電腦操作環境之改進

廖賀田 劉勉志
淡江大學資訊管理系 中央研究院資訊所

摘要

以 MsWindow 為代表之視窗介面大幅度地改進了傳統的命令列介面，但它們本身卻仍未支援命令列介面的許多重要功能，本論文分析這兩種介面的功能，並提出改進視窗介面的建議：(1)監控並掌握標準輸出資訊，(2)加速一般指令之取用，(3)補足指令參數及管線化功能，(4)改進線上批次處理的操作。(5)添加視窗化的遠端操控能力。我們依前述理念實作了一套使用介面，這個雛型系統以視窗方式操控遠端的 UNIX 平台，並整合檔案傳輸的功能。

關鍵詞：視窗桌面、圖形操作介面、人機介面、遠端操控

Refinement on Computer Operating Environment

Heh-Tyan Liaw* and Biam Chee Low**
*Department of Information Management,
Tamkang University, Tamsui, Taiwan, R.O.C.
htliaw@mail.im.tku.edu.tw
**Institute of Information Science, Academia Sinica, NanKang,
Taipei 115, Taiwan 115, Taiwan, R.O.C.
bclow@iis.sinica.edu.tw

Abstract

Window user interfaces such as MsWindow improve traditional command-line user interfaces. However, they do not directly support several important functions of command-line user interfaces. In this paper, further refinements for window user interfaces are proposed: (1) monitoring and ulitilizing standard output, (2) speeding up the invokation of general commands, (3) supporting both command arguments and piping capabilities, (4) facilitating online batch processing, and (5) supporting window-style remote control. According to above suggestions, a prototype for refined window user interface has been implemented. It operates in window-style to control remote UNIX platform and integrates FTP functions.

***Key Words*:** Window Desktop, GUI, Human Computer Interface, Remote Control

一、緒論

傳統上，電腦是透過鍵盤在命令列介面(command-line interface)上操作。在命令列介面下工作不能直接看到處理中的檔案，不能直接看到有哪些程式正在執行；在操作上不但須記憶指令、檔名，還容易發生打字錯誤。這對普通使用者(average users)來說是充滿困難，即使對專家使用者(expert users)來說也是不方便。

近年來，由於硬體的升級，視窗操作介面(GUI, graphic user interface)[16]已經流行起來。視窗介面的操作方式友善(user friendly)、直接、迅速並正確。但它本身仍未直接支援命令列介面的許多重要功能。簡單地說:(1)缺乏標準輸出管道(stdin)，使程式撰寫及維護變困難。(2)使用者不知道有哪些指令在預設目錄(以 PATH 指令設置的目錄)中；也不能迅速取用稍早執行過的指令。(3)必須回到命令列才能輸入指令參數(parameters)及選項(options)。(4)線上批次處理操作手續不便。(5)缺乏滑鼠型遠端操控(remote control)的功能。

本論文希望進一步地調和命令列介面與視窗介面的優點，使視窗介面能直接涵蓋命令列介面的所有功能。

以下我們先觀察幾個常見的操作介面(user interface)：

UNIX[1](或常見的 LINUX)是一個被廣泛使用的作業系統，而 BASH(GNU Bourne-Again Shell)[9]是 UNIX 上功能豐富的操控介面(Shell)。BASH 除了具備一般基本功能外，還提供豐富的批次處理能力。但 BASH(以及其它的 shell 如 SH, CSH)在使用上只能用鍵盤，遇到複雜的檔案目錄以及長檔名就不方便，還容易打錯。當指令產生大量輸出資料時，使用者就被迫要引用一些不自然的手段(如 more，output-redirection)。UNIX 是一個多工(multi-tasking)系統[11]，但使用者卻不能隨時看到有哪些程序在執行。此外，一般使用者在輸入提示(prompt)面前經常會不知道有哪些可用的指令。由以上的觀察，我們確認有需要讓諸如 BASH 的操控介面也能引用迅速便利的滑鼠(mouse)輸入，並能擁有顯現檔案圖示(icons)的視窗，及監控執行程序的工作列(task bar)。但在引入圖形介面的同時，我們也不希望操控介面原先的功能被打折扣。

Microsoft 公司的 MsDOS[15]曾經是個人電腦上最常用的作業系統。MsDOS 基本上只是單工(single-tasking)系統，它的功能比 UNIX 差很多，不值得討論。

Microsoft 公司近年來發展的 Windows 3.1[16], Windows 95[5], Windows NT[6]以及 Windows 98[7]都是目前世界上普及率很高的視窗作業系統，本文統稱以上的作業系統為 MsWindow。MsWindow 經過各個版本的改良後，已經廣泛的被使用。但是 MsWindow 本身(DOS 模式不算)仍然缺乏標準文字輸出管道，未支援參數列功能，更不具備批次處理功能。

早在 MsWindow 發行前，麻省理工學院(MIT)電腦科學研究室的 X Window[8]就已經是著名的視窗操作介面。X Window 的操控方式保持開放，因而也就五花八門，缺乏統一的操控習慣。以目前常見的各種 X Window 桌面來看，雖然操作方式各具巧思，但也含有前述 MsWindow 的弱點。

Symantec 公司的 pcANYWHERE[14]讓使用者以視窗化的方式操控遠端 MsWindow 平台，它在操控過程中傳送大量的圖形資料，這不但使網路塞車嚴重化，也降低了圖形介面的解析度。

本論文首先針對目前視窗操作介面與傳統命令列介面進行探討，根據它們的優缺點提出改進建議。此外，為檢驗所提出的改進，我們以 BASH 為例，實作了一套雛型系統，稱為 TkBASH。TkBASH 將 BASH 的操作視窗化，讓使用者能在 MsWindow 平台(platform)以視窗操作方式操控遠端的 UNIX 平台，並能以滑鼠拖移的方式進行檔案傳輸。TkBASH 在功能上可完全取代現有的終端機模擬程式(Telnet)及檔案傳輸程式(FTP)。

本論文分為四節。以下第二節探討傳統命令列介面與目前視窗桌面的弱點，並提出改善的方法。第三節介紹 TkBASH 的特色及操作方式。第四節為總結及後續工作。

二、視窗桌面與命令列介面的探討與改進

在視窗環境下，作業系統以視窗桌面(Window Desktop)取代傳統的命令列介面。傳統的操控方式是以文字串流(text stream)做輸入輸出；而視窗桌面引入圖形介面(graphic user interface)，使用者以滑鼠(mouse)配合按鈕(button)、圖示(icon)、選單(menu)等進行操作。這使得操作上變得友善、迅速、正確、直接。

命令列介面最大的弱點是強迫使用者記憶大量的指令，並需要鍵入複雜的檔名及路徑。在輸出方面，命令列介面的文字螢幕無法容納大量的輸出。此外，命令列介面在作批次處理時，必須另外再叫用編輯器，也不太方便。

視窗桌面應該可以改進並取代傳統的文字介面，可惜的是目前的視窗桌面仍未完全涵蓋傳統操控介面(Shell)的一切功能。目前世界上最流行的視窗桌面是 MsWindow，以下分項討論 MsWindow 及傳統命令列的弱點，並提出改進之道。

2.1 文字輸出

系統的輸出依性質可分為四類：在交談過程(interaction)中，使用者所下的指令及所產生的輸出資料依序呈現，本文稱它們為「正規輸出」。有時使用

者的操作含有錯誤或危險，系統就必須顯示警告訊息，這是「警告性輸出」。若使用者在線上查詢某個指令或某件事，系統的輸出是「臨時性輸出」。若使用者需持續使用某些資訊，這是「持久性輸出」。

傳統命令列介面將各種輸出都以文字印到螢幕上，使用者可藉著螢幕回憶最近的指令及結果。這種方式適合「正規輸出」，但對大量輸出資料的指令，使用者就還要特別處理以免遺失資料。另一方面，「警告性輸出」與「臨時性輸出」不必一直存在，它們若持續顯示在螢幕上，祇是對使用者造成干擾。「持久性輸出」必須切割輸出螢幕，在程式設計上比較麻煩，也與整個作業環境不協調。

視窗介面取消文字螢幕，改用彈出式的訊息窗(popup message box)、狀態列(status bar)、雲形提示(tools-tip)，協助窗(help window)等方式做輸出。彈出式的訊息窗會強迫使用者立刻注意，適用於「警告性輸出」。狀態列和雲形提示的資料很快就消失，適用於「臨時性輸出」。協助窗可用於「臨時性輸出」和「持久性輸出」。但是，它們全都不適合做最常用的「正規輸出」。彈出式訊息窗、狀態列、雲形提示使用後就消失得無影無蹤，使用者必須努力記憶剛才的訊息，甚至還必須用筆在紙上做記錄，這在使用介面上變成是在開倒車。以下我們以一個實例進行說明：

實例 1 在輸出方面的探討

sort 指令會將輸入檔按某欄做排序，再把結果顯示出來。有許多 UNIX 指令和 sort 一樣，會有大量的正規輸出，例如：du(disk usage, 顯示磁碟使用狀況)、grep(檔案內容之字串搜尋)等等，這些輸出資訊都必須保留供使用者參考。MsWindow 本身缺乏作正規輸出的標準途徑(stdout)，無法將這些訊息保留起來。於是使用者祇好回到 MsDOS 的命令列介面來操作。

在命令列下操作時，由於輸出空間有限，還必須額外引用輸出暫停(經 pipe 串接 more 指令)。或是經輸出轉向(output redirection)將輸出導入檔案，之後再以編輯器閱讀資料。這種手續既不自然，也不方便。

在傳統命令列介面下，程式師用簡單的指令就可進行輸出動作，但現行的視窗環境未提供標準文字輸出管道。於是要做輸出就必須引用複雜，不統一，又難維護的程式庫(如 Microsoft Win32 SDK)。按理說使用這種程式庫應該只是改變輸出管道，程式邏輯不必受影響。但事實上這些程式庫不但影響程式邏輯，還更改程式架構。程式的入口點(如 C 語言的 main)不見了，視窗 I/O 的事件驅動(event driven)方式甚至讓傳統的 Turing machine 程式模型[2][10]不再適用。傳統的程式師在視窗時代竟忽然就變成"不會寫程式"了。此外，程式模型的改變讓本就很困難的軟體維護(software re-engeneering)變得更加困難。

本論文主張視窗介面應配置持續存在的「文字輸出窗」(text output window)當做標準輸出管道，用來滿足正規輸出的需求。「文字輸出窗」的功能包括：(1)顯示使用者指令的回應(echo)和程式的輸出。 (2)可保留大量的訊息，並提供捲軸讓使用者能自由閱讀。(3)輸出窗內的資料能被[選取][複製]，以便再利用。有了文字輸出窗當標準輸出管道，絕大多數的程式都可以繼續用傳統的方式寫作，已寫好的工具(utilities)也不須改寫。所有命令列介面的程式都自動變成可以在視窗環境使用，而且比純粹的命令列介面還好用(大量資料可回捲)。目前有些終端機模擬程式(Telnet)已提供可回捲的捲軸，但 MsWindow 內的「DOS 模式」仍未提供可閱讀大量輸出資料的捲軸。

2.2 指令位置

所謂指令位置指的是指令在檔案系統內的擺放位置，也涉及指令如何被取用。在傳統命令列介面下，使用者必須記憶指令名稱及存放的路徑。視窗環境主要是以滑鼠點按(click 或 double click)的方式輸入指令，這種方式不必記憶指令的拼法，但是使用者必須先能看到所要的指令圖示才能執行。

在傳統文字介面上，指令依位置可分為以下幾種：

(1) 內建指令(shell built-in)或別名(alias)。例如：cd(換目錄)，mv(更名)等。
(2) 在「目前工作目錄」 (current working directory)的指令。
(3) 在任意目錄的指令。執行此種指令須鍵入完整的路徑。
(4) 在預設路徑下的指令。預設路徑可經由指令 PATH 設定並更改。
(5) 稍早以前下過的指令。例如：UNIX 的!-1 代表上個指令。MsDOS 執行常駐程式 doskey 後也可重複執行之前的指令。

目前 MsWindow 的指令位置僅支援以上述第(1)、(2)、(3)項：內建指令已包含在視窗的一般操作中，視窗「捷徑」 (short-cut)的功能與別名相似。檔案總管(file manager 或 explorer)隨時呈現目前工作目錄供點選。對於在任意目錄的指令，則採用彈出式的瀏覽視窗，讓使用者選取。

而第(4)項，MsWindow 的「捷徑」並不算是路徑功能：路徑功能是以目錄為單位，而捷徑功能只是以個別指令為單位。

至於第(5)項，目前 MsWindow 的操作介面本身完全不幫助使用者回顧操作過程。歷程記錄其實十分重要，一方面可以幫助使用者回憶長時間的操作過程，另一方面還能夠讓使用者快速執行稍早前執行過的指令。

本論文主張視窗桌面上應有「常用窗」，用來顯示第(1)種指令。有「工作窗」，用來顯示第(2)種指令。有「瀏覽窗」，用來顯示第(3)種指令。 使用者通常是在「工作窗」內瀏覽檔案、下達指令或切換工作目錄。一些常用的指令收集在「常用窗」內，使者用祇

要按一下"子視窗標籤(tab)"切到「常用窗」，就能直接執行常用的指令。有時候使用者想要執行其它目錄的執行檔而不想改變工作目錄，則可以透過「瀏覽窗」來點選(或查看)檔案。

本文建議添加「索引功能表」滿足第(4)種指令，讓使用者可依字母順序經少數次按鍵就可取得預設路徑中的指令。此種指令數量通常很多(約有四、五百個)，所以「索引功能表」必須引用階層式的彈出選單。為便於選取，每層的選項應在 20 項之內。在一個典型的系統裡，假設預設路徑下擁有 500 個指令，祇要 2-3 層的選單就能夠支援所有的指令。對一般含有 n 道指令的系統，使用者祇需經少數次按鍵($\log_{20}n$)就可以找到所要的指令。

本論文主張桌面上應配置下拉式指令歷程選單以處理第(5)種指令，讓使用者能夠輕易地回顧或執行先前下過的指令。

2.3 命令列的參數與選項

作業系統的指令常帶著參數(arguments)及選項(options)，此外通常還提供管線(piping)及輸出入轉接(IO-redirection)的功能。在命令列介面上有輸入緩衝區(input buffer)，使用者將指令完整輸入完畢，按下送入鍵(enter)後才執行這整個指令。命令列的缺點是：使用者必須完全以鍵盤輸入長串的指令及參數。

MsWindow 雖然改用以滑鼠點選取代鍵盤輸入的操作方式，但是其操作是屬於立即式的操作方式：一但指令被要求執行，就沒有機會在執行前提供完整的參數及選項。MsWindow 對這個問題的處理分成兩種方式：

(1)特殊指令專案處理

特殊指令是一些最常用的指令，MsWindow 採用專案處理的方式事先為這些指令規劃出特別的操作方法，包括下列各種形式：

(A)最常用的指令被視窗操作取代，如 ls(查閱目錄)，cd(換目錄)。

(B)某些檔案處理的指令由點選功能，配合剪貼簿(clipboard)完成。例如：複製檔案 COPY file1 path2\file1 的操作如下：點選[file1]圖示，點選[複製]，切換到[path1]，點選[貼上]。前述操作也可以用滑鼠拖放直接操作。像這種方式雖然方便，但並未周延。某些常見的操作像 COPY file1 file2 還必須外加其它動作才能完成。此外，有些稍複雜的用法在視窗介面上完全不能用，例如：COPY file1+file2 file3。又如 DIR > filelisting。

(C)某些特殊指令配加對話盒(dialog box)。例如：在 MsWindow 下[尋找]，以彈出式的「尋找對話窗」取代原本的 FIND 指令。

(2)一般指令未處理

一般指令則是其他 MsWindow 未事先安排的指令或使用者自行安裝的其他程式。作業系統無法預知使用者將會引入什麼新指令，無法為一般指令安排對話盒或特殊的操作方式。因此 MsWindow 就不處理一般指令的參數問題，只是讓使用者回去用命令列操作。

當然一般指令也可能在執行時，自行彈出對話盒(Dialog Box)讓使用者輸入或選取所需的資料。但這種輸入方式是執行期輸入(run-time input)，而不是執行前的命令參數。執行期輸入是使用者對程式以互動的方式提供資料，程式等待使用者的輸入才繼續執行。程式若使用命令列參數(例如：C 語言的 argc, argv)[3]，MsWindow 的使用者就只能用命令列操作。另外，祇有支援命令參數的指令才能做不進行交談(interaction)的批次處理。

我們觀察一些常見的實例:

實例 2 指令參數的探討

許多指令或執行檔都以檔名當作參數，例如：使用者想不經過編輯器(editor)把文字檔 file1 及 file2 合併起來儲到 file3。在 UNIX 下可鍵入指令：cat file1 file2 > file3。在 MsDOS 下可鍵入指令：copy file+file2 file3。MsWindow 並未支援這樣的指令。當有這種需求時，使用者只好回到命令列介面操作。

目前使用者都將檔案以多階層目錄及長檔名的方式儲存以管理大量的檔案。於是在命令列操作時就變成經常要輸入一長串文字。例如以下是一個典型的例子：

FC F:\user\example\C\Chap3\exe1.c C:\家庭作業\c 語言\Chap3\exe1.c

由於 MsWindow 本身未支援參數列操作，使用者祇好進入 DOS 模式用命令列操作。在命令列下，使用者常須以鍵盤輸入長檔名，這不但不方便，也容易打錯。這種現象在使用中文時就更加惡化。

本論文主張視窗介面應保留傳統命令列介面上「輸入緩衝區」(input buffer)的功能，以滿足使用者為指令輸入參數的需求。我們主張將桌面的工作模式區分為「立即模式」(immediate mode)及「緩衝模式」(buffer mode)。所謂「立即模式」，就是如目前 MsWindow 的操作模式，使用者點選(click 或 double click)某個圖示、按鈕或選單，代表立即執行。「緩衝模式」分為「單行緩衝模式」和 2.4 節將討論的「多行緩衝模式」。在「單行緩衝模式」下，點選按鈕、圖示或選單只是將其名稱抄入輸入緩衝區內，並不立即執行。使用者可以更正或繼續輸入其他參數，直到按下[送入](Enter)鈕後才開始執行。利用「緩衝模式」，使用者可藉滑鼠點選輸入檔名，這使輸入變得迅速，正確，直接，而且免記檔名及路徑。緩衝模式下若對某個圖示作雙敲(double click)，系統將名稱抄入輸入緩衝區後就立

即執行。所以緩衝模式也兼顧立即模式的方便性。

緩衝模式下的「輸入緩衝區」不只是適用於輸入作業系統級的指令。它還能自動支援任何由標準輸入(stdin)讀取資料的程式。當某程式所需的輸入資料是檔名時，使用者可以經滑鼠點按，迅速輸入正確無誤的檔名及路徑。對一般性的輸入資料(像公司名，地址等)，使用者也可在「常用窗」建立片語圖示，之後就可反覆利用滑鼠點按的方式做輸入。

2.4 流程控制

作業系統的操作除了交談式操作外，對例行或冗長的工作還常需要用到批次處理(batch processing)。當傳統命令介面的使用者想進行批次處理時，由於命令列下祇有一行的輸入緩衝區，就不容易直接進行多行的指令編寫。於是，使用者必須利用編輯器進行寫作，完工後再儲存為批次檔，然後再執行這個批次檔，執行完畢後可能還要刪除批次檔。而 MsWindow 的視窗桌面本身則完全未支援流程控制及批次處理。

以下是一個運用批次處理的實例:

實例 3 批次處理的探討

在作資料備份或是軟體系統開發的版本維護時，常常需要比對兩個版本間增加或減少了哪些檔案。這類工作若以人工方式檢查，不但緩慢，而且容易發生錯誤，應該改用批次作業。由於 MsWindow 不支援批次能力，使用者祇能回到命令列下作處理。在 BASH 系統下，使用者可以編寫[1]如下的函數 DirComp 來比較兩個目錄下的檔案是否有增減：

```
DirComp()
{
    SaveDir=`pwd`
    cd $1
    for FILE in *
    do
        if [ ! -e $2/$FILE ]
        then
            echo $FILE ": only in $1 "
        fi
    done
    cd $SaveDir
}
```

定義好所需的函數後，使用者就可執行下列指令來完成工作:

```
DirComp/Backup/19980812/TkBASH
/User/Working/TkBash/Current
```

在傳統命令列下，由於祇有一行的輸入緩衝區，在使用流程控制時總是涉及多道指令，要直接編寫像實例 3 這樣的函數相當困難。本論文主張進一步添加「多行緩衝模式」。「多行緩衝模式」與「單行緩衝模式」的運作方式相同，但使用多行的資料輸入區。使用者可以從容地寫好一整段批次指令後，才按下[送入](Enter)鈕交付執行。

這種方式比另外再開啟編輯器來得直接且方便，尤其是在「緩衝模式」下除了可用鍵盤輸入外，還可經滑鼠點按的方式輸入環境變數(environment variable)、指令、檔名、或自訂的片語。因此桌面上還要有「環境窗」、「片語窗」(Phrase Window)及「關鍵字工具列」。

2.5 遠端操控

在目前方便與發達的網路環境下，作業系統應提供遠端操控其他電腦的能力。目前 MsWindow 提供「檔案及列印分享」讓使用者使用其它電腦的檔案或印表機，但卻不能使用遠端電腦的 CPU。

使用者能夠以終端機模擬(Telnet)程式，將本身的電腦模擬為遠端主機的終端機而進行操作。但 Telnet 程式祇提供命令列介面，於是使用者在操控其他電腦時就喪失原先便利的視窗介面。pcANYWHERE[14] 雖然能以視窗介面操控遠端 MsWindow 主機，但是在操作過程中傳送大量圖形資料，而造成執行速度緩慢。

本論文主張在視窗環境下應提供視窗介面的遠端操控能力，讓使用者也同樣能夠以視窗化的方式操控遠端主機。在操控的過程中，應傳送簡潔的內部控制訊息，而不是傳送佔用大量頻寬的影像資料。本地端接收到控制訊息後，再以圖形介面顯示出來，這樣不但可以增加執行效率，而且不致於要降低影像的解析度。

三、TkBASH

本論文依據上節所提出的各項理念，實作完成一套新的使用介面，稱作 TkBASH。TkBASH 為 UNIX 下的 BASH 提供視窗操控介面，讓使用者以視窗操作的方式操控遠端的 UNIX 主機，並且完全支援原先 BASH 的各種功能。此外還整合了檔案傳輸服務，功能上改進並涵蓋終端機模擬程式(telnet)及檔案傳輸程式(FTP)。我們選用 BASH 只是拿它當做操控介面(shells)的代表，其它操控介面像 SH, CSH 等在本質上並無不同。

3.1 TkBASH 操作簡介

TkBASH 執行主畫面如圖 1 所示，以下分項做簡單的介紹。

(1) 「文字輸出窗」

圖 1 最上方為「文字輸出窗」，負責顯示輸入的回應(echo)及程式的輸出資料。

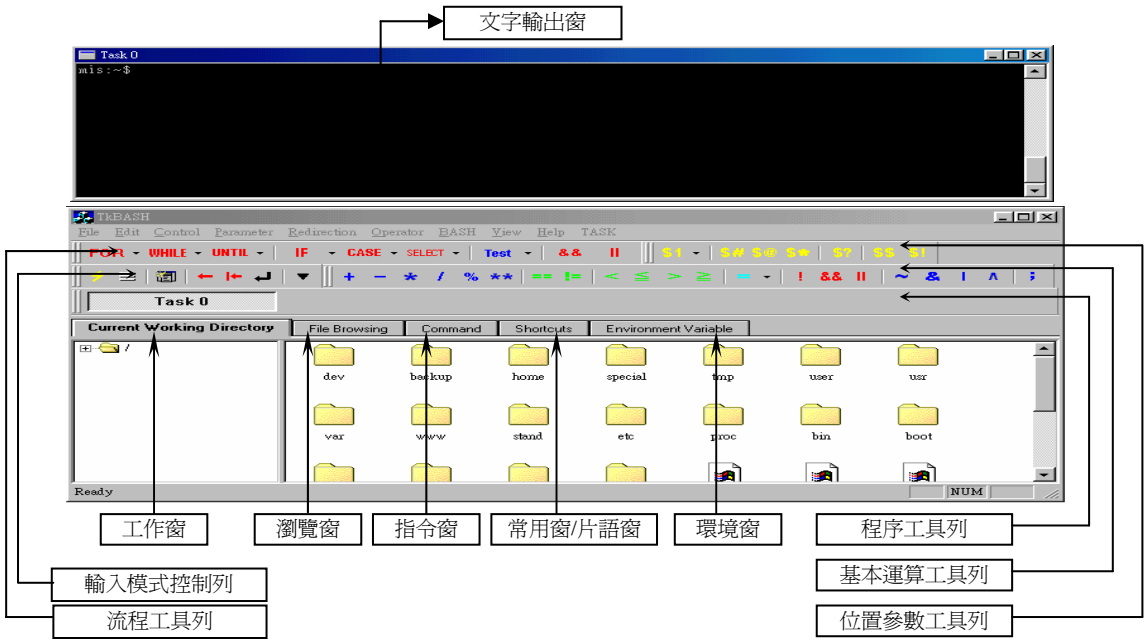

圖 1 TkBASH 程式主畫面

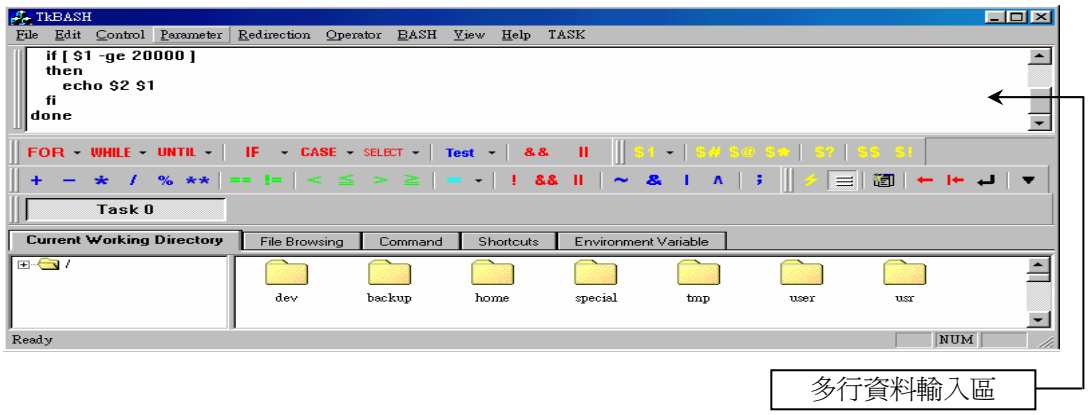

圖 2 「多行緩衝模式」下的資料輸入區

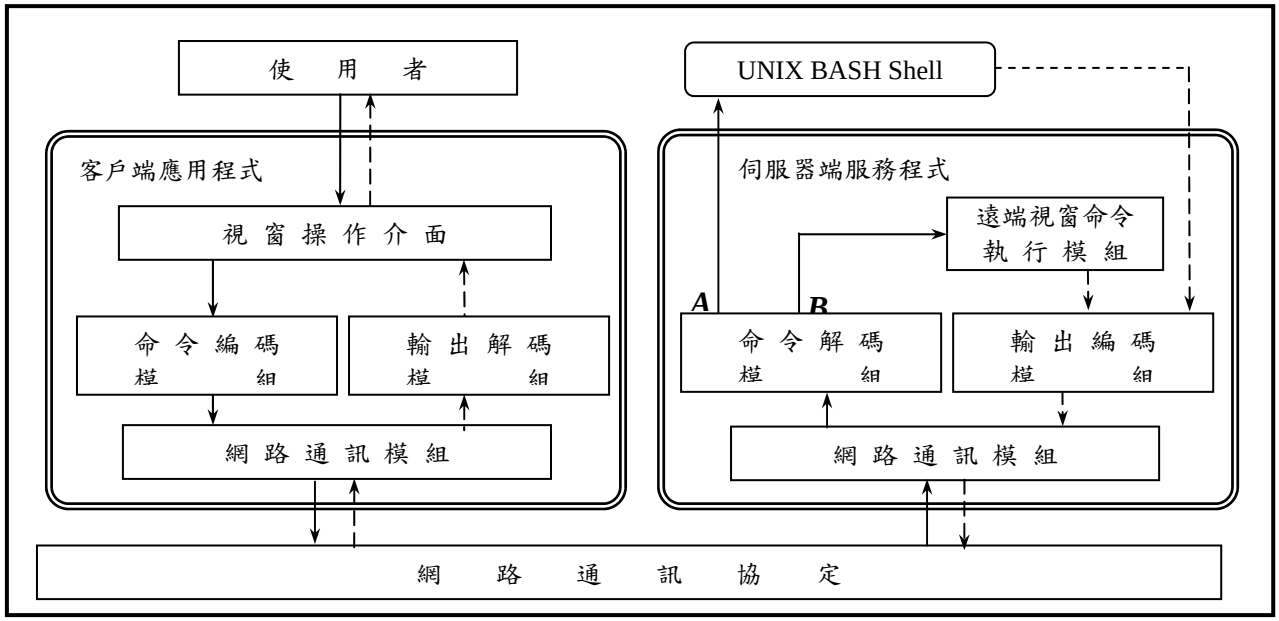

圖 3 TkBASH 系統架構圖

(2) 「程序工具列」

在「程序工具列」上會以按鈕的方式列出執行中的程序，使用者可以方便地利用按鈕切換程序到前景或背景執行。TkBASH 在啟動時產生的第一個程序稱作「Shell 程序」，使用者想執行某個可很快結束的程式時，就直接在 Shell 程序內執行該程式。若程式需要執行比較長的時間，可利用[獨立執行]鈕(位於輸入模式控制列)啟動這類程式，TkBASH 會為它多開啟一個文字輸出窗，並將它設為前景執行。這時「程序工具列」會多一個按鈕。

(3)「輸入模式控制列」

「輸入模式控制列」包括：1. 「立即模式切換鈕」、 2.「行緩衝模式切換鈕」、 3.「送入鍵」、 4.「獨立執行鈕」、 5.「空白鍵」、 6. 「倒退鍵」、 7.「歷程選單鈕」。

使用者可以利用[立即模式切換鈕]切換輸入模式為「立即模式」或「緩衝模式」。[行緩衝模式切換鈕]則切換緩衝模式為[單行]或[多行]。「歷程選單鈕」可幫助使用者回憶或重複執行稍早前的指令。

(4)「多行資料緩衝區」

在「多行緩衝模式」下，TkBASH 提供一個多行的資料緩衝區(如圖 2)，讓使用者可以同時輸入多行指令，直接在線上進行批次處理。

(5) 「流程工具列」與「位置參數工具列」

BASH 的流程控制指令非常多，我們將同性質的指令組織在一起，放入「流程工具列」內，並以動態顯現的雲形提示(tooltips)提供簡單的說明。「位置參數工具列」支援 BASH 的參數語法，包括：[$#]代表參數個數、[$?]代表函數回傳值、[$$]代表目前執行程序的編號等等。

(6)各類標籤視窗(Tab Windows)

在圖 1 主視窗下方的工作區劃分為五個不同功能的標籤視窗，有：1. 「工作窗」、2.「瀏覽窗」、 3.「指令窗」、 4.「常用窗/片語窗」、 5.「環境窗」。這些標籤視窗讓使用者能以滑鼠迅速輸入指令、檔名、及常用的片語。

一般情況，使用者是在「工作窗」運用目前工作目錄下的檔案及指令(輸入時不含路徑)，必要時也可切換工作目錄。若不想改動工作目錄，還可經「瀏覽窗」運用其它目錄內的指令或檔案(輸入時含路徑)。在「指令窗」內則包括了「開始功能表」及「索引功能表」，分別按照功能及字母順序將指令進行分類。使用者可自行在「常用窗/片語窗」加入一些常用的指令或片語，以提昇操作上的方便性。「環境窗」以圖示方式列出環境變數，讓使用者不必記憶變數或函數名稱。

以下我們以兩個實例介紹 TkBASH 的操作，實例中我們以方括號框起代表點選按鈕、圖示，或某種視窗操控動作。

實例 4 : TkBASH 在立即模式下的操作

在「立即模式」下，適合執行無參數的指令，使用者點選後，指令即刻被執行。例如我們可經滑鼠用 ps 指令查詢程序執行狀態： [切換到常用窗] [按 ps] 執行結果將顯示在「文字輸出窗」。

實例 5 TkBASH 在緩衝模式下的操作

在「單行緩衝模式」下，TkBASH 支援參數列並讓使用者利用「瀏覽窗」協助輸入檔名。我們可用 diff 指令比較目前工作目錄下的檔案 hello.cc 及另一目錄下的檔案/user/10/bclow10b/c/hello.c 的內容：
[切到常用窗] [按 diff 圖示] [切到工作窗] [按 hello.cc 圖示] [切到瀏覽窗] [改變目錄到 hello.c 所在位置][按 hello.c 圖示] [按送入鍵]

經上述滑鼠動作後，系統就會比較兩個檔案的內容，並將結果顯示在「文字輸出窗」。

3.2 TkBASH 系統實作

TkBASH 系統分為兩大部份：1.「客戶端應用程式」、2.「伺服器端服務程式」。「客戶端應用程式」是在 MsWindow 下執行，為 BASH 提供視窗介面。「伺服器端服務程式」則是在遠端 UNIX 平台的一個等待被連線的服務程式，負責處理客戶端所傳來的指令，並回應結果。使用者、客戶端應用程式、伺服器服務程式、BASH 之間的互動關係如圖 3 所示。其中實線箭號代表輸入、虛線箭號代表輸出。客戶端各個模組的功能介紹如下：

1. 「視窗操作介面」: 負責處理使用者的輸入並顯示輸出結果。
2. 「命令編碼模組」: 負責接收「視窗使用介面」所傳來的輸入訊息，編碼為對應的內部通訊命令，然後交給「網路通訊模組」負責傳出。
3. 「輸出解碼模組」: 負責解譯伺服器端傳來的的內部通訊命令，產生客戶端的輸出。
4. 「網路通訊模組」: 透過 socket 並以 TCP[13]的網路傳輸協定進行兩端的程式溝通。

在伺服器端，「網路通訊模組」功能與客戶端相同，其它模組功能如下：

1. 「命令解碼模組」: 負責判斷某個內部通訊命令屬於“轉接命令”(圖 3 實線箭號 A)或“遠端視窗命令”(圖 3 實線箭號 B)。若是轉接命令則解碼後轉送給 BASH 執行，若是遠端視窗命令則將通訊命令解碼後轉送給遠端視窗命令執行模組，
2. 「遠端視窗命令執行模組」: 負責執行遠端視窗命令(如[切換目錄]，[按下圖示]等)，然後再將執行結果送給「輸出編碼模組」進行處理。
3. 「輸出編碼模組」: 負責接收「遠端視窗命令執行模組」或 BASH 的執行結果，並封裝為內部通訊命令的格式，再由「網路通訊模組」傳送給客戶端程式。

TkBASH 的客戶端及伺服器端皆以 C++語言

[12]並且以物件導向方法實作。客戶端程式是設計在 MsWindow 下執行的應用程式，我們以 Microsoft Visual C++ 5.0 當作開發環境，視窗介面部份則使用 Microsoft Win32 SDK[4] 以及 MFC(Microsoft Foundation Class) 之規格。伺服器端程式在 FreeBSD 3.1 Release 環境下，以 GNU C++ 2.7.2.1 當作開發工具。目前 TkBASH 已可在 Microsoft Windows 98 及 FreeBSD 3.1 下正常執行。

四、總結與後續工作

在傳統的命令列介面下，使用者不能直接切換執行中的程式，不能直接看到處理中的檔案；以鍵盤輸入不但須強記指令、檔名，還容易發生打字錯誤。近年來的視窗介面操作簡單、方便，因而迅速取代了命令列介面。可惜目前視窗介面本身尚未支援命令列介面的許多重要功能。本論文分析命令列介面與目前視窗介面的功能，並為視窗介面提出具體的改進辦法，以下分為五點說明：

(1)現有視窗桌面缺乏標準輸出管道，我們建議添加「文字輸出窗」，以顯示使用者指令的回應及程式的輸出。文字輸出窗還應具備「捲軸」、「複製-貼上」的功能。

(2)桌面工作列上應增加階層式「索引功能表」，它自動容納預設路徑中的所有指令，讓使用者能經少數次按鍵取得指令。另外，桌面工作列上應有「指令歷程選單」，讓使用者能更方便地重複執行稍早前的指令。

(3) 對作業系統下命令時常需用到參數(parameters)。本文提出「單行緩衝模式」的視窗操作方式，使視窗操作也能輸入參數。命令參數常涉及冗長的檔名，所以應引入「工作窗」及「瀏覽窗」，讓使用者以滑鼠按鍵的方式迅速並正確地輸入。

(4)對例行或重複性的工作常需用批次處理(batch processing)，我們建議視窗介面應提供「多行緩衝模式」，讓使用者在緩衝區寫好批次指令後，就可直接交付執行。批次指令的撰寫過程中，使用者可以自由運用滑鼠點選的方式進行編寫。

(5)目前許多使用者在視窗介面透過終端機模擬程式(Telnet)，以命令列介面的方式操控遠端的 UNIX 平台。這種作法只運用鍵盤，因而失去視窗環境的便利性。我們認為 Telnet 應加入滑鼠、圖示窗這些便利的視窗操控功能。

依前述理念，我們實作了一個視窗操作介面(window-style shell)的雛型系統，稱作 TkBASH。TkBASH 的使用者透過網路操控遠端的 UNIX 主機。一方面可以充份運用便利的視窗操作方式，同時還可完全發揮操控環境(shell)的所有功能。在實務上，TkBASH 可以取代現有的終端機模擬程式(Telnet)及檔案傳輸程式(FTP)。

目前 TkBASH 只是一個實驗用的雛型系統，等發展成熟後，它的客戶端程式應該要取代整個視窗桌面，使它能同時操控自己的 MsWindow 平台和遠端的 Unix 平台。此外，我們也要讓 TkBASH 的客戶端程式移植到 Unix 平台，用它來取代 X-Window 的視窗桌面。等新系統完成後，我們將進行實證研究以探討這種操作環境的確實效益。

致謝

作者感謝審稿者提供卓越的見解.

參考文獻

[1] Arthur, L. J., *UNIX Shell Programming*, John Wiley & Sons, Inc. (1990).

[2] HopCroft, J. E. and Ullman, J. D., *Introduction to Automata Theory, Languages, and Computation*, Addison Wesley, Chap.7, (1979) .

[3] Kernighan, B. W., and Ritchie, D. M., *Thc C Programming Language*, 2nd Ed. Prentice-Hall, Section 5.10, (1988).

[4] Microsoft Corp., *Microsoft Win32 Programmer's Reference*, Vol. 1, Vol. 2, Microsoft Corporation, (1993).

[5] Microsoft Corp., *Microsoft Windows 95 使用手冊*, Microsoft Corporation, (1995).

[6] Microsoft Corp., *Microsoft Windows NT Server resource Kit*: for Windows NT Server 4.0, Microsoft Corporation, (1996).

[7] Microsoft Corp., *Microsoft Windows 98 入門指南*, Microsoft Corporation, (1998).

[8] The Open Group, *X Window Version 11 Release 6.4 Reference Manual*, ftp://ftp.x.org/pub/R6.4/xc/doc/hardcopy/man/man.PS.gz, (1998).

[9] Ramey, C., *BASH Reference Manual*, http://www.gnu.org/manual/bash-2.02/html_chapter/bashref_toc.html, (1998).

[10] Sethi, R., *Programming Languages Concepts and Constructs*, Addison Wesley, (1994).

[11] Silberschatz, A. and Galvin, P. B., *Operating System Concepts*, Addison Wesley (1994).

[12] Stroustrup, B., *The C++ Programming Lanauage*, 2nd Ed. Addison Wesley, (1991).

[13] Stevens, W. R., *UNIX Network Programming*, Prentice Hall, (1991).

[14] Symantec Corp., *pcANYWHERE 32 Version 8.0 User's Guide*, Symantec Corporation, (1997).

[15] Van Wolverton, *Running MS-DOS, 6th ed.*, Microsoft Press, (1993).

[16] 黃明達, *Windows 3.1 中文版入門*, 松崗電腦圖書資料股份有限公司 (1993).

Manuscript Received: Sep. 14, 1999
Revision Received: Jan. 21, 2000
And Accepted: Jan. 28, 2000