have the indicated meaning only when immediately preceded by an IAC.

NAME	CODE	MEANING
SE	240	End of subnegotiation parameters.
NOP	241	No operation.
Data Mark	242	The data stream portion of a Synch. This should always be accompanied by a TCP Urgent notification.
Break	243	NVT character BRK.
Interrupt Process	244	The function IP.
Abort output	245	The function AO.
Are You There	246	The function AYT.
Erase character	247	The function EC.
Erase Line	248	The function EL.
Go ahead	249	The GA signal.
SB	250	Indicates that what follows is subnegotiation of the indicated option.
WILL (option code)	251	Indicates the desire to begin performing, or confirmation that you are now performing, the indicated option.
WON'T (option code)	252	Indicates the refusal to perform, or continue performing, the indicated option.
DO (option code)	253	Indicates the request that the other party perform, or confirmation that you are expecting the other party to perform, the indicated option.
DON'T (option code)	254	Indicates the demand that the other party stop performing, or confirmation that you are no longer expecting the other party to perform, the indicated option.
IAC	255	Data Byte 255.

CONNECTION ESTABLISHMENT

The TELNET TCP connection is established between the user's port U and the server's port L. The server listens on its well known port L for such connections. Since a TCP connection is full duplex and identified by the pair of ports, the server can engage in many simultaneous connections involving its port L and different user ports U.

Port Assignment

When used for remote user access to service hosts (i.e., remote terminal access) this protocol is assigned server port 23 (27 octal). That is L=23.