的值一樣是存在於sub bash之中，但接下來的test2.sh會在同樣的sub bash中執行。當執行完成後，不論是test.sh或test2.sh，都可以在sub bash中看到var1變數的值「123」，就可以成功列印出兩次「123」。

不過，在此要注意的是，雖然test.sh與test2.sh都可以看到var1，但是，要知道當程式結束後，是會回到bash的，而sub bash已經被關掉了，所以使用者是沒有機會再看到var1，因為使用者是在原本的bash上操作。

5-4 X Window

對X Window最基本的認識，就是要知道X Window在Linux下，是一個獨立的軟體，可以隨時視需求安裝或移除。也因為如此，很多管理者在安裝Server時，會選擇不要安裝X Window，因為會佔用不少的記憶體，而影響到整體的效能。像前一章所提到的效能測試，在CPU2006測試的環境中，就不會有人將Linux進入X Window後，再進行benchmark，因為這樣所得到的數據，絕對會比沒有進入X Window所得到的數據來得差。且大多數Linux所提供的功能，是不需要透過X Window就可以做到的，除非是各廠商另外提供的工具程式。

至於為何叫X？X Window名稱（或可稱為X與X11，11為版本名稱）的由來是因為原本在Unix下的Window系統是由史丹福大學所發展的，叫做「W」；到了1984年，由MIT的電腦科學實驗室將「W」的協定換掉，並將其改名為「X」，只是因為X在W字母的後面。

筆者一直有個很深的疑問，X Window為何無法推行？即使介面做的再漂亮、程式寫得再好，大部份較資深的Linux工程師大都還是會選擇以console的方式進行管理。但X Window是一個很成熟的軟體，也已經運行多年，為何一直無法打入人心？這是一個很大的問題，尤其是當筆者在使用X Window發現越來越多工具已經變為預設的產品，更是覺得X Window偉大的地方就在於默默經營。

一般會用X Window只有兩種可能，一是非常不熟悉X Window，二是Linux功力非常強。會造成這樣的結果，原因與結果不外乎以下幾個學習階段。

在不熟悉Linux的時候，勢必要倚靠X Window所帶來的便利性，大部份的系統設定、管理、進階服務設定，在X Window下都可以找到相對應的軟體。就算沒有，也可以在市面上找到類似的軟體（像Webmin），安裝後就可以透過Web或管理介面操作系統。但筆者認為，這樣的方式已然失去Linux最基本的概念。使用一個作業系統，若只是把該系統透過一個額外的介面做管控，萬一該介面失控時，該如何處理？當介面的更新速度無法趕上作業系統的更新，又要如何處理？所以，這並不是一個解決之道，反而會在問題產生時，帶來更大的瓶頸。

當使用Linux熟悉到一個階段，已可獨自作業並解決問題、處理各種疑難雜症時，這時已經知道X Window所帶來的問題，因此，會選擇不安裝或是不使用X Window，以避免這些問題的產生。可是，以一個系統管理者的角度，若不需要幫助別人也就算了，若有某些使用者硬是在Notebook或PC上安裝了X Window，遇到問題時，則會因為X Window不常使用，在解決問題時就會卡在一些基本的問題上（像部份顯示卡需搭配某些module的參數），這是很可惜的。

再來，就是Linux超強者的天下，對一個Linux從kernel到服務全面精通，且X Window架構完全掌握的使用者而言，X Window是一個非常好用的工具，因為所有的細節都可以完全掌握在自己手中，不像一般Windows較封閉的特性。很多有此特性的使用者，會連Notebook都安裝Linux為預設的作業系統，這其實在難度上是非常高的，因為X Window大部份的功能都有，一旦遇到問題，尤其是在Notebook上產生的，往往是X Window的死角，這需要靠長期使用的經驗與累積的技術背景才有辦法解決，因此，很多使用者才會一開始安裝好Linux，最後還是再換回Windows，原因就在於使用習慣及問題處理。

要如何善用X Window，必須先將Linux的基本運作及操作方式弄清楚，接著，就是針對X Window的整體架構有一個清楚的概念，這樣才可以在X Window的世界無往不利，以下就來介紹X Window的架構。

基本上，X Window是由三個部份組裝而成的：

■ X Server：主控台。

■ X Client：應用軟體。

■ Window Manager：視窗管理軟體。

從主要組成要件上可以看出，X Window其實是以Client-Server的架構所規劃的，因此，並不受限於主機與遠端的限制，但現在很少會用這樣的機制來做遠端的管理，因為其他像VNC等太方便了。當然在本章所介紹的，不論是X Window、X Server或X Client等相關軟體，都一樣會從開機後（X Window是從進入作業系統後才開始執行的）所執行的每一步，逐一介紹其作業流程與之間的關係。

5-4-1 | X重要元件

X Window不是由一個軟體所完成的，其中包含許多的元件、溝通協定、架構所組成，也因為如此，每個單一的軟體原則上都可以替換掉（有些會有相依性的問題），但也因為架構較複雜，且可用的元件太多（這好像是Linux的優點兼通病），常常造成使用者在觀念上有些混淆，以下幾個錯誤的觀念就常聽到：

X Window有分GNOME和KDE？

【實際情況】：其實GNOME和KDE是所謂的Desktop Manager。

X Client是Client端軟體？

【實際情況】：X Client是Server上安裝的軟體，Client端所需的是X Server。

X Window只能有一個主視窗（一個console）？

【實際情況】：X Windows可連續使用好幾個console。

startx和init 5是一樣的意思？

【實際情況】：startx和init 5執行的方式與過程都不一樣。

以上問題在本章都可釐清所有的觀念，接著，將介紹X Window的各大項說明。

X Server與X Client

在使用任何的Window系統，最重要的就是像滑鼠這一類的工具。在X Window中，X Server的目的就在於控制這些使用者經常使用的基本設備。在介紹X Server之前，我們先來看wiki上，一張針對X Server的架構圖（如圖5-49所示）。

在圖5-49中，X Server是X Client與滑鼠、鍵盤等週邊設備的介面，簡單的說，X Server的主要目的，就是幫使用者在電腦螢幕上繪出該出現的圖形。因此，任何在X Server上所執行的軟體，使用者都要透過X Server才可以和該軟體接觸，即使是執行一個console軟體（像xterm），使用者若要在上面輸入任何文字，其實也都是透過X Server才能將文字傳達到console上，所以，在啟動X Window時，若X Server出了問題，X Window是很有機會無法使用的，甚至連進入X Window的畫面都會有困難。

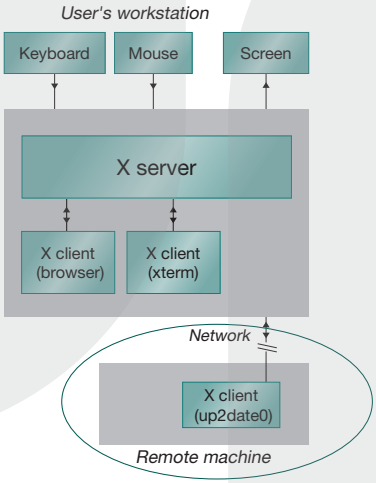


圖5-49：X Server與X Client示意圖
（資料來源：http://en.wikipedia.org/wiki/X_server）

圖5-49下方被框框所圈選的部份，則是強調在Client-Server的使用環境下，使用者是透過遠端連線使用遠端主機上的X Client，也就是說，遠端主機上只要有X Server負責呈現的部份，則X Client可以使用遠端主機上的軟體即可，這就是X Window所謂的Client-Server的架構，有點像是之前提的ASP（Application Service

Provider）的概念，只是作法不同。至於X Server與X Client、X Server與遠端的X Client如何溝通，則是透過其專屬的X Window System Core Protocol協定（如圖5-50所示）。

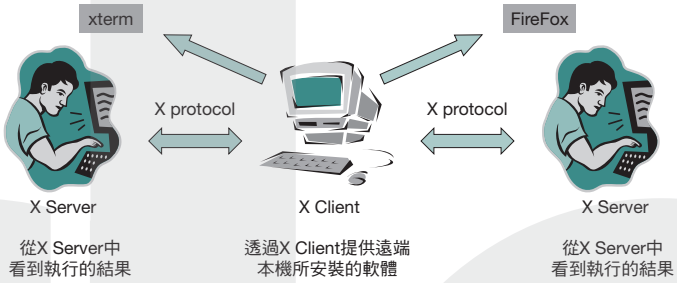


圖5-50：Client-Server架構下的X Client與X Server關係圖

X Window提供一個彈性很大的使用方式，不論是本機、遠端都可以使用，甚至在Windows下，只要可以支援Protocol，也可以使用遠端安裝在Linux中的X Client，真的是很方便的一個架構。美中不足的是，因為這是之前的協定，因此，是以圖片的方式不斷更新畫面，這在頻寬上是比較弱勢的，因為這種方式會佔用太多的網路資源。

在知道X Server與X Client的架構後，我們也應該要知道X Server在Linux下所對應的軟體是哪一個。目前Fedora所使用的X Server軟體，為Xorg組織所提供的。

X Server

在Linux下，「X」與「Xorg」這兩個執行檔（X是Xorg的連結檔，其實是同一支程式，也就是Xorg）都是啟動X Server的程式，只要console在執行X或Xorg後，預設會在第7個console啟動第1個X Server（如圖5-51所示），此時，主機上就已經有一個X Server可以使用了，只是這時的X Server並沒有很大的用處，因為只有環境而沒有軟體的配合，仍是沒有用的。

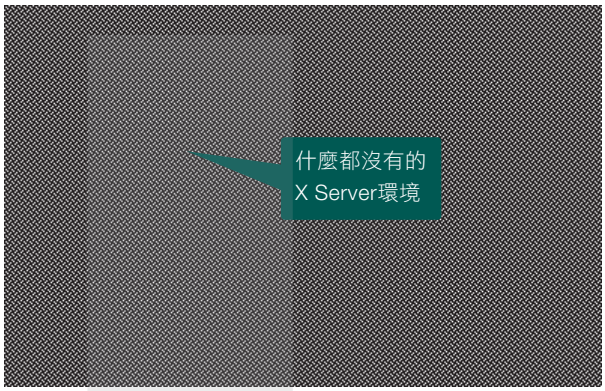


圖5-51：純X Server的畫面

在執行X或Xorg時，首先要知道的是，在一個Linux中可以同時存在好幾個X Server，因此，要先給所執行的X Server一個編號，像「X：2」，就是編號為2的X Server。當下面介紹的X Client軟體執行時，就可以指定該軟體要在哪一個X Server環境下產生，如此就可以分辨哪一個軟體在哪一個X Server中存在了。

要特別注意的是，很多使用者會誤會該「X Server編號」等於「console的順序」。其實不然，在X啟動時，預設的編號會從「0」開始計算，並以第「7」個console（也就是【Ctrl+Alt+7】）產生第一個X Server，而第二個X Server則會在第8個console，依此類推。但編號為1不代表會在第8個console，因為X Server的編號只是該X Server的名稱，並不是console的順序，這一定要弄清楚。

■ X Client

X Client就有非常多的軟體可以使用，最簡單的就是xterm了。只要在有X Server的環境下，先回到command line，在指令模式下輸入「xterm –display:0」，就可以將該xterm軟體在編號為0的X Server上執行（如圖5-52所示）。

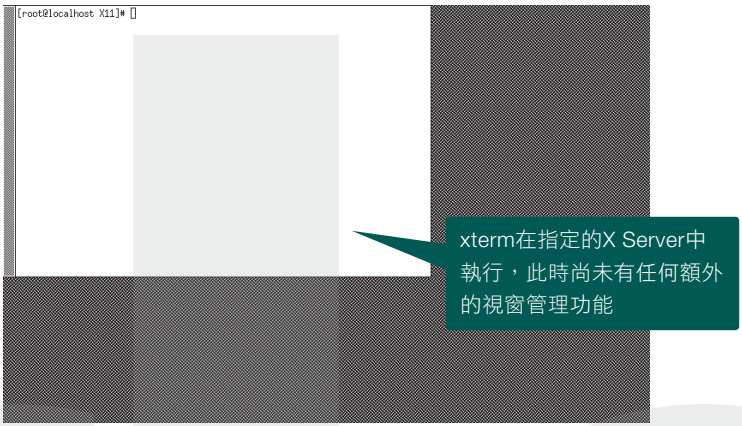


圖5-52：xterm執行後的畫面

這時候xterm已經可以使用，但無法搬動或更改任何視窗的設定，只是一個很簡單的介面，供使用者輸入所希望的指令，以及用滑鼠的左、右鍵將xterm內容上下移動，這是因為視窗的移動或設定是由另一種類別的X軟體所負責，也就是Window Manager，下一小節將會有詳細的介紹。

Window Manager

什麼是Window Manager？何時又該使用Window Manager？Window Manager可以提供什麼服務？從上一節的例子知道，光有X Server是不夠的，X Server僅啟動一個背景環境供所有需要使用的軟體使用，所以在X啟動後，就算是將軟體執行在上面，也無法給使用者一個方便的環境（像視窗的拖拉、大小的更改等），因此，這時候就需要Window Manager上場。

前一節已經介紹到X Server與X Client兩者共處的情況（相當單純的一個環境），接著看一下，當Window Manager介入之後，對整個X Window會有怎樣的影響？Fedora 7已經預設提供一支叫做twm的Window Manager，預設的設定檔也已存放在「/etc/X11/twm」目錄中，使用者可以在X Server及X Client存在的環境中（其實只要有X Server就可以了）執行twm（記得要配合X Server的編號），便會提供所指定的X Server及X Client較彈性的介面（如圖5-53所示）。

```
[root@localhost X11]# X :1&
[root@localhost X11]# xterm -display :1&
[root@localhost X11]# twm -display :1 -f /etc/X11/twm/system.twmrc &
[root@localhost X11]# _
```

圖5-53：twm的執行方式

圖5-54：當X Window使用到WM的畫面

每一種Window Manager所提供的樣式與功能也不盡相同，像GNOME預設的Window Manager為metacity（如圖5-55所示）。和圖5-54一樣的xterm軟體，但套用在不同的Window Manager後，就有完全不一樣的風格（如圖5-56所示），這就是為何Linux可以用不一樣的造型，呈現在使用者面前，因為只要很簡單的將Window Manager換掉，就是全新的版面了。

```
[root@localhost ~]# metacity --display=:0&
[2] 10259
[root@localhost ~]# _
```

圖5-55：metacity的執行方式

圖5-56：metacity所呈現的視窗感覺

不論是哪一套軟體，當Window Manager執行後，整個畫面變得生動、漂亮多了（純粹和剛才圖5-52的畫面做比較），馬上可以很明顯感受到Window Manager的作用，就是在將所有執行中的各視窗做管理的功能，使用者也才能夠在目前的X Window下如此的方便。當然，各位手上所使用的並不是twm，而是GNOME內建的Metacity。目前市面上常聽到的Window Manager，除了預設的twm外，大致有以下幾種，如表5-4：

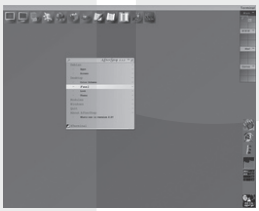
Window Manager 名稱	圖形介面
■AfterStep 蠻有名的一支Window Manager	
■Sawfish GNOME在2.2版本之前所預設使用的Window Manager	

表 5-4：常看到的 Window Manager 種類（圖片來源：wiki 各 Window Manager 之介紹）

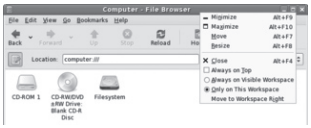
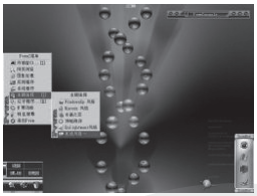
Window Manager 名稱	圖形介面
■Metacity GNOME目前預設的Window Manager	
■FVWM 從twm發展出來的版本	
■KWin KDE的預設Window Manager	

表5-4：常看到的Window Manager種類（續）
（圖片來源：wiki各Window Manager之介紹）

Widget

有了X Server、X Client及Window Manager就可以達到目前一般使用的環境嗎？在上一節的Window Manager基本畫面中，我們看到視窗的畫面已經被加入一些顏色及按鈕或圖案，這些是Window Manager自己的東西嗎？其實不是的，那到底是什麼呢？相信有撰寫過Windows視窗軟體經驗的人就會知道，在寫一個GUI介面時，都會需要設計友善的使用者介面，否則又會被質疑好不好用、有沒有價值？一個好用的介面往往勝過強大的功能。不要忘了，科技始終來自於人性。所謂的友善介面，包含該軟體選項的畫面、選項的按鈕、滾動軸的樣式等等，這些都需要有一個圖形的函式庫，也就是Widget來提供的。

你可能沒看過Widget這個名詞，但另一個名稱就會引起大家的注意，那就是GNOME下的GTK+以及KDE下的QT。以Fedora為例，GNOME的使用者在安裝某些軟體或是使用X Window時，有時會看到有關gtk的訊息（部份的套件如圖5-57所示），因為只要在GNOME下面執行的軟體，都是使用GTK+的；換句話說，若使

用者要以KDE為預設的Desktop Manager，就會變成全部使用QT，因此，這兩種Widget在GNOME與KDE間是不能夠混用的，因為當初開發的程式就已經決定要用GTK+還是QT。

```
[root@localhost ~]# rpm -qa|grep gtk | head -5
gtk2-engines-2.10.0-3.fc7
gtkhtml2-2.11.0-4
gtkspell-2.0.11-3.fc7
GConf2-gtk-2.18.0.1-2.fc7
pygtk2-2.10.4-2.fc7
[root@localhost ~]# _
```

圖5-57：部份GTK的相關RPM套件

反向思考，若Window Manager沒有像GTK+或QT的底層配合，就無法完成整個Window的視覺效果，也因為像metacity使用GTK+，才能做出目前GNOME上看到的每一個視窗外觀。

Display Manager

Display Manager並不是一個絕對需要的套件，以上介紹的各相關套件，就已經可以組成一個X Window；但對整個X Window來說，採用Display Manager與不採用，最大的差別就在於以下兩點：

■ 提供使用者登入的畫面：GNOME預設使用的Display Manager為gdm，當使用者啟動Fedora 7進入X Window後，首先看到的，就是登入畫面（如圖5-58所示）。在這個畫面中，除了登入帳號、密碼之外，還有供使用者選擇一些像語系之類額外的設定。

圖5-58：gdm的登入畫面

■幫X Server建立session：Display Manager的另一個主要任務，就是在啟動X Server時，建立一個對X Server的溝通管道，也就是session。這session的建立包含本機與遠端的X Server，因此，當之前所提到遠端使用X Client時，其實就是透過Display Manager與各X Client做溝通（如圖5-59所示），所以當使用者本地端的X Server啟動，就可以透過和遠端Display Manager建立session，再透過該session和遠端的各X Client軟體，便可以將其畫面與結果，呈現在本地端的使用者面前。

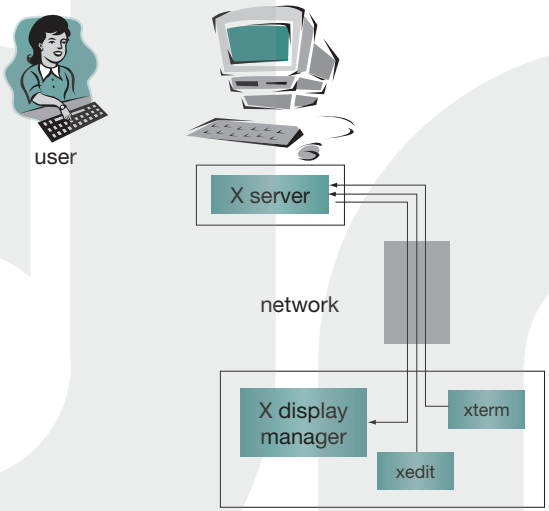


圖5-59：Display Manager概念圖
（資料來源：http://en.wikipedia.org/wiki/Xdmcp#X_Display_Manager_Control_Protocol）

熟悉session這個名詞的人，一定具備一個基礎概念，通常建立session即表示會有相對應在該session上溝通的protocol。沒錯，既然X Server與Display Manager之間要進行溝通，就會有制定的protocol，也就是XDMCP（X Display Manager Control Protocol）。在本地端要不要使用XDMCP，使用者可自行決定（當直接使用X時，就不會透過Display Manager，也就不會使用到XDMCP），但若是遠端Client-Server的架構下，就必須要支援XDMCP才可以做連結的動作。

但這個protocol有一個缺點，因為XDMCP是以telnet的方式，在UDP的port177（當Display Manager下的XDMCP功能被開啟時，會自動將這個port打開）上傳輸使

用者的所有視窗，因此，也代表沒有加密的功能，所以使用者在做遠端連線的同時，要切記在上面所傳輸的資料，任何人都有機會看到。

Desktop Manager

這麼多的套件、管理程式，使用者在一個一個安裝時很容易會產生問題。就像Linux一樣，Linux本身只是一個kernel，但經過Red Hat或SuSE等廠商包裝完之後，變成一個具漂亮安裝介面、很多實用工具（廠商可選擇要放入哪些套件）、不同的X Window風格的作業系統。相同的，剛剛所提到的這些X Server、X Client、Window Manger及Widget，分開來用對使用者十分不方便，如果有人願意將這些東西包在一起，就可以變成非常簡單好用的Window介面。

Desktop Manager就是這樣的一個角色，GNOME是兩大主流的其中之一（另一個是KDE），這兩大組織為使用者從啟動、登入、管理、安裝都已經寫好一連串的軟體，只要使用者安裝好，就可以享受這些方便性。因此，如果說GNOME是X Server也不對；是Window Manager也不對。其實GNOME是這些軟體、介面再加上自行開發的介面（像登入所使用的gdmgreeter）之組合套件（如圖5-60所示），因而稱為Desktop Manager。

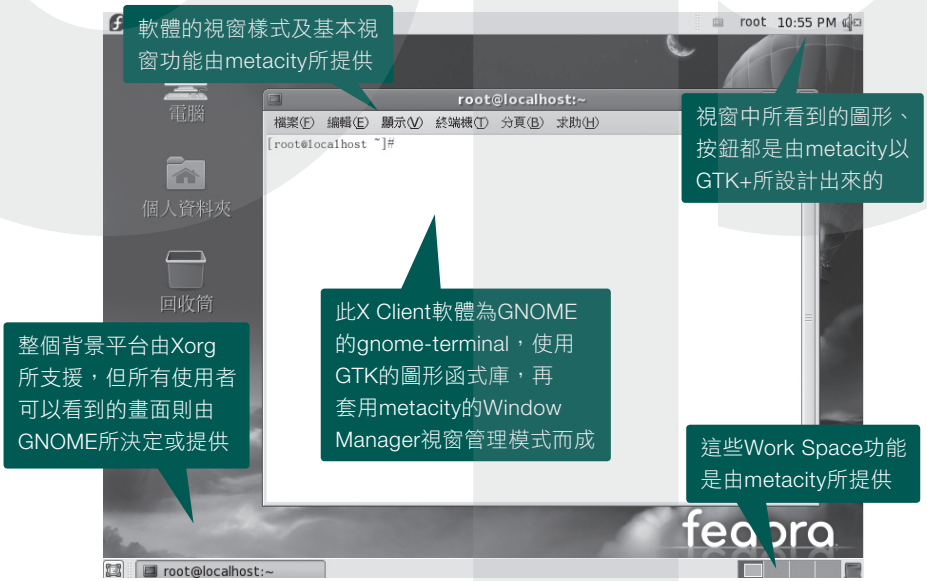


圖5-60：X Window的總體概念

圖5-60可以讓我們以整體的概觀來看X Window，了解整個X Window的視窗到底是由哪些軟體組合而成，讓每位X Window的使用者都可以方便使用。使用者所看到的畫面或細部結構，其實都是透過不同的軟體結合而成，這也是GNOME或KDE這樣的Desktop Manager好用之處。

5-4-2｜進入X Window流程之差異

在進入系統的另一個重點，其實還包含了X Window的啟動，因為一般預設啟動的runlevel為5，也就是X Window模式。因此，在進入Linux作業系統的最後一個步驟（如果inittab的預設值是5），就是啟動X Window。

X Window大致有以下兩種啟動的方式：

- startx
- init 5

一般或許會認為這幾種方式都一樣，不過就是進入X Window嘛！但以下的介紹就是希望可以澄清這個觀念，因為這兩個步驟除了最後呈現的畫面是一樣的，其執行方式與過程卻是全然不同的。

startx

在runlevel 3的文字介面中，若想要將X Window啟動，最一般的指令就是執行startx，其實startx只是xinit指令的前置作業。xinit是一支負責啟動X Window的軟體，startx功用說穿了，只是幫使用者更方便的透過xinit將X Window呼叫起來。

xinit若單獨來看，是一個非常簡單的軟體，預設只負責將X Server啟動，之後就將xterm執行在該X Server環境中（如圖5-61所示），便功成身退。因此，xinit主要是一個啟動X Server的軟體。

```
7345 tty1     S+     0:00 xinit
7346 tty7     S<s+   0:00 X :0
7352 ?        Ss     0:00 xterm -geometry +1+1 -n login
```

圖5-61：將xterm執行在該X Server環境中