前言

文字編輯是任何電腦系統中最常見的任務之一，vi 是系統上最有用的標準文字編輯器之一。你可以使用 vi 建立新檔案或編輯任何現有的純文字檔案。

vi 與早期 Unix® 開發的許多經典應用程式一樣，有著難以駕馭的說法。Bram Moolenaar 強化 vi 版本成為 Vim 後，這個說法逐漸消除。Vim 包含了需許多的便利性、視覺化指引和螢幕協助。

時至今日，Vim 已成為更最普及的 vi 版本，因此本書將重點放在以下幾個面向：

- 第一部分，「vi 與 Vim 的基礎」，將說明基本的 vi 技巧，適用於 vi 所有版本中，而在 Vim 亦是如此。
- 第二部分，「Vim」，有很多章節專門介紹 Vim 的進階特性。
- 第三部分，「大環境中的 Vim」，介紹更多 Vim 的使用環境。

本書範圍

本書共有 17 個章節和 4 個附錄，分成四個部分。第一部分「vi 與 Vim 的基礎」，目的在讓讀者快速開始使用 vi 和 Vim，並有效使用進階功能。

前面兩章，第一章「vi 與 Vim 簡介」、第二章「簡單的文字編輯」提供一些簡單的編輯指令，給供初學者上手。應該多加練習，直到熟悉為止。在第二章學到一些基礎的編輯指令操作後，可以稍做休息。

然而 vi 不僅只有做一些基礎的文字編輯而已；它的各種命令與選項，都可以簡化大量的編輯工作。第三章「快速移動位置」與第四章「越過基礎的藩籬」，專注於更簡單的

碁峰 www.gotop.com.tw

方法來完成任務。第一次閱讀時，只需要大致瞭解 vi 與 Vim 可做的事情，以及哪些命令可能對你特別有用。之後，可以隨時回到這些章節，做更深入的學習研究。

第五章「ex 編輯器簡介」、第六章「全域代換」與第七章「進階編輯」，提供一些可將許多繁重編輯工作交給電腦的協助工具。其中介紹位於 vi 底層的 ex 行編輯器，並且示範如何在 vi 中使用 ex 的命令。

第二部分「Vim」，描述 Vim 在 21 世紀中，是如何成為最流行的 vi 複製版本。其中會詳細介紹 Vim 相對原始 vi 所具有的許多功能。

第八章「Vim：對 vi 的改進與簡介」，提供了對 Vim 的一般介紹。本章會綜觀 Vim 對 vi 的主要改良，像是內建輔助說明，初始化控制，附加的動作命令，可延伸的正規表示式，以及其他部分。

第九章「圖形化 Vim（gvim）」，檢視現代 GUI 環境中的發展，例如商用 Unix 系統上的標準，GNU/Linux 與其他 Unix 類似的產品，以及 MS Windows。

第十章「Vim 多視窗功能」，著重在多個視窗下的編輯，或許這是對標準 vi 最重大的附加功能。本章提供所有建立與使用多個視窗的細節。

第十一章「Vim 為程式設計師強化的功能」，重點介紹 Vim 作為程式設計師的編輯器，超越一般文書編輯的能力。特別像是摺疊與大綱功能、智慧縮排、語法特別標示與「編輯 – 編譯 – 除錯」週期的加速。

第十二章「Vim 指令稿」，深入探討 Vim 的命令語法，可撰寫或修改自訂的指令搞，以符合需求。Vim 大部分的簡便特性是因為來自其他用戶已經編寫好的大量腳本，並且整合於其中，為 Vim 發行版做出了貢獻。

第十三章「其他好用的 Vim 功能」，這章涵蓋了前面幾章有趣的部分，但不適合放在稍早章節的要點。

第十四章「一些 Vim 更強大技術」，展現一些基於個人化鍵盤重新映射的有用技術，提高更多生產力的方法。

第三部分「大環境中的 Vim」，更廣泛地檢視 vi 與 Vim，在軟體開發和計算機世界所扮演的角色。

第十五章「Vim 作為 IDE 所需要的組裝需求」，會觸及到一些 Vim 外掛，不過那只是冰山一角，而重點將說明如何將 Vim 從「單純」的文字編輯器轉變成一個成熟的整合開發環境（IDE）。

碁峰 www.gotop.com.tw

第十六章「vi 無所不在」，將探討其他那些帶有 vi 風格的軟體環境，來發揮更高的生產力。

第十七章「結語」，為本書做最終的總結。

第四部分「附錄」，提供有用的參考材料。

附錄 A「vi、ex 和 Vim 編輯器」，列出標準 vi 與 ex 命令，依照功能排序。還提供按字母順序排列的 ex 命令列表。也包括從 Vim 中可用的 vi 和 ex 命令。

附錄 B「設定選項」，列出 vi 與 Vim 的設定選項。

附錄 C「vi 輕鬆的一面」，呈現一些與 vi 相關的有趣主題。

附錄 D「vi 和 Vim：原始碼和建置」，說明在 Unix、GNU/Linux、MS-Windows 或 Macintosh 等系統下，如何取得 vi 與 Vim。

本書的寫作方式

我們的想法是讓讀者對於 vi 與 Vim 有一個好的綜觀以及對於一個新手所需要的基本知識。學習一個新的編輯器，尤其具備眾多選項的 Vim 編輯器，似乎是一項艱鉅的任務。我們已經努力將內容，用淺顯易懂且合乎邏輯的方式，來呈現基本的概念與命令。

在說明完 vi 與 Vim（任何版本都能適用）的基礎後，我們將繼續深入介紹 Vim。接下來說明本書使用的編排慣例。

vi 命令的討論

對於每個鍵盤命令或一組相關群組命令，會先對主要觀念做一段簡短介紹，然後段落說明各個項目。接著，描述在不同情況下，提供適合的命令和使用的正確語法。

本書編排慣例

在語法的描述和範例中，需要實際打出的字以 Ubuntu Mono 字型表示，命令名稱與程式選項也是。變數（不會直接打出來，而是用實際值來代替的字）則是用 *Ubuntu Mono Italic* 表示。中括號表示這個變數為可選擇的項目。例如，以下這行語法：

```
vi [filename]
```


碁峰 www.gotop.com.tw

其中 *filename* 會用到實際的檔名來替代。中括號表示命令 vi 可以忽略不必加上檔名。中括號本身不必輸入。

某些範例會顯示在 shell 提示符號下，輸入命令而產生的結果。其中，實際的輸入文字會用 **Ubuntu Mono bold** 來呈現，以便與系統回應的結果做區分。例如：

```
$ ls
ch01.xml ch02.xml ch03.xml ch04.xml
```

在程式碼範例中，以斜體表示註解，不必輸入。在內文中，也會以斜體指出檔名、引用特殊術語，並以楷體強調其他事情。

依循傳統 Unix 文件慣例，*printf*(3) 這類格式參考到線上手冊（可透過 man 命令取得）。這個例子參考到手冊第三節的 printf() 函數（在大多數系統中，可透過輸入 man -s 3 printf 來取得說明）。

按鍵

特殊按鍵會顯示在一個方框之中。例如：

```
iWith a ESC
```

在整本書中，還將看到 vi/Vim 的命令列表與其結果：

按鍵順序	結果
ZZ	"practice" [New] 6L, 104C written 輸入 ZZ（寫入並存檔的命令）後，檔案會儲存成一般的磁碟檔案。

在前面的範例中，命令 ZZ 顯示在左側欄位中。在右側欄位是螢幕中，顯示命令回應一行（或多行）的結果。在這種情況下，由於 ZZ 儲存並寫入檔案，將會看到寫入檔案時顯示的狀態列；而游標並未顯示。命令與結果的下方是對命令及其作用的說明。

其中一些範例中，我們也會呈現 shell 命令及其結果。在這種情況下，命令前面是標準 shell 的提示符號 $，命令以粗體顯示：

按鍵順序	結果
$ ls	ch01.asciidoc ch02.asciidoc ch03.asciidoc

有時透過同時按下 CTRL 鍵和另一個鍵來呼叫發出 vi 命令。在文字中的組合按鍵，通常寫在一個框內（例如，CTRL-G）。在程式碼範例中，它會是利用在按鍵名稱前加上

碁峰 www.gotop.com.tw

插入符號（^）來撰寫的。例如，^G 表示同時按住 CTRL 再按下 G 鍵。我們使用大寫字母（^G，而不是 ^g）來操作控制字元，這是一般的約定，即使在輸入控制字元時是**不需要**按住 SHIFT 鍵[1]。

另外，當我們使用按鍵表示要呈現大寫字母時，對任何字元 *X* 執行 SHIFT-X。因此，a 表示為 A，而 A 表示為 SHIFT-A。

注意事項、重點與提示

這個圖示代表一個警告性說明。它描述了需要注意或小心的事情。

這個圖示代表一般注意事項。它指出可能感興趣或可能不明顯的事情。

這個圖示代表一個提示或建議。它提供有用的快速方式或可節省時間的事情。

問題確認事項

某些章節會包含一些問題與解決的方法，可以暫時跳過，之後有需要時再回來參考。

預備知識

本書假設你對 Unix 的使用已經有基礎。你應該知道的有：

- 在電腦或工作站啟動終端機，進入 shell 命令介面
- 使用 ssh 軟體登入與登出遠端系統
- 執行 shell 指令
- 切換目錄
- 列出目錄中的檔案

1 這可能是因為鍵盤的按鍵是大寫字母，而不是小寫字母。

碁峰 www.gotop.com.tw

- 建立、複製和移除檔案

熟悉 `grep`（一個全域搜尋的程式）和萬用字元，也很有幫助。

雖然現今系統環境可以由圖形使用者介面（GUI）下執行 Vim，卻失去了使用 Vim 命令列選項所提供的靈活性。因此，在整本書中，我們的範例會持續示範，如何從命令列提示符號下執行 `vi` 和 Vim。

使用範例程式碼

你可以在 *https://www.github.com/learning-vi/vi-files* 下載補充材料（程式碼範例、練習等等）。

如果你在使用程式碼範例時，遇到技術上的問題或困難，請發送 email 至 *bookquestions@oreilly.com*。

本書目的在幫助你完成工作。一般來說，如果本書提供程式碼範例，可以在你的程式和文件中使用它。除非你要複製程式碼的重要部分，否則不需要聯繫我們來取得許可。例如，使用本書中的多個程式碼區塊來編輯一個程式，這是不需要許可。販售或發行 O'Reilly 書籍中的範例，確實需要取得許可。引用本書和範例程式碼來回答問題是不需要許可。將本書中的大量範例程式碼合併到你的產品文件中，也確實需要許可。

我們感謝你標示引用資料來源，但這並不是必要的。來源的標示通常包括書名、作者、出版商以及 ISBN。例如：「*Learning the vi and Vim Editors* by Arnold Robbins and Elbert Hannah (O'Reilly) Copyright 2022 Elbert Hannah and Arnold Robbins, 978-1-49207880-7.」。

如果你認為程式碼範例的使用不屬於合理使用或上述範圍中，請隨時透過 *permissions@oreilly.com* 與我們聯繫。

關於之前的版本

在本書的第五版（當時稱為《精通 `vi`》）中，首先全面地討論 `ex` 編輯器命令。在第五、六、七章中，增加更多範例來闡明 `ex` 和 `vi` 的複雜特性，涵蓋諸如正規表示式語法、全域代換、*.exrc* 檔案、單字縮寫、鍵盤映射等主題，以及編輯指令稿。其中一些範例來自 *UnixWorld* 雜誌的文章。Walter Zintz 在 `vi` 上寫了一個由兩部分組成的教學，

碁峰 www.gotop.com.tw

教授一些我們不知道的東西，還有很多聰明的例子，來說明我們已經在書中介紹過的特性[2]。Ray Swartz 也在他的一個專欄中也貢獻一個有用的技巧[3]。

《精通 vi（第六版）》介紹四種免費的「複製」或類似運作方式的編輯器。其中有許多都比原來的 vi 有所改進。因此可以說有一個 vi 編輯器「家族」，本書的目標是教你使用它們需要知道的知識。書中同樣描述 nvi、Vim、elvis 和 vile。在新的附錄中描述了 vi 在更大的 Unix 和網際網路文化中的地位。

《精通 vi 和 Vim 編輯器（第七版）》保留了第六版的所有優點。時間已經證實 Vim 是最受歡迎的 vi 複製版本，因此第七版加大 Vim 編輯器所佔的章節篇幅。但是，盡可能滿足更多的使用者，保留並更新有關 nvi、elvis 和 vile 的內容。

關於第八版

本書保留了第七版的所有優點。Vim 現在已經「稱霸」，所以這個版本更新 Vim 在書中所涵蓋的範圍，並刪除了關於 nvi、elvis 和 vile 的內容。在第一部分，現在使用 Vim 作為指令和範例的內容。此外，已經刪除對舊版本 vi 中，不再引述相關的問題。我們試圖精簡本書，並儘可能保持它的相關性和實用性。

新增內容

在新版本的內容增加如下：

- 我們再次更正基本文字中的錯誤。
- 我們對第一部分和第二部分中的內容，進行徹底的修改和更新。在第一部分中，我們將重點從 Unix 原始版本的 vi 轉移到「Vim 中的 vi」。我們還在第二部分，增加了一個新章節。
- 第三部分附加全新的章節。
- 我們改變附錄 C 的重點。
- 我們已將有關取得和建置 Vim 的說明，從主要內容文字移至附錄 D。
- 其他附錄也一併更新。

2 《vi Tips for Power Users》*UnixWorld*，1990 年 4 月；和《Using vi to Automate Complex Edits》*UnixWorld*，1990 年 5 月。兩篇文章均由 Walter Zintz 撰寫。

3 《Answers to Unix》*UnixWorld*，1990 年 8 月。

碁峰 www.gotop.com.tw

第一部分

vi 與 Vim 的基礎

第一部分，安排讓讀者能快速的開始使用 vi 與 Vim 編輯器。提供一些進階技巧，可以更有效率使用它們。以下章節將涵蓋最原始、最核心的 vi 功能與命令，可以在任何版本上使用的命令。後面章節將介紹 Vim 進階技巧的特性。這部分包含以下章節：

- 第一章，vi 與 Vim 簡介
- 第二章，簡單的文字編輯
- 第三章，快速移動位置
- 第四章，越過基礎的藩籬
- 第五章，ex 編輯器簡介
- 第六章，全域代換
- 第七章，進階編輯

碁峰 www.gotop.com.tw

第一章

vi 與 Vim 簡介

電腦最重要的日常用途之一，就是文字處理：撰寫新文字、編輯和重新排列現有文字、刪除或重寫不正確且過時的文字。如果使用過文字處理程式，例如：手邊的 Microsoft Word。亦或是程式設計師的你，也是在處理原始碼裡的文字內容，以及開發時所需的輔助文件。文字編輯器處理任何文字內容的檔案，無論這些檔案是否包含數據資料、原始碼或寫作句子。

本書內容是關於兩個相關文字編輯器 `vi` 與 Vim 的使用。`vi` 是在標準 Unix 上作為傳統悠久的文字編輯器[1]。而 Vim 建立在 `vi` 的命令模式與命令語言之上，提供比原來快一倍以上的能力。

文字編輯器和文字編輯

讓我們開始吧！

文字編輯器

Unix 文字編輯器隨著時間的推移而發展。最初的是**行編輯器**（line editor），例如：`ed` 和 `ex`，用於連續進紙的串列終端設備上列印。（是的，真是這樣進行作業，至少作者也曾如此。）之所以稱為行編輯器，是因為程式每次處理都僅限於一行到數行之間。

1 如今「Unix」一詞包含源自原始 Unix 的商業系統，以及開放原始碼的類 Unix 系統。Solaris、AIX 和 HP-UX 是前項的代表，GNU/Linux 和各種衍生於 BSD 系統是後項的代表。也泛指 macOS 的終端環境、在 MS-Windows 上，適用於 Linux 的 Windows 子系統（WSL，Windows Subsystem for Linux），以及 Cygwin 和其他類似 Windows 的環境。除非另有說明，本書中的所有內容全面適用於這些系統。

碁峰 www.gotop.com.tw

隨著可呈游標位置的黑白螢幕設備（cathode-ray tube；CRT）的推出，行編輯器演變成**螢幕編輯器**（screen editor），例如：`vi` 和 Emacs。螢幕編輯器可以一次全螢幕處理檔案，可輕鬆的在行與行之間移動，畫面上的變化正如我們所期望。

接著，圖形使用者介面（GUI）環境的導入，螢幕編輯器更進一步演變成圖形化文字編輯器，可以在其中使用滑鼠捲動，檢視檔案的一小部分，並且移動到檔案中的特定位置，然後選擇所要的文字上執行操作。以上大多是以 X Window 系統上的文字編輯器來說的，若在 Gnome 的系統上則是 `gedit`，而在 MS-Windows 上是 Notepad++。還有其他的。

特別感興趣的是，流行的螢幕編輯器已經演進到圖形化編輯器[2]。如：GNU Emacs 提供多個 X Window，而 Vim 則是 `gvim`。即便如此，圖形化編輯器依舊與原有螢幕編輯器有著相同的運作模式，使用 GUI 版本的編輯器沒有太大的區別。

在 Unix 系統上的所有**標準**編輯器當中，`vi` 是最有用的第一首選[3]。與 Emacs 不同的是，它在每一個近代版本的 Unix 上，都幾乎以相同的形式在系統裡出現，因此從單純的文字編輯器，變成文字編輯的一種通用語[4]。也可以說，相較 `ed` 和 `ex`，與之後變化的螢幕編輯器、圖形化編輯器，後者更容易使用。（事實上，行編輯器大多數已經棄之不用）

`vi` 存在多個化身。有原始的 Unix 版本，還有多個「複製」版本：從頭開始撰寫可以像 `vi` 一樣執行的程式，但不是出自於原始 `vi` 的原始碼。其中，Vim（*https://www.vim.org/*）是最受歡迎的版本。

在第一部分的章節中，帶給讀者 `vi` 的一般概念。在這部分每個章節中，所提到的內容，都適用於所有 `vi` 版本。然而，我們以 Vim 做為本書內容使用的依據；因為，這比較可能出現在目前系統的版本之中。在閱讀時，可將「`vi`」想像成為標準的「`vi` 與 Vim」。

`vi` 是視覺化編輯器（*visual* editor）的縮寫，讀作「vee-eye」。請參考，圖 1-1。

2 或許跟神奇寶貝一樣？

3 如果還沒有安裝 `vi` 或 Vim，請見附錄 D：vi 與 Vim 原始碼與編譯。

4 GNU Emacs 已經成為 Emacs 的通用版本。唯一的問題是它在大多數的系統中不是標準的；必須自行取得和安裝，即使在某些 GNU/Linux 系統上也是如此。

碁峰 www.gotop.com.tw

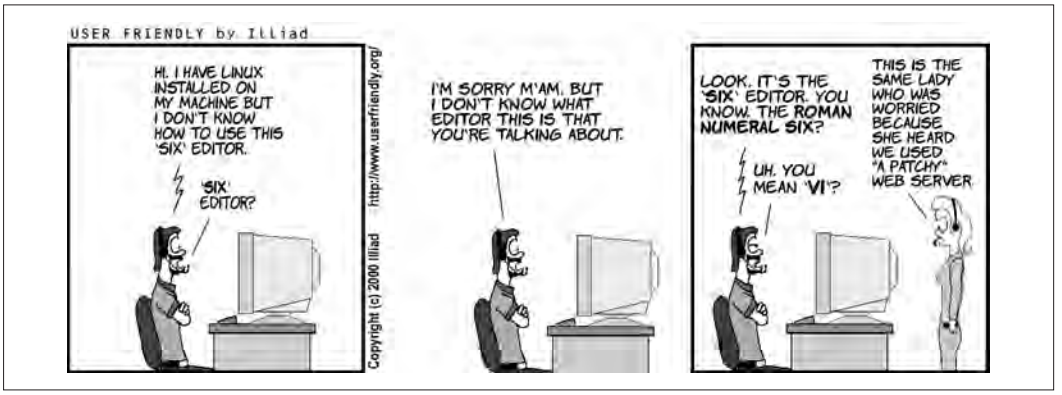

圖 1-1　vi 的正確發音

對於許多初學者來說，vi 看起來不直觀又笨重。它不用特殊控制鍵作為文字處理的功能鍵，讓我們能夠正常輸入，反而使用一般常用按鍵來執行命令。當按鍵用於執行命令時，vi 處於「**命令模式**」（*command mode*）。必須先進入「**插入模式**」（*insert mode*），才能輸入實際的文字。而且，命令多如牛毛。

然而，一旦當我們開始學習之後，將瞭解這個編輯器的確經過精心設計。只需要按幾個按鍵，便可以完成複雜的工作。在學習 vi 過程中，我們可以把編輯工作逐漸交給電腦；這本來就是電腦的工作。

vi 與 Vim（如同任何文字編輯器一樣），並非是一個「所見即所得」的文字處理器。如果需要產生格式化文件，需要輸入特定指令；有時稱做**格式化代碼**（*formatting code*），會使用個別的格式化程式來控制輸出的結果。以縮排數段文字為例，你需要在縮排的開始與結束的地方插入代碼。格式化代碼讓你嘗試或變更文件顯示的結果，相較文字處理器，對於視覺上的呈現有更好的控制。

格式化代碼通常被視為是一種**標記式語言**（*markup language*）的動詞[5]。近幾年來，標記式語言重新流行起來，其中 Markdown 和 AsciiDoc 較為值得注意的[6]。而用於網際網路上，建構網頁中的超文本標記語言（HyperText Markup Language，HTML），也許才是當今更為廣泛使用的標記式語言。

5　源自於排版與校對時，用紅色鉛筆「標記」修改的變化而來。

6　關於這些語言的更多資訊，可參考 https://en.wikipedia.org/wiki/Markdown 與 http://asciidoc.org。本書使用的是 AsciiDoc。

碁峰 www.gotop.com.tw

除了剛才提到的標籤語言，Unix 支援 troff 格式化套件[7]。Tex（*http://www.ctan.org/*）與 Latex（*http://www.latex-project.org*）也是很常使用的格式化程式。使用這些標籤語言的最簡單方式，就是使用文字編輯器編輯。

vi 支援一些簡單的格式化機制。例如，可以要求它在一行結束時自動換行，或是自動縮排進新的一行。此外，Vim 提供自動化拼自檢查。

如同任何技能一樣，當編輯進行的越多，這些基礎知識就變得越容易，你可以完成的工作就越多。一旦習慣了編輯時所擁有的功能，可能永遠不會想回到任何「簡易」的編輯器。

文字編輯

文字編輯的工作有那些部分？首先，你想要**插入**文字（一個忘掉的或是句子），接著會**刪除**文字（錯字或整個段落）。你也會想要**變更**文字或句子（變更錯字，或是改變某個用詞）。也可能將文字從檔案的一處**移動**到另一處，或是**複製**到另一處。

但 vi 的初始狀態與其他文書處理程式不一樣，它的預設狀態是命令模式。只需要幾個按鍵，就可以進行複雜的互動式編輯。要插入文字，只要從「插入」命令中挑一個，就可以開始輸入文字。

基本的命令可能有一個或兩個字元，例如：

`i`

插入

`cw`

更改文字

用字母作為命令，可以大幅增加速度。不需要死記一大堆的功能鍵，或是為了按出組合鍵而相當不自然地伸展手指。永遠不必將手從鍵盤上移開，也不必弄亂多階層的選單！大部分命令都可利用相關字母而記憶，幾乎所有的命令都有類似的模式，並且互相關聯。

---

7 troff 用於雷射印表機與排字機（typesetter）。它的「孿生兄弟」是 nroff，用於列表機和終端機。依照 Unix 慣例，我們把兩者並稱為 troff。現在，任何使用 troff 的人，都在使用 GNU 版的 groff（*http://www.gnu.org/software/groff*）。

碁峰 www.gotop.com.tw

一般來說，vi 與 Vim 的命令：

- 有字母大小寫的區塊（大寫與小寫表示不同的意義，I 與 i 功用不同）。
- 在輸入時不會顯示在螢幕上。
- 不需要在命令後加上 ENTER 鍵。

同時也有另一組命令顯示在螢幕的底端，而這些命令前有特殊的符號。斜線（/）與問號（?）會開啟搜尋命令，將於第三章「快速移動位置」中討論。冒號（:）會開啟所有的 ex 命令。ex 命令是 ex 行編輯器使用的命令。在任何版本，都可以使用 vi 的 ex 行編輯器，因為它是底層的編輯器，而 vi 只是它的「檢視」模式而已。ex 命令與觀念，將於第五章「ex 編輯器簡介」中討論，但是本章會介紹關閉檔案而不儲存的 ex 命令。

簡史

在深入瞭解 vi 的裡裡外外前，知道一些來龍去脈有助於理解你的環境上 vi 的世界觀。特別有助於想通許多讓人不解的 vi 錯誤訊息，也將欣賞 Vim 如何演進到超越原始的 vi。

vi 可回溯到電腦使用者還在終端機上操作，必須透過串列線路（serial line）與中央迷你電腦（minicomputer）連線的時代。連上中央電腦的終端機可能有好幾百台，而且分散在世界各地。每台終端機都能做同樣的動作（如清除螢幕、移動游標），但動作所需的命令各不相同。

除此之外，Unix 系統能讓使用者選擇用於倒退（backspace）、產生中斷訊號，及其他適合用於串列終端機（serial terminal）的指令，例如暫緩與繼續輸出。這些功能都（現在也仍然）使用 stty 命令管理。

最初的 Berkeley Unix 版本的 vi，將終端控制資訊從原始碼（很難更改）中抽出來，放入由 termcap 函式庫所管理的終端功能（**term**inal **cap**abilities）文字檔資料庫（比較容易更改）。

1980 年代初期，System V 引入一個二進制的終端機資訊（**term**inal **info**rmation）資料庫，與 terminfo 函式庫。以上兩個函式庫在功能上大致相同。為了告訴 vi 採用哪個函式庫，你必須設定環境變數 TERM；這個變數通常在 shell 起始檔中設定，例如 *.profile* 或 *.login*。

而 termcap 函式庫已不再使用。GNU/Linux 和 BSD 系統中使用 ncurses 函式庫，它提供兼容於 System V terminfo 函式庫中相似的資料與能力。

碁峰 www.gotop.com.tw

現在，大家都在圖形環境中使用終端仿真器（如 Gnome Terminal）。系統幾乎也都為我們設定了 `TERM`。

當然，你也能從個人電腦的非 GUI 控制台使用 Vim。在單一使用者模式下修復系統時，非常好用。不過，現在已經沒有太多人願意把這種方式當成日常工作的一環了。

在日常使用時，你很可能想要 GUI 版的 `vi`，例如 `gvim`。在 Microsoft Windows 或 Mac OS X 系統上，GUI 版的編輯器大概都是預設編輯器。然而，在虛擬終端上執行 `vi`（或相同年代的其他全螢幕編輯器），仍然使用 `TERM` 和 `terminfo`，並且需注意 `stty` 設定。在虛擬終端上使用它們，就跟學習 `vi` 和 Vim 的方式一樣簡單。

還有一項關於 `vi` 重要的事實需要瞭解，與現在相比，當時處於開發階段，在 Unix 系統中被視為較不穩定的。過去的 `vi` 使用者必須隨時應付系統不定時的當機（crash），所以 `vi` 支援回復正在編輯中的檔案[8]。所以，當各位學習 `vi` 和 Vim，看到各種關於潛在問題的說明時，請記得這些過往的發展。

開啟與關閉檔案

你可以使用 `vi` 編輯任何文字檔。編輯器將編輯的檔案複製到**緩衝區**（*buffer*，記憶體中另外設置的暫存區域）、顯示緩衝區（雖然一次只能看到一個螢幕大小的部分），並且讓你增加、刪除與更改文字。儲存編輯的結果時，則把緩衝區寫回永久的檔案中，替換同名的舊檔案。有一點要記住，你永遠是在緩衝區的檔案**副本**上作業，除非儲存緩衝區，否則編輯結果不會影響原始的檔案。儲存編輯的結果也稱為「寫入緩衝區」，或是更常見的「寫入檔案」。

從命令列開啟檔案

`vim` 是啟動 Vim 編輯器或編輯新舊檔案所用的 Unix 命令。`vim` 命令的語法是：

```
$ vim [filename]
```

或

```
$ vi [filename]
```

8 慶幸的是，這種事情不太常見，儘管系統仍然可能由於外部環境（例如停電）而崩潰。如果系統有不間斷電源，或者筆記型電腦上的健康電池，便無須擔憂。

碁峰 www.gotop.com.tw

在現代系統中，vi 通常是一個 Vim 連結。上述命令列出現了中括號，表示括號中的 filename 是選用項目，可有可無；中括號本身不用輸入。$ 是 Unix 的提示符號（shell prompt）。

如果省略 filename，vi 會開啟一個未命名的緩衝區。當你想將緩衝區裡的內容寫入檔案時，可在此時命名。不過我們還是保持良好習慣，先在命令列上給予檔案名稱。

檔名在目錄中必須是唯一的。（某些作業系統稱呼目錄為**資料夾**，兩者是一樣意思）

在 Unix 系統中，檔名可以包括除了斜線（/）與 ASCII 的 NUL 以外的任何八位元字元；斜線保留給路徑中檔案與目錄的分隔之用，而 ASCII NUL 則全部的位元都是 0。你甚至可以在檔名中包含空白字元，只要在前面加上反斜線（\）即可。（MS-Windows 系統中，不允許反斜線（\）和冒號（:）存在檔名之中。）實際上，檔名通常包含任意的大寫與小寫字母組合，再加上點（.）與底線（_）字元等。請記住，Unix 會區分大小寫：小寫字母與大寫字母視為不同字元。還要記得按下 ENTER 鍵，告訴 Unix 你已經結束命令了。

於目錄中開啟新檔時，應該在 vi 命令中加上新的檔名。例如，要在現行目錄中開啟一個名為 *practice* 的新檔時，你應該輸入：

```
$ vi practice
```

因為這是個新檔，緩衝區會是空的，螢幕的顯示將如下所示：

```
~
~
~
"practice" [New file]
```

最左邊的波浪符號（~）表示檔案中沒有文字，連空白行都沒有。底下的提示列（也稱為狀態列）顯示了檔案的名稱與狀態。

你也可以編輯任何已存在目錄中的檔案，只要指定檔名即可。假設有一個 Unix 檔案位於 /home/john/letter。如果你已經位在 /home/john 目錄中，可以使用相對路徑。例如：

```
$ vi letter
```

會將檔案 letter 的副本帶入畫面中。

如果是在另一個目錄中，則提供完整路徑名稱來編輯：

```
$ vi /home/john/letter
```


碁峰 www.gotop.com.tw

從 GUI 開啟檔案

儘管我們（強烈）建議要熟悉命令列，但可以直接從 GUI 環境對文件執行 Vim。在文件上點選滑鼠右鍵，然後從彈出的選單中選擇「開啟檔案」。如果 Vim 安裝正確，它將是打開文件可用的選項之一。

通常，也可以直接從選單系統中啟動 Vim，在這種情況下，需要使用 ex 命令 `:e` *`filename`* 告訴它要編輯哪個文件。

我們不能具體指出哪一種方式比較好，因為當今有很多不同 GUI 的使用環境。

開啟檔案可能發生的問題

- 見到下列任何一種訊息：

  ```
  Visual needs addressable cursor or upline capability
  terminal: Unknown terminal type
  Block device required
  Not a typewriter
  ```

  表示終端機型式沒有設定好，也可能是 `terminfo` 中有錯誤。輸入 `:q` 離開。通常將 `$TERM` 環境變數設置為 `vt100` 就足夠執行。如需更進一步幫助，可以使用網路搜索引擎或流行的技術問題論壇，例如：Stack Overflow（*https://stackoverflow.com*）。

- 當你認為檔案已存在時，卻出現 `[new file]` 訊息。

  檢查檔案名稱的大小寫是否正確（Unix 會區分檔名的大小寫）。如果正確，很可能位於錯誤的目錄。輸入 `:q` 離開，檢查是否位於正確的目錄中（在 shell 提示符號下輸入 `pwd`）。如果位於正確的目錄中，則檢查目錄中的檔案列表（使用 `ls`），以確定此存在的檔名是否和你輸入的檔名有一點點不同。

- 啟動 `vi`，卻得到 `:` 提示符號（表示你在 ex 行編輯模式下）。

  你可能在 `vi` 重畫螢幕前將其中斷（通常是 CTRL-C）。請在 ex 提示符號（`:`）下輸入 `vi`。

- 出現以下訊息之一：

  ```
  [Read only]
  File is read only
  Permission denied
  ```


碁峰 www.gotop.com.tw

「Read only」表示你只能查看檔案，無法儲存任何更動。你可能以**唯讀**模式（使用 `view` 或 `vi -R`）啟動了 `vi`；或是你對檔案沒有寫入的權限。參考第 8 頁的「從命令列開啟檔案」章節。

- 出現以下訊息之一：

```
Bad file number
Block special file
Character special file
Directory
Executable
Non-ascii file
file non-ASCII
```

表示你要編輯的檔案，不是一般的文字檔。輸入 `:q!` 離開，再檢查你要編輯的檔案，可以使用 `file` 命令。

- 當你遇上前述問題，而輸入 `:q` 後，卻出現如下訊息：

```
E37: No write since last change (add ! to override)
```

表示你更改了檔案而不自知。輸入 `:q!` 離開。你所做的改變將不會儲存到檔案中。

作業模式

曾稍早提過，現行「模式（mode）」的概念對 `vi` 的運作而言是最基礎的。模式有兩種，「命令模式」（*command mode*）與「插入模式」（*insert mode*）。（ex 命令模式可以被認為是第三種模式，但現在我們將它忽略。）一開始是命令模式，此時所有按鍵都代表命令[9]。在插入模式中，你輸入的東西都成為檔案的內容。

有時，你可能意外地進入插入模式，或是反過來，意外地離開插入模式。無論何種情況，輸入內容可能影響檔案，但又不是你想要的結果。

按下 ESC 鍵，迫使編輯器進入命令模式。如果你已經處於命令模式，編輯器會在你按下 ESC 鍵時發出「嗶」聲。（因此命令模式有時被稱為「嗶嗶模式」）。

一旦安全地進入了命令模式，即可動手修復意料之外的改變，並繼續編輯文字。（參考第 32 頁的「刪除可能發生的問題」與第 36 頁的「還原」章節）。

9 注意，`vi` 與 Vim 沒有針對每個可能的按鍵配置命令。因此，在命令模式下，編輯器希望接收代表命令的鍵，而不是將輸入的鍵寫入到檔案中。稍後會在第 126 頁的「使用 map 命令」章節中，充分運用未使用的鍵。

碁峰 www.gotop.com.tw

儲存與結束檔案

如果在終端機視窗下執行，你可以隨時結束正在工作的檔案，儲存編輯結果，並回到命令提示符號下。用於結束並儲存編輯結果的命令是 ZZ。請注意 ZZ 字母均為大寫。

假設建立了一個名為 *practice* 的檔案，用於練習 vi，並輸入了六行文字。想儲存這個檔案時，首先按下 ESC 鍵，檢查是否處於命令模式，而後輸入 ZZ。

按鍵順序	結果
ZZ	"practice" [New] 6L, 104C written
	輸入 ZZ（寫入並存檔的命令）後，檔案會儲存成一般的磁碟檔案。
$ ls	ch01.asciidoc ch02.asciidoc practice
	列出目錄中的檔案，顯示新建立的 *practice* 檔案。

此外，也可以用 ex 命令儲存編輯結果。輸入 :w 以儲存（寫入）檔案，但不離開 vi；若尚無編輯動作，可輸入 :q 退出；輸入 :wq，則是儲存編輯結果並結束（:wq 與 ZZ 相同作用）。我們會在第五章中完整解釋如何使用命令；現在，只需要記住一些寫入與儲存的命令即可。

結束而不儲存編輯結果

在初學 Vim 時，如果很喜歡大膽地作各種嘗試，有兩個 ex 命令可以輕鬆地回到原來的樣子。當想要消除這一次所有的編輯結果並且想載入回到原來的檔案時，採用命令：

```
:e! ENTER
```

將回到上一次儲存的檔案內容，你可以從頭來過。

假設想消除所有的編輯結果，直接離開編輯器，採用命令：

```
:q! ENTER
```

將迅速離開正在編輯的檔案，並回到命令提示符號下。使用這兩個命令後，自上一次存檔以來在緩衝區中所做的所有編輯，都將清除。編輯器通常不會放棄編輯的結果。然而在 :e 與 :q 命令後的驚嘆號，使得編輯器覆寫這個命令，即使緩衝區有所改變，仍然會執行這個命令。

之後，我們不會在 ex 模式命令中顯示 ENTER 鍵，但卻必須使用它來讓編輯器執行。

碁峰 www.gotop.com.tw

儲存檔案可能發生的問題

- 嘗試寫入檔案，卻得到以下的訊息：

```
File exista
File file exists - use w!
[Existing file]
File is read only
```

  輸入 `:w! file` 以覆蓋現存的檔案；或是輸入 `:w newfile`，把編輯的結果寫入新的檔案。

- 寫入檔案，卻沒有寫入的權限，並得到「*Permission denied.*」的訊息。

  使用 `:w newfile` 將緩衝區寫入一個新檔。如果擁有目錄的寫入權限，則可使用 `mv`，用新的檔案蓋掉原來的檔案。如果沒有目錄的寫入權限，就輸入 `:w pathname/file`，把緩衝區寫入某個擁有寫入權限的目錄（如：使用者的家目錄，或是 */tmp*）。注意不要覆蓋該目錄中的任何現有檔案。

- 嘗試寫入檔案，卻得到檔案系統已滿的訊息。

  現今，一個 500 GB 的硬碟都被認為很小，這樣的錯誤通常很少見。如果確實發生了這樣的事情，提供幾個步驟參考一下。首先，嘗試將檔案寫入不同檔案系統（如：*/tmp*）上的某個安全位置，以便保存資料。然後使用 ex 命令 `:pre`（`:preserve` 的縮寫）強制系統保存緩衝區。如果這不起作用，請搜尋一些可刪除的檔案，如下所示：

  — 打開圖形檔案管理器（如：GNU/Linux 上的 Nautilus），試著尋找不需要並且可以刪除的舊檔案。

  — 用 CTRL-Z 暫停 `vi` 並返回到 shell 提示符號。然後，可以使用各種 Unix 命令來尋找適合刪除的大檔案：

    — `df` 表示指定的檔案系統或整個系統上有多少可用磁碟空間。

    — `du` 表示指定的檔案和目錄使用了多少磁區塊。`du -s * | sort -nr`，這是一種取得檔案和目錄列表的簡單方法，並且按照使用的空間遞減排列。

  刪除文件後，再使用 `fg` 將 `vi` 放回前景運作；就可以正常儲存工作。

當這樣做時，除了使用 CTRL-Z 和作業控制之外，還可以鍵入 `:sh` 啟動一個新的 shell 完成工作。輸入 CTRL-D 或 `exit` 終止 shell 並返回 `vi`。（這個也是用於 `gvim`）

碁峰 www.gotop.com.tw

還可以使用 `:!du -s *` 之類命令，從 vi 中執行 shell 命令，並且在命令完成後返回編輯。

練習題

學習 vi 和 Vim 唯一的方法就是練習。目前已經瞭解如何建立新檔，以及回到命令提示符號。試著建立一個名為 *practice* 的檔案，插入一些文字，接著儲存並結束此檔。

在現行目錄下開啟一個名為 *practice* 的檔案	`$ vi practice`
切換插入模式	`i`
插入文字	隨便輸入一些文字
回到命令模式	ESC
結束 vi，儲存編輯結果	`ZZ`

碁峰 www.gotop.com.tw