前言

追求幸福

Perl 是一種能夠完成任務的語言。

當然，如果你的工作就是寫程式，那麼從理論上來講，你可以使用任何「完整」的程式語言來完成任務。但是我們可由經驗得知，各程式語言之間最大的區別不在於它**能做**什麼，而在於用它做事情是否**簡單**。從一個極端來說，使用那些所謂的第四代語言來做某些事情相當容易，但幾乎不可能用它們來做其他事情。從另外一個極端來看，用那些所謂的工業級語言做任何事情，幾乎都一樣困難。

Perl 則不同。從一開始，Perl 就被設計來輕鬆完成簡單的工作，同時又不失去處理難題的能力。

那什麼是「簡單的工作」呢？當然就是那些你每天都要做的工作。你需要一種語言讓你得以很容易地操作數字、文件、檔案、目錄、電腦和網路，尤其是操作其他程式。這種語言應該讓你輕易就能執行外部的程式並擷取這些程式的輸出，以取得你感興趣的東西。而且它還應該能夠很容易地把這些你感興趣的東西，交給其他程式做特殊的處理。當然，這種語言也應該能夠輕易地在任何現代的作業系統上進行移植、編譯和執行的工作。

Perl 可以做到上述所有要求，而且遠不止於此。

Perl 最初是設計來在 Unix 中當成膠水來用的一種語言，但是它早已被移植到大多數其他作業系統上。因為 Perl 幾乎可以在任何地方執行，所以 Perl 可以說是當今最具移植性的程式設計環境。要想編寫可移植的 C/C++ 程式，你得在程式裡加上一大堆奇怪的 `#ifdef` 標籤來區分不同的系統。要想編寫可移植的 Java 程式，你必須理解每種新的 Java 實作方法的風格。要想編寫可移植的 shell，你可能要記住每條命令在每種作業系統上的語法，運氣好的時候你可能可以找到一些共通的因子。而要想編寫可移植的 Visual Basic 程式，那麼你只需要對「可移植」（portable）這個詞下個更有彈性的定義即可 :-)

很高興的是，Perl 避免了所有這些問題，同時保留了這些語言中的許多優點，而且還有自己的一些特性。Perl 的特性來自許多方面：它特有的工具組、Perl 社群的創造性，以及開放源碼運動的大環境。不過，這些特性大多都是混合的東西；Perl 的身世複雜，它總是認為如此的多樣性是個優點，而不是缺點。Perl 是一種「讓我來撫慰勞累、困乏的你吧！」的語言。如果你覺得自己陷入了一團亂麻之中，非常渴望自由，那麼請使用 Perl。

Perl 跨越了各種文化。Perl 的爆炸性增長大部分是源自於那些前 Unix 系統程式設計師的渴望，他們希望從自己的「老地方」帶來儘可能多的東西。對於他們而言，Perl 是 Unix 文化的結晶，是「此路不通」沙漠中的綠洲。另一方面，Perl 還可以在另一個方向運作：在 Windows 上工作的 Web 設計者通常會非常開心地發現，他們的 Perl 程式可以不加修改地在 Unix 伺服器上執行。

儘管 Perl 在系統程式設計師和 Web 設計師中非常流行，但這只是因為，是他們最早發現 Perl 的，Perl 還可以用於更廣泛的用途。Perl 最早是一種文字處理語言，但它已經發展成為一種非常複雜、通用的程式語言，有著完整的開發環境，包括除錯器、分析工具、交互參照、編譯器、程式庫、具有語法提示功能的編輯器，以及其他「真正」的程式設計語言具有的所有陷阱，如果你需要的話。當然這些都是讓我們可能用來處理難題的東西，而且很多其他語言也可以做到這一點。Perl 之所以成為 Perl，是因為它從來不會因為要使簡單的事情更加簡單，而喪失其他方面的特性。

因為 Perl 既強大又好用，所以它被廣泛地用於各個領域，從航太工程到分子生物學、從數學到語言學、從圖形到檔案的處理，從資料庫操作到網路管理。很多人用 Perl 快速處理那些很難分析或需要轉換的大量資料，不管是處理 DNA 序列、網頁，還是其他意想不到的事情。實際上，在 Perl 社群有一個笑話，下次的股市大崩盤很有可能是因為某個傢伙寫的 Perl 命令稿裡頭有瑕疵所造成的。（樂觀的想，任何還在失業的股票分析師，仍然有足以糊口的技能。）

Perl 的成功有許多原因。早在開放原始碼運動（open source movement）成名之前，Perl 已經是一個成功的開放原始碼計劃了。Perl 是自由的，並將永遠自由下去。你可以在任何合適的場合使用 Perl，只需要遵守一個非常自由的授權規則就可以了。如果你正在從事商業活動並且想要使用 Perl，只管用就是了。你可以把 Perl 內嵌到你所寫的商業軟體中而不需要支付任何費用，也沒有任何限制。如果你碰上一個 Perl 社群解決不了的問題，那麼你有最後一招可用：回歸原始碼本身。Perl 社群不會在「升級」的偽裝下將商業秘密租給你。而且 Perl 社群永遠也不會「停業」而讓你孤立無援。

自由軟體的身份對 Perl 的發展確實有幫助。但這並不足以解釋 Perl 何以如此繁榮的現象，因為許多自由的套裝軟體都沒能興盛起來。Perl 不僅自由，還很好玩。人們覺得自己在 Perl 裡可以有創造力，因為他們有表達的自由：他們可以選擇是為計算機速度還是為程式設計師的速度做最佳化、是冗長還是簡潔、是選擇可讀性還是可維護性，或者選擇重複使用性、移植性、易學性和易教性等等。如果你正在參加 Perl 晦澀程式碼大賽（Obfuscated Perl Contest），你甚至還可以用最無法辨識的方法寫出 Per 程式來。

Perl 可以給你所有這些自由，因為它是一門有著多重性格的語言，集簡單與豐富的特質於一身。Perl 從其他地方拿來好的構想，然後把它們裝到易用的架構裡面。對於只是喜歡它的人來說，Perl 是實用的**萃取和報表語言**（Practical Extraction and Report Language）。對那些熱愛它的人而言，它是**有病的雜七雜八垃圾製造者**（Pathologically Eclectic Rubbish Lister）。在極簡主義者眼裡，Perl 像是毫無意義的重複練習（pointless exercise in redundancy）。這個世界總是需要一些化約主義者的（如物理學家）。化約主義者總是想把事物分裂開來，而我們則總是企圖把它們合併在一起。

Perl 之所以如此簡單，有很多原因。你用不著知道什麼特殊的指令就可以編譯 Perl 程式：只要把它當成批次檔或者 shell 命令稿執行就可以了。Perl 的資料型別和結構很容易使用和理解。Perl 對你的資料沒有任何限制，你的字串和陣列要多長就可以有多長（只要你有足夠的記憶體），而且它們都會自動增長。Perl 不會強迫你學習新的語法和語義，Perl 的語法大量借用自許多你已經熟悉的其他語言（比如 C、*awk*、BASIC 和 Python、英文、希臘語等）。實際上，任何程式設計師都可以從編寫良好的 Perl 程式碼中讀懂它的含義。

最重要的是，在學完 Perl 的一切之前，你就可以用它寫出有用的程式。你可以先從學寫很小的 Perl 程式開始，像小孩子牙牙學語那樣寫 Perl 程式，我們保證不會嘲笑你。或者更準確地說，我們絕不會嘲笑小孩做事情的創造性。Perl 的許多觀點都是從自然語言借來的，其中一條最好的觀點就是，只要你能把自己的意思表達清楚，那麼你就可以使用這些語言的一個子集。Perl 文化可以接受任何熟練程度的成員。我們不會在你背後放個語言警察。只要能夠在老闆炒你魷魚之前用你的 Perl 命令稿把工作完成，那麼它就是「正確」的。

儘管 Perl 很簡單，但它仍然是一個豐富的語言，如果你想要發揮這些特質，你還是得學一些東西。這也是為了解決難題所要付出的代價。縱然你仍需要一些時間來了解 Perl 到底能做多少事，但當你需要這些功能時，你會非常開心地發現，你已經可以用 Perl 做這些事情了。

一開始，Perl 只是一個將資料簡化的語言，它被設計來瀏覽檔案、大量擷取文件、產生動態資料，並輕鬆地印出這些資料的格式化報表。但由於 Perl 所繼承的特質，它也是個豐富的語言。有一天，Perl 開始大放異彩，成為可以操作檔案系統、行程管理、資料庫管理、進行主從架構程式設計和安全程式設計、Web 資訊管理，甚至可以進行物件導向和功能化程式設計的語言。而且這些功能並非只在 Perl 中適用，每種新功能都和其他東西交流得很好，別忘了，Perl 從一開始就是設計來做為膠水的語言。

但 Perl 不僅能黏合本身的特性。Perl 更被設計為可模組化的擴充。你不但可以用 Perl 快速設計、編寫、除錯和開發應用程式，還能在需求成長時輕易擴充這些程式的功能。你可以在其他語言中內嵌 Perl，也可以在 Perl 中內嵌其他語言。透過模組匯入機制，你可以將這些外部定義當成 Perl 的內建功能。物件導向的外部程式庫在 Perl 裡仍然保持物件導向的特性。

Perl 還可以在其他方面協助你。與一次執行一項命令的嚴格直譯式語言（比如命令檔和 shell 命令稿）不同的是，Perl 會先將整個程式快速編譯成一種中介格式。就和其他編譯器一樣，在此同時會進行一些最佳化，並且立刻回報從語法和語義錯誤到程式庫繫結災難等問題給你。一旦 Perl 的編譯器前端對你的程式尚感滿意，它就會把這些中介程式交給直譯器（或視需要交由其他能產生 C 或 bytecode 的模組後端）執行。這聽起來有點複雜，但 Perl 的編譯器和直譯器都很有效率，典型的「編譯－執行－修正」週期幾乎都只有幾秒的時間。

再加上 Perl 許多的容錯特性，這種敏捷應變的能力使得 Perl 很適合用來快速製作程式原型。當你的程式逐漸成熟，你可以自己將螺絲旋緊，使程式少一點直覺而多一點紀律。如果你想好好的做，Perl 也會很樂意幫你這個忙的。

Perl 還可以幫你增進程式的安全性。除了其他語言提供的典型安全介面，Perl 還會透過一種特殊的資料追蹤機制，自動判斷資料的來源是否安全，並預先防範可能發生的危險操作。最後，Perl 會讓你設定特別受到保護的區域，以便安全地執行來源不明的程式，避免意外的發生。

不過矛盾的是，Perl 對你的最大貢獻幾乎和 Perl 本身無關，反倒是和使用 Perl 的人有關。坦白講，Perl 使用者可以說是地球上最熱心的人了。如果說 Perl 運動中有那麼一點宗教情懷的話，那就是這種精神。Larry 希望 Perl 社群能像一個小天堂般，目前看來他的願望基本上是實現了。我們也請你為此盡上一份心力。

不管你是打算學習 Perl 以拯救世界，或只是覺得好奇，還是你的老闆強迫你學，本書都將由淺入深的告訴你一些東西。雖然我們沒有刻意教你寫程式，但深具慧眼的讀者還是能從中拾取一些程式藝術與一點點科學。我們鼓勵你發揚程式設計師的三項美德：**懶散**、**不耐煩**和**傲慢**。閱讀本書的同時，我們希望你能從中找到一些有趣的地方（以及其他爆笑之處）。如果這些還不足以讓你保持清醒，那麼就不停提醒自己，學習 Perl 至少可以提升你的履歷表的價值。所以請繼續讀下去吧。

第三版有哪些新內容

這個版本幾乎是全新的。

即使保留了前一版中相當不錯的片段（我們得承認有不少），我們也已經依據幾個目標做了大幅的修訂和整頓。首先，我們希望本書對那些沒有計算機科學背景的人更易一窺堂奧。我們假設讀者開卷前知道的更少。同時，我們保有活潑的演示，以防那些已經有一定了解的人看著看著就睡著了。

其次，我們希望呈現 Perl 最近期的發展。關於這點，我們絲毫不以展現目前的工作情況為恥，即使它仍處在實驗階段。儘管 Perl 的核心已久經考驗，但開發擴充功能的進程仍頗為顛躓。老實說，我們認為線上文件比這兒寫的東西更可靠。Perl 是藍領階級的語言，所以我們直言不諱。

第三，我們希望你能更自在地漫遊書海，因此，我們將這個版本拆成更小、更連貫的章節，並且把它們重組成更蘊意涵的各個部分。下面是新版本的內容：

第一部分，概論

萬事起頭難。此一部分以非正式、適合窩在椅上閱讀的方式介紹了 Perl 的基本概念。這不是完整的教程，只是個未必能滿足所有人的快速入門。你可以參考《離線文件》一節以找尋可能更貼合你學習風格的書。

第二部分，深度剖析

本部分係由「對 Perl 各層次的抽象概念，深入而毫不保留的討論」所組成，範圍從資料型別、變數、正規表示式，到副常式、模組和物件。你會對 Perl 的運作原理有更深刻的了解，並在過程中習得一些優良的軟體設計訣竅。（如果你從來沒有用過具樣式比對功能的語言，這是你大快朵頤的良機！）

第三部分，技術的 Perl

光用 Perl 就能夠做一堆事，但這部份會把你帶到更高的境界。在此你將學會如何帶領 Perl 飛渡陰山，範圍從處理萬國碼（Unicode）、內部的行程間通訊（IPC）以及多執行緒，到編譯、調用、除錯，剖析 Perl，以及依你的喜好使用 C 或 C++ 或任何既有的 API 寫出外部擴充。Perl 將會非常樂意與你電腦裡（甚至網路上其他電腦）的所有介面溝通。

第四部分，文化的 Perl

每個人都明白每種文化必有其語言，但 Perl 社群始終深信每種語言必定會有自己的文化。此部分中，我們將設計 Perl 程式視為一種深植於現實世界的人類行為。我們會談到如何改善你與好人和壞人相處的方式。我們還會花點時間討論如何做個好人，還有怎麼讓你的程式幫助更多人。

第五部分，參考資料

我們把所有你想查的資料依字母排序放進本書，內容包括特殊變數和函式，以及標準模組和編譯命令（pragma）。詞彙表對那些不熟悉電腦術語的人特別有用。比如說，如果你不知道「pragma」的含義，你現在就可以把它找出來！（當然啦，如果你不知道「含義」的含義，我們可就沒輒了。）【編註：由於本書英文版發行時，Perl 的最新穩定版為 5.6，編譯本書中文版時，Perl 的最新穩定版為 5.8.5，其中有很多資料可能已經過時了。由於這些資料大都可以線上查詢，況且本書英文版厚達 1067 頁，做成中文版之後勢必更厚，在眾多考量下，本書中文版不得不刪除這個部分（不過會保留詞彙表），請讀者見諒。當然本書中文版的價格也會因此而降低不少。】

標準發行套件

如今大多數的作業系統供應商把 Perl 當成作業系統的標準元件。目前已經有 AIX、BeOS、BSDI、Debian、DG/UX、DYNIX/ptx、FressBSD、IRIX、LynxOS、Mac OS X、OpenBSD、RedHat、SINIX、Slackware、Solaris、SuSE 和 Tru64 的標準版隨附 Perl 一起發行。有些公司在獨立的 CD 上提供 Perl 發行版本，或者是透過各自的客戶服務提供。協力廠商為多種不同的作業系統提供預先編譯好的 Perl 發行版本，如 ActiveState，這些平台包括微軟的作業系統。

即使你的系統供應商把 Perl 當成標準元件來提供，有時候你也需要自己編譯和安裝 Perl。這樣你就才可以使用最新的版本，以及選擇程式庫和文件的安裝位置。你還可以選擇是否需要一些延伸功能，比如多執行緒、大型檔案，或是許多可以透過 -D 命令列選項來使用的低階除錯選項。（當然使用者等級的 Perl 除錯器是一定可以使用的。）

下載 Perl 原始碼發行套件最簡單的方法，可能就是以網頁瀏覽器指向 Perl 的主網站 *www.perl.com*，在那裡你可以找到很明確的下載訊息，以及一些預先編譯好的二進制檔案（這是針對那些不提供 C 編譯器的平台）。

你還可以直接跑到 Perl 綜合典藏網（CPAN），方法是連接 *http://www.perl.com/CPAN* 或 *http://www.cpan.org*。如果你覺得太慢（因為它們**非常**熱門），那麼你應該找一個離你比較近的鏡射站台。下面這些 URL 只是世界上眾多 CPAN 鏡射站台中的幾個，現是鏡射站台的數目已經超過一百了：

http://www.funet.fi/pub/languages/perl/CPAN/

ftp://ftp.funet.fi/pub/languages/perl/CPAN/

ftp://ftp.perl.org/pub/perl/CPAN/

http://www.perl.com/CPAN/

http://www.cpan.org/

http://www.perl.org/CPAN/

http://www.cs.uu.nl/mirror/CPAN/

ftp://coda.nctu.edu.tw/UNIX/perl/CPAN/

http://CPAN.pacific.net.hk/

這個列表中的頭兩個（就是位於 *funet.fi* 的那兩個），指向主要的 CPAN 檔案庫。欲選擇你想使用的鏡射站台，可以點選 *CPAN sites* 連結（指 http 網站上）或是取得 *MIRRORED.BY* 文件（指 ftp 網站上）。這些站台有些可以透過 FTP 或 HTTP 協定來連接（在一些公司的防火牆背後，這兩種模式有些不同）。儘管 *http://www.perl.com/CPAN* 的多重檢驗程式會試著替你選擇，不過你可以在稍後改變選擇。

一旦下載原始碼發行套件並解壓縮後，你應該閱讀 *README* 和 *INSTALL* 文件，學習如何建立 Perl。同時你可能還要閱讀 INSTALL.*platform* 檔案，看看跟你的作業系統平台相關的訊息。

如果你的平台（*platform*）碰巧是某種 Unix，那麼用來取得、設定、建立和安裝 Perl 的命令可能是如下這幾個。首先，必須選擇取得原始碼發行套件的命令。你可以用 ftp：

```
% ftp ftp://ftp.funet.□/pub/languages/perl/CPAN/src/latest.tar.gz
```

（在這裡，可以選擇離你比較近的 CPAN 鏡射站台，當然，如果你住在芬蘭，那這個站台就是離你最近的 CPAN 鏡射站台。）如果不能用 *ftp*，那麼你也可以用瀏覽器或者命令列工具從網站下載：

```
% wget http://www.funet.□/pub/languages/perl/CPAN/src/latest.tar.gz
```

現在進行解壓縮、設定、建立和安裝：

```
% tar zxf latest.tar.gz          或者先 gunzip，然後 tar xf。
% cd perl-5.6.0                  或者是 5.* 之類的數字。
% sh Con□gure -des               假設使用預定的答案。
% make test && make install      安裝通常要求超級用戶。
```

這些工作要用到傳統的 C 開發環境，所以如果你沒有 C 編譯器，那你就不能編譯 Perl。請參閱 CPAN 的 *ports* 目錄，以獲取各個平台隨附 Perl 的最新訊息（以及隨附的 Perl 版本），還有你是否可以取得標準的原始碼發行套件，或者是否需要一個特定的移植版本等訊息。網站上會針對那些需要特別移植版本的系統，或是供應商通常不提供 C 編譯器（或是，反常地不提供 C 編譯器）的系統提供下載連結。

線上文件

Perl 標準的發行套件隨附了大量的線上說明文件。（離線說明文件請見下一節。）其他的說明文件將在你從 CPAN 安裝模組時顯示。

在本書提到「Perl 線上文件」時，我們指的是在你的電腦上所保存的 Perl 說明文件。**線上文件**（manpage）只是對包含說明文件之檔案的傳統稱法，你不一定需要使用 Unix 中的 *man* 程式來檢視它。你甚至可以把 Perl 說明文件安裝成 HTML 格式，尤其是在非 Unix 平台上。

Perl 的線上說明文件分成幾個獨立的部分，這樣你就可以很容易找到你要的東西，而不用翻閱幾百頁的文件。因為頂層（top-level）說明文件就是叫 *perl*，所以 Unix 命令 *man perl*（或 *perldoc perl*）會帶你到那裡去【註】。你所得到的說明頁面會引導你參考更準確的頁面。比如，*man perlre* 將會顯示 Perl 正規表示式的說明文件。*perldoc* 命令可供那些無法使用 *man* 命令的系統使用。在 Mac 上，你就需要使用 *Shuck* 程式。你的 Perl 版本還可能提供了 HTML 格式或是系統本機格式的 Perl 說明文件。請你的系統管理員檢查一下（除非你自己就是系統管理員）。

閱讀標準的線上文件

開始時（當然是 Perl 開始的時候，在 1987 年左右），*perl* 線上文件只是一個簡單的文件，如果印出來的話大約是 24 頁。例如正規表示式一節只有兩段長。（如果你知道 *egrep* 的話，那麼這些也就足夠了。）從某個角度來看，從那以後幾乎所有東西都改變了。包括標準說明文件、各種工具、與平台相關的移植訊息，以及許多標準模組等，現在我們有大約 1500 頁的說明分佈在許多獨立的線上文件中。（而且這些甚至還不包括你所安裝的任何 CPAN 模組，那裡還有不少。）

不過從另外一個角度來看，也沒有什麼改變：它們仍然是 Perl 線上文件。而且仍然是一個從頭開始的好地方。區別是一旦開始瀏覽，你就不能只停留在那裡。Perl 的說明文件不再是棉花工廠，而是有著幾百家商店的超級市場。當你走進大門時，你需要先知道自己在哪裡，哪個商店或者是在哪個櫃台，這樣你方能找到自己想買的東西。當然，一旦你熟悉了這個大市場，那麼你就能直接跑到相關商店了。

下面是幾個你會看到的商店招牌：

線上文件	內容
perl	有哪些 Perl 線上文件
perldata	資料型別
perlsyn	語法
perlop	算符和優先順序
perlre	正規表示式
perlvar	事先定義的變數

註　如果你用 *man perl* 仍然得到一個非常大的說明頁面，那麼你可能選到了 4.0 版的舊式線上說明文件。檢查你的 MANPATH 看看是不是有舊版的遺跡。（用 *perldoc perl* 來找，如何根據 *perl -V:man.dir* 的輸出來設定你的 MANPATH。）

線上文件	內容 （續）
perlsub	副常式
prelfunc	內建函式
perlmod	如何讓 perl 模組動起來
perlref	參考手冊
perldbj	物件
perlipc	行程間通訊
perlrun	如何執行 perl 指令，以及命令列選項參數
perldebug	除錯
perldiag	診斷訊息

這些只是摘錄的一小部分，但卻是重要的部分。如果你想看看算符，那麼使用 *perlop* 比較合適。如果你想找找事先定義的變數，那麼最好看看 *perlvar*。如果你看到了一條自己不明白的診斷訊息，可以看看 *perldiag*，等等。

有一些標準線上文件是答客問（FAQ）列表。它們被分割成了下面九個不同的說明頁面：

線上文件	內容
perlfaq1	關於 perl 的一般訊息
perlfaq2	獲取和學習 Perl
perlfaq3	程式設計工具
perlfaq4	資料操作
perlfaq5	文件和格式
perlfaq6	正規表示式
perlfaq7	一般的 perl 語言訊息
perlfaq8	與作業系統的互動
perlfaq9	網路

有些線上文件包含與平台相關的訊息：

線上文件	內容
perlamiga	Amiga 移植
perlcygwin	Cygwin 移植
perldos	MS-DOS 移植
perlhpux	HP-UX 移植

線上文件	內容 （續）
perlmachten	Power MachTen 移植
perlos2	OS/2 移植
perlos390	OS/390 移植
perlvms	DEC VMS 移植
perlwin32	MS-Windows 移植

（移植訊息還可以參考我們早先提到過的 CPAN *ports* 目錄。）

非 perl 的線上文件

在我們談到非 perl 的線上文件，如 *getitimer(2)*，時，指的是《Unix Programmer's Manual》之第二節編目的 *getitimer* 線上文件【註】。

像 getitimer 這種系統呼叫的線上文件在非 Unix 平台上可能無法找到，不過這應該不會造成問題，因為此時你也不能使用 Unix 系統呼叫。如果你真的需要 Unix 指令、系統呼叫或者程式庫的文件，那麼有許多單位把它們的線上文件到網頁上，比如在 Google 上找一下「xcrypt(3)+manual」就能找到許多。

雖然頂層 Perl 線上文件通常安裝在標準 *man* 目錄的第一節編目，但在本書中我們通常還是把 (1) 給省掉了。你應該可以很方便地找到它們，因為它們都叫「perl 某某」。

離線文件

如果你想多學些 Perl 的東西，下面是我們推薦的一些相關出版物：

- 《Perl 5 Pocket Reference》第三版，Johan Vromans 編著（歐萊禮公司出版）。這本小冊子中有很方便的 Perl 快速參考資料。
- 《Perl Cookbook》，Tom Christiansen 和 Nathan Torkington 編著（歐萊禮公司出版）。這是你手中目前這本書的姐妹作。

註　第二節編目，只包含對作業系統的直接呼叫（這些通常稱為「系統呼叫」，但是我們在本書接下來的說明中，只會叫它 *syscall* 以免跟系統函式混淆，系統函式跟 syscall 沒有關係。）然而，系統各有不同，其中某些系統的呼叫是用 syscall 實作而成的，而某些則是以 C 程式庫呼叫實作而成的，所以你有可能在第三節編目中找到 *getitimer* 的說明頁面。

- 《Elements of Programming with Perl》，Andrew L. Johnson 編著（Manning 公司出版）。本書教那些非程式員從零開始使用 Perl 程式設計。
- 《Learning Perl》，Randal Schwartz 等編著（歐萊禮公司出版）。這本書的內容是 Unix 系統管理員和程式員在 70% 的時間裡要用到的 30% 的基本 perl 內容。Erik Olson 針對微軟系統上的程式員撰寫了這本書的另一版本，書名是《Learning Perl for Win32 Systems》。
- 《Perl: The Programmer's Companion》，Nigel Chapman 編著（Wiley 公司出版）。這本好書主要是給專業計算機科學家和程式員看的，並沒有考慮平台。它迅速而完整地介紹了 Perl。
- 《Mastering Regular Expressions》，Jeffrey Friedl 編著（歐萊禮公司出版）。它對任何想學習和了解正規表示式內部工作原理的人來說都是無價之寶。
- 《Object Oriented Perl》，Damian Conway 編著（Manning 公司出版）。對新或老物件導向程式員而言，本書解釋了在 perl 中寫強大的物件系統之普通及高級技巧。
- 《Mastering Algorithms with Perl》，Jon Orwant、Jarkko Hietaniemi 和 John Macdonald 編著（歐萊禮公司出版）。 電腦演算法課上所有有用的訊息，不過沒有讓人痛苦的證明。此書包含圖形、文字、集合等領域的許多基本且實用的演算法。
- 《Writing Apache Modules with Perl and C》，Lincoln Stein 和 Doug MacEachern 編著（歐萊禮公司出版）。本書會指導你如何為 Apache 寫一個擴充其能力的模組，特別是使用快速的 mod_perl 做快速的 CGI 命令稿等。
- 《The Perl Journal》Jon Orwant 編。這本季刊是程式員寫給程式員看的，包括程式設計技巧、程式設計方法和新聞等等。

還有許多其他 Perl 書籍和出版物，毫無疑問的是我們肯定會漏掉一些很不錯的（請原諒，我們沒有提到那些差勁的）。

除了上面列出之與 Perl 相關的出版物，我們還推薦下面的書籍。它們和 Perl 沒有直接關係，但是用做參考、顧問以及從中找到靈感還是很不錯的。

- 《The Art of Computer Programming》的第一卷《Fundamental Algorithms》、第二卷《Seminumerical Algorithms》和第三卷《Sorting and Searching 》，Donald Knuth 編著（Addison-Wesley 公司出版）。
- 《Introduction to Algorithms》Cormen，Leiserson 和 Rivest 編著（MIT 出版社和 McGraw-Hill 出版）。
- 《Algorithms in C: Fundamental Data Structures，Sorting，Searching》第三版，Robert Sedgewick 編著（Addison-Wesley 公司出版）。

- 《The Elements of Programming Style》，Kernighan 和 Plauger 編著（Prentice-Hall 公司出版）。
- 《The Unix Programming Environment》，Kernighan 和 Pike 編著（Prentice-Hall 公司出版）。
- 《POSIX Programmer's Guide》，Donald Lewine 編著（歐萊禮公司出版）。
- 《Advanced Programming in the UNIX Environment》，W. Richard Stevens 編著（Addison-Wesley 公出版）。
- 《TCP/IP Illustrated》第一卷至第三卷，W. Richard Stevens 編著（Addison-Wesley 公司出版）。
- 《The Lord of the Rings 》，J. R. R. Tolkien 編著（最近一次印刷：Houghton Mifflin，1999 年）。

其他資源

網際網路真是一個偉大的發明，我們大家仍然在發掘它更多的潛在用途。

Perl 的網站

Perl 主網站位於 *http://www.perl.com/*。其中包括 Perl 世界的新聞以及原始碼程式和移植版本、專欄文章、檔案、會議安排，等等。

還有 Perl 推廣組（Perl Mongers，簡稱 PM）位於 *http://www.perl.org* 的網站。這裡都是 Perl 的基礎理念，就如草根一樣，在世界上所有地方都生長得十分茂盛（南極除外）。各地 PM 會定期召開小型會議，會議期間，你可以跟與你居住在同一地區的其他 Perl 黑客交流有關 Perl 的經驗和知識。

Usenet 新聞群組

如果你不知道該到哪裡找 Perl 的訊息，Perl 新聞群組會告訴你。你的第一站應該是 *comp.lang.perl.moderated*，它是一個有評比、低流量的新聞群組，包括訊息公告和技術討論。因為有評比，所以這個群組相當易讀。

高流量的 *comp.lang.perl.misc* 群組討論關於 perl 的所有事情，從技術問題到 Perl 哲學、Perl 遊戲與 Perl 詩歌。和 Perl 本身一樣，*comp.lang.perl.misc* 群組即是有用的代名詞，沒有什麼問題會被認為是太愚蠢的【註】。

comp.lang.perl.tk 群組討論如何在 Perl 中使用流行的 TK 工具箱。*comp.lang.perl.modules* 群組討論 Perl 模組的開發和使用，是取得可重用程式碼最好的地方。當你閱讀到這裡時可能還有其他 *comp.lang.perl.whatever* 群組被成立，你可以找找看。

如果你不是用普通的「新聞群組閱讀器」存取 Usenet，而是用網頁瀏覽器，那麼你要在新聞群組的名字前面加上「news:」。（前提是你有一個新聞群組伺服器可用。）另外，你也可以使用 Google 的「網上論壇」功能，指定「perl」為所要搜尋的新聞群組。

還有另外一個新聞群組你可能也會想看看，至少如果你正在網站上寫 CGI 的話：comp.infosystems.www.authoring.cgi。雖然嚴格說來它不是 Perl 新聞群組，但在那裡討論的大多數程式都是用 Perl 撰寫的。與網頁相關的 Perl 問題到這個群組比較合適；除非你是在 Apache 裡用 mod_perl，這個時候你可能就需要造訪 *comp.infosystems.www.servers.uinx* 了。

瑕疵報告

在某些不尋常的狀況下，你不只會在自己的程式中找到瑕疵，也會在 Perl 中找到。這時，你應該把它化約成一個最小的測試單元，然後用 Perl 隨附的 *perlbug* 程式回報給我們。請參閱 *http://bugs.perl.org* 以取得更多的資訊。

排版風格

某些風格本身就可以形成一個大章節。像是「程式設計風格」將會在第 24 章中討論。基本上，我們的用語習慣在詞彙表就可以看到。

本書使用下列編排習慣：

斜體字

用於網址、線上文件、路徑名稱和程式名稱。

註　當然有些問題太愚蠢了以致於無法回答（尤其是那些線上說明文件跟答客問已經回答過的問題。當你只要花比打問題還少的時間就可自行找到答案時，為甚麼還要在新聞群組上找人幫忙呢？）

定寬字

用於程式碼範例，以及程式的輸出。

定寬粗體字

用於命令列選項，以及例子中你所鍵入的文字。

定寬斜體字

用於程式碼中必須替換成特定值的項目。

我們舉了許多例子，這些例子多半是大程式的一部分。有些例子是完整的程式，你應該看得出來，因為它們是以 `#!` 列開頭的。較長的程式範例幾乎都是以

```
#!/usr/bin/perl
```

開頭的。書中還會舉一些在命令列上鍵入文字的範例。我們會用「%」來表示常見的 shell 提示符：

```
% perl -e 'print "Hello, world.\n"'
Hello, world.
```

這個風格是標準 Unix 命令列的代表，單引號是最常被引用（most quoted）的形式。其他系統裡的引號和萬用字符（通配符）習慣可能有所不同。比如，當你需要在命令列中使用帶空格的參數或萬用字符時，MS-DOS 和 VMS 的許多命令列直譯器，所需要的是雙引號而不是單引號。

致謝

在此我們要公開向以下我們私下抱怨過的技術審稿人致謝：Todd Miller、Sharon Hopkins Rauenzahn、Rich Rauenzahn、Paul Marquess、Paul Grassie、Nathan Torkington、Johan Vromans、Jeff Haemer、Gurusamy Sarathy、Gloria Wall， Dan Sugalski 和 Abigail。

我們還要特別感謝 Tim O'Reilly（和他的公司）鼓勵我們寫這些讀者可能愛看的東西。

建議與評論

歐萊禮公司是世界性的電腦資訊出版公司。我們永遠樂意聽到讀者對出版品的意見，包括如何讓本書可以更好的建議、指正本書的錯誤、或是讀者建議本書往後改版時，應該再加進來的其它主題。以下是本公司的聯絡資料：

美商歐萊禮股份有限公司台灣分公司

電話：（02）2709-9669　　傳真：（02）2703-8802

網頁：*http://www.oreilly.com.tw*

電子郵件：

sales@oreilly.com.tw　　（業務部）

edit@oreilly.com.tw　　（編輯部）

bookquestion@oreilly.com.tw　　（書籍內容的問題）

與本書有關的線上資訊（包括勘誤、範例程式、相關連結）：

原文書

http://www.oreilly.com/catalog/pperl3

中文書

http://www.oreilly.com.tw/chinese/perl/pperl3.html