前言

本書將讓我們的 Linux 命令列技巧更上一層樓，使得工作可以更快、更聰明、更有效率的完成。

像大多數 Linux 使用者，都是在工作中學習過往的命令列技巧，或者透過閱讀相關書籍的介紹，亦可在家中安裝 Linux 並嘗試一下。作者寫這本書是為了幫助大家跨出下一步——在 Linux 命令列中由基礎至進階的技能。裡頭充滿了技術和觀念，希望能改變讀者與 Linux 互動的方式，並提高工作效率。將其視為作者第二本關於 Linux 使用的書籍，讓讀者跨越基礎知識的藩籬。

命令列是最單純的界面，但也是最具有挑戰性的。因為簡單，所以我們只會看到一個提示符號（prompt），等待執行可能輸入的任何命令[1]：

```
$
```

在沒有友善的圖示介面、按鈕或選單來引導操作的情況下，這變得相當具有挑戰性，因為除了提示符號以外的一切都成為使用者的責任。相反地，每個輸入的命令都具有創造性的動作。以下的基本命令是正確的，例如列出目前的檔案：

```
$ ls
```

以及更複雜的命令，例如：

```
$ paste <(echo {1..10}.jpg | sed 's/ /\n/g') \
        <(echo {0..9}.jpg  | sed 's/ /\n/g') \
  | sed 's/^/mv /' \
  | bash
```

1 本書以錢字符號（dollar sign），作為本書的 Linux 提示符號。但在不同環境下，讀者環境中的提示符號可能有所不同。

如果我們盯著前面的命令思考：「那些到底是什麼？」或者「我可能永遠不需要這麼複雜的命令！」那麼這本書正是為此目的而準備的[2]。

本書將可學習到什麼內容

本書針對以下三個必要技能，變得更快、更有效：

- 選擇或建構命令來解決手邊的任務問題
- 有效率地執行這些命令
- 輕鬆瀏覽 Linux 檔案系統

到最後，我們將理解執行命令時背景所發生的事件，這樣就可以更好地預測結果（而非演變成缺乏科學論證的街談巷語）。也將看到用於啟動命令的多種不同方法，並理解何時使用，來獲得最佳效果。還將學習常用的例子和技巧，提高工作效率，例如：

- 從建立簡單到複雜的命令，逐步解決實際問題，如：管理密碼或產生一萬個測試檔案。
- 透過聰明的組織使用者的家目錄，用以節省時間，如此就不必尋找檔案。
- 轉換文字檔案並如同查詢資料庫一樣檢視它們，實現達成任務目標。
- 從命令列控制 Linux 的點選功能，例如：使用剪貼簿進行複製、貼上，以及檢索和處理 Web 數據資料，而無須將手從鍵盤上移開。

最重要的是，我們無論執行哪些命令，都可以在日常 Linux 使用中獲得更多的成功例子，並在就業市場上更具有競爭力，才是一般學習的最好實踐方式。這是作者當初在學習 Linux 時所希望擁有的書籍。

哪些內容不在本書的範疇

本書不會優化 Linux 電腦，使其更高效地運作執行，但會讓我們與 Linux 在互動上更有效率。

2 我們將在第 8 章中，說明這個神祕命令的目的。

本書也不是全方位的命令列參考書籍——有數百個命令及其功能是作者所未提及到的。此外，內容是討論相關專業知識；並精心安排一系列教學順序，挑選常用的命令來說明其中的內容，以培養讀者的技能。倘若讀者需要指南風格的書籍，請參考作者之前的著作 *Linux Pocket Guide*（O'Reilly）。（*https://oreil.ly/46N1v*）

讀者的必備知識

本書**不是基礎教學**，而是假設讀者已有 Linux 經驗。適用於希望提升命令列技巧的使用者，如學生、系統管理員、軟體開發人員、網站可靠性工程師、測試工程師和一般 Linux 使用者；以及進階使用者也可能會找到一些有用的資料。如果反覆地實驗學習執行命令，會更加鞏固對概念的理解。

要從本書中獲得最大效益，希望讀者應該已經熟悉以下主題（如還不熟悉，請參考附錄 A ，快速複習）：

- 使用文字編輯器，建立並編輯文字檔，例如：`vim(vi)`、`emacs`、`nano` 或 `pico`。
- 檔案處理的基本操作，例如：`cp`（複製）、`mv`（移動或重新命名）、`rm`（刪除）和 `chmod`（修改檔案權限）。
- 檔案檢視命令，例如：cat（檢閱整個檔案）和 less（一次瀏覽一頁）
- 目錄命令，例如：`cd`（修改目錄）、`ls`（列出目錄中的檔案）、`mkdir`（建立目錄）、`rmdir`（刪除目錄）和 `pwd`（顯示當下目錄名稱）。
- shell 指令稿的基礎知識：將 Linux 命令儲存在一個檔案中，變更檔案屬性成為可執行（使用 `chmod 755` 或 `chmod +x`），之後再執行檔案。
- 使用 `man` 命令來查看 Linux 的內建說明文件：manpage 手冊（範例：`man cat` 顯示有關 `cat` 命令的文件）。
- 使用 `sudo` 命令成為超級使用者，如此才能完全存取 Linux 系統（例如：`sudo nano /etc/hosts` 編輯系統檔案 */etc/hosts*，這對於一般使用者而言，檔案都是被保護的）。

如果讀者還能夠操作常見的額外命令功能，例如檔案名稱的樣式比對（使用符號 `*` 和 `?`）、輸入 / 輸出重新導向（`<` 和 `>`）以及管線（`|`），那麼會是一個不錯的起點。

第一章

組合命令

當我們在 Windows、macOS 和其他大多數作業系統中工作時，可能會花時間執行 Web 瀏覽器、文字處理器、資料表格計算和遊戲等應用程式。一個典型的應用程式可能包含以下功能：程式設計師會為他們的使用者，提供所需要的一切。因此，大多數應用程式都是自給自足的。與其他應用程式互不依賴。我們可能會需要在應用程式之間複製和貼上，但在大多數情況下，它們是分開的。

這與 Linux 命令列環境有些許不同。Linux 沒有提供類似大型應用程式所具有的大量功能，而是提供數千個功能很少的小命令。例如，命令 `cat` 在螢幕上列印檔案，僅此而已。`ls` 列出目錄中的檔案，`mv` 對檔案重新命名，等等。每個命令都有一個簡單、明確的定義與目的。

如果我們需要做一些更複雜的事情怎麼辦？不用擔心。Linux 使用**組合命令**，讓各自功能的命令相互合作，讓實現目標將變得容易。這種工作方式產生了一種截然不同的運算思維。將問題從「我們應該啟動哪個應用程式？」變成「我們應該組合哪些命令？」來取得成果。

在本章中，我們將學習如何使用不同的組合方式，來執行命令，完成需要的操作。為簡單起見，作者將只介紹六個 Linux 命令及其最基本的操作方法，這樣讀者可以專注在更複雜、更有趣的部分——將它們組合起來——而無須花費大量的學習時間。這有點像學習只使用六種食材做菜，或者只用鎚子和鋸子學習木工。（我們將在第 5 章中加入更多命令到 Linux 工具箱中。）

我們將使用**管線**（*pipe*）來組合命令，這是一種將前一個命令的輸出，連接到另一個命令的輸入。在介紹每個命令（`wc`、`head`、`cut`、`grep`、`sort`、`uniq`）時，將會立即示範它們與管線的使用方法。

某些範例對於 Linux 的日常操作來說是相當常見的，而其中也有一些是用來作為示範重要功能的簡單範例。

輸入、輸出和管線

大多數 Linux 命令是由鍵盤讀取輸入、輸出至螢幕，或兩者同時皆有。Linux 替這種讀寫方式取了特殊的名字：

stdin（讀作「標準輸入」（*standard input* 或 *standard in*））

Linux 從鍵盤讀取輸入的資料。當我們在提示符號下輸入任何命令時，就是在標準輸入中提供相關資料。

stdout（讀作「標準輸出」（*standard output* 或 *standard out*））

Linux 將資料輸出到顯示裝置。當我們執行 `ls` 命令列印檔案名稱時，結果顯示在標準輸出上。

接下來是很酷的部分。我們可以將一個命令的標準輸出連接到另一個命令的標準輸入，讓第一個命令的輸出結果提供給第二個命令的輸入。先從熟悉 `ls -l` 命令開始，在一個大的目錄中列出檔案列表，例如 */bin*：

```
$ ls -l /bin
total 12104
-rwxr-xr-x 1 root root 1113504 Jun  6  2019 bash
-rwxr-xr-x 1 root root  170456 Sep 21  2019 bsd-csh
-rwxr-xr-x 1 root root   34888 Jul  4  2019 bunzip2
-rwxr-xr-x 1 root root 2062296 Sep 18  2020 busybox
-rwxr-xr-x 1 root root   34888 Jul  4  2019 bzcat
⋮
-rwxr-xr-x 1 root root    5047 Apr 27  2017 znew
```

這個目錄包含的檔案比上面所顯示的行數要來得多，因此輸出的大量資料會快速捲動到螢幕之外。遺憾的是 `ls` 不能一次列印一個螢幕，並且暫停直到我們按下某個鍵繼續列印。不過也因為這樣，可推測另一個 Linux 命令該具有的功能。`less` 命令一次只顯示一個螢幕的檔案列表：

```
$ less myfile
```

如此可以將這兩個命令連接起來，讓 `ls` 寫入 stdout，而 `less` 可以從 stdin 讀取。使用管線將 `ls` 的輸出傳送至 `less` 的輸入：

```
$ ls -l /bin | less
```

這個組合命令形成每次顯示一整個螢幕的目錄內容。命令之間的垂直線（|）是 Linux 的管線符號[1]。管線將第一個命令的標準輸出連接到下一個命令的標準輸入。任何包含管線的命令列都稱為**管線**（*pipeline*）。

命令通常不知道它們是管線的一部分。當 `ls` 認為自己正在寫入到顯示器時，實際上輸出已被重新導向到 `less`。而當 `less` 認為自己正在從鍵盤讀取資料時，實際上它卻在讀取 `ls` 的輸出。

什麼是命令？

命令（*command*）這個用字，在 Linux 中可能有三種不同的含義，如圖 1-1 所示：

一個程式

由一個單字命名，並且是可執行的程式，例如 `ls` 或 shell 內部所擁有的類似功能、又例如：`cd`（又稱為 *shell 內建功能*；*shell builtin*）[2]。

一個簡單的命令

程式名稱（或 shell 內建功能）之後緊接著一些可選擇性的參數，例如：`ls -l /bin`。

組合命令

將好幾個簡單的命令視為一個執行單元，例如：管線 `ls -l /bin | less`。

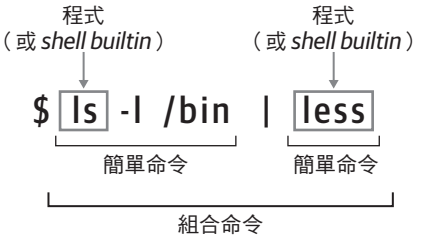

圖 1-1　程式、簡單命令、組合命令，統稱為「命令」

在本書中，作者將全面使用**命令**這個詞。通常透過相關的前後文，可以清楚瞭解其中的意思，但若非如此，作者會使用另一個更具體的詞彙。

1　在美式鍵盤上，垂直線符號與反斜線（\）符號在同一個鍵上，通常位於 Enter 和 Backspace 鍵之間或左邊 Shift 鍵和 Z 之間。

2　POSIX 標準將這種形式的命令稱為工具程式（*utility*）。

六個幫助我們入門的命令

管線是 Linux 專業知識的重要組成部分。接下來我們會深入討論如何使用一組 Linux 命令，來建構管線的技巧，這樣無論之後遇到其他命令，都可以將它們組合起來。

這六個命令分別是 `wc`、`head`、`cut`、`grep`、`sort`、`uniq`。其中有許多選項和操作方式，我們暫時先跳過這些細節，只專注於管線的使用上。要學習任何有關命令的更多資訊，請執行 `man` 命令來顯示完整文件。例如：

```
$ man wc
```

為了示範這六個命令，作者將使用一個名為 *animals.txt* 的檔案，其中列出一些 O'Reilly 圖書資訊，如範例 1-1 所示。

範例 1-1　在檔案 animals.txt 中的內容

```
python  Programming Python      2010    Lutz, Mark
snail   SSH, The Secure Shell   2005    Barrett, Daniel
alpaca  Intermediate Perl       2012    Schwartz, Randal
robin   MySQL High Availability 2014    Bell, Charles
horse   Linux in a Nutshell     2009    Siever, Ellen
donkey  Cisco IOS in a Nutshell 2005    Boney, James
oryx    Writing Word Macros     1999    Roman, Steven
```

每一欄包含關於 O'Reilly 書籍的四個資訊項目，由 tab 控制字元作為分隔，分別為：封面上的動物、書名、出版年份和第一作者的姓名。

命令 #1：wc

`wc` 命令列印檔案中的行數、單字數量和字元數量：

```
$ wc animals.txt
  7  51 325 animals.txt
```

`wc` 回報檔案 *animals.txt* 有 7 行、51 個單字、325 個字元。但我們實際數一下字元（包括空格和 tab）會發現只有 318 個字元，然而 `wc` 還包括每行結尾那不可見的換行符號。

此外，選項 `-l`、`-w`、`-c` 分別告知 `wc` 僅顯示行數、單字數量和字元數量：

```
$ wc -l animals.txt
7 animals.txt
$ wc -w animals.txt
51 animals.txt
```

```
$ wc -c animals.txt
325 animals.txt
```

一般而言，計算數量是非常有用的任務，因此 wc 的開發者設計與管線一起操作命令。如果省略檔案名稱，它會從 stdin 讀取，然後寫入 stdout。讓我們使用 ls 列出現在目錄下的內容，並將透過管線傳到 wc 來計算行數。整個管線最後的結果，將回答「現在目錄中有多少個可見檔案？」這個問題。

```
$ ls -1
animals.txt
myfile
myfile2
test.py
$ ls -1 | wc -l
4
```

選項 -1 告訴 ls，單一欄印出其結果，在這裡並非絕對必要的。要理解使用它的原因，請參考第 8 頁的「在重新導向時利用 ls 改變其行為」專欄。

wc 是我們在本章中看到的第一個命令，因此使用管線可以做的事情有些侷限。以下只是純粹為了好玩，將 wc 的輸出透過管線傳遞給自己，用以證明同一命令可以在管線中多次出現。這個組合命令 wc 輸出結果為字數 4：以及三個整數和一個檔案名稱：

```
$ wc animals.txt
  7  51 325 animals.txt
$ wc animals.txt | wc -w
4
```

為何要停止下來？將計算輸出的結果「4」，添加到第三個 wc，合併到管線中繼續計算：

```
$ wc animals.txt | wc -w | wc
      1       1       2
```

這樣的輸出結果，表示一行（數字 4 只有一行）、一個單字（數字 4 本身）和兩個字元。為什麼是兩個字元？因為「4」是以行表示，後面帶有一個不可見的換行符號作為結尾。

到這裡 wc 的管線例子應該足夠了。隨著後續的更多命令，管線也將變得更加常用。

在重新導向時利用 ls 改變其行為？

ls 幾乎與所有其他 Linux 命令不同，ls 知道 stdout 是螢幕，還是被重新導向（到管線或其他地方）。主要是因為它對使用者的友好特性。如果當 stdout 是螢幕時，ls 會在排列輸出上，盡量以多列的方式呈現以便於閱讀：

```
$ ls /bin
bash        dir         kmod      networkctl       red     tar
bsd-csh     dmesg       less      nisdomainname    rm      tempfile
⋮
```

但是，當 ls 的 stdout 被重新導向時，其輸出結果將會變成一欄的資料形式。我們設計以下的簡單範例，將 ls 命令的輸出，傳遞給 cat 來示範這一點，結果如下[3]：

```
$ ls /bin | cat
bash
bsd-csh
bunzip2
busybox
⋮
```

這種行為會導致結果看起來很奇怪：

```
$ ls
animals.txt   myfile   myfile2    test.py
$ ls | wc -l
4
```

第一個 ls 命令將所有檔案名稱列印在同一行之中，但第二個 ls 命令卻回報結果為四行。如果不瞭解 ls 的古怪行為，很可能因為這種差異而令人困惑。

ls 具有覆蓋原先預設動作的選項。使用 -1 選項，強制 ls 以單欄列印；或使用 -C 選項，強制以多欄列印。

3 根據設定，ls 也可能在列印輸出到螢幕時，使用其他格式化功能，例如顏色標示，但在重新導向時則不會。

命令 #2：head

head 命令會列印檔案的第一行。使用 head 的選項 -n 列印輸出 *animals.txt* 的前三行：

```
$ head -n3 animals.txt
python    Programming Python      2010    Lutz, Mark
snail     SSH, The Secure Shell   2005    Barrett, Daniel
alpaca    Intermediate Perl       2012    Schwartz, Randal
```

如果我們輸入需要的行數，大於檔案內容的行數，則 head 會列印整個檔案（如同 cat 一般）。如果省略 -n 選項，head 預設為 10 行（-n10）。

然而就其本身而言，倘若無須在意檔案的多餘內容時，head 可以很方便檢視前幾行的內容。這是一個快速且有效率的命令，即使用在非常大的檔案也亦是如此，因為無須讀取整個檔案。此外 head 寫入標準輸出，使其在管線中顯得更加有功用。以下是統計 *animals.txt* 前三行的單字數：

```
$ head -n3 animals.txt | wc -w
20
```

head 還可以從 stdin 讀取更多管線的內容，找出有趣的資訊。一個常見的用途是當我們不想看到所有內容的時候，用以減少前一個命令的輸出，例如一串很長的檔案列表。以下列出 */bin* 目錄中的前五個檔案名稱：

```
$ ls /bin | head -n5
bash
bsd-csh
bunzip2
busybox
bzcat
```

命令 #3：cut

cut 命令用來取得檔案中，某一欄或多欄的內容，並呈現出來。例如，只列印出在 *animals.txt* 中第二欄的所有書名：

```
$ cut -f2 animals.txt
Programming Python
SSH, The Secure Shell
Intermediate Perl
MySQL High Availability
Linux in a Nutshell
Cisco IOS in a Nutshell
Writing Word Macros
```

cut 提供兩種定義「欄位」的方法。第一種是當輸入的字串是由 tab 字元所分隔的，並根據選項（-f）來選取被切割後需要的欄位。這剛好是檔案 *animals.txt* 的格式。由於前面的 cut 命令選項為 -f2，表示列印每行的第二個欄位。

為了縮減輸出的資訊，將透過管線傳輸到 head 用來限制只列印前三行：

```
$ cut -f2 animals.txt | head -n3
Programming Python
SSH, The Secure Shell
Intermediate Perl
```

我們還可以透過用逗號，區隔欄位編號，來取得裁切後的多個欄位：

```
$ cut -f1,3 animals.txt | head -n3
python  2010
snail   2005
alpaca  2012
```

或依照數值範圍：

```
$ cut -f2-4 animals.txt | head -n3
Programming Python      2010    Lutz, Mark
SSH, The Secure Shell   2005    Barrett, Daniel
Intermediate Perl       2012    Schwartz, Randal
```

第二種方法是使用 -c 選項，表示由 cut 所依照字元位置所定義的「欄」。以下是列印檔案每一行的前三個字元，我們可以使用逗號（1,2,3）或範圍（1-3）來指定需要的部分：

```
$ cut -c1-3 animals.txt
pyt
sna
alp
rob
hor
don
ory
```

現在已經瞭解基本功能後，再進一步嘗試使用 cut 和管線來進行更有用的操作。假設 *animals.txt* 檔案有數千行，而我們只需要取得作者的姓氏。首先，需要區隔第四個欄位，作者姓名：

```
$ cut -f4 animals.txt
Lutz, Mark
Barrett, Daniel
Schwartz, Randal
⋮
```

然後透過管線，將結果再次裁切，這次使用的是選項 `-d`（分隔符號），將分隔符號修改為逗號而非 tab，用來區隔作者的姓氏：

```
$ cut -f4 animals.txt | cut -d, -f1
Lutz
Barrett
Schwartz
⋮
```

透過歷史記錄來節省修改命令的時間

讀者是否發現到重新輸入很多次命令？我們可以透過不斷按向上方向鍵來捲動瀏覽之前執行過的命令。（這個 shell 功能稱為*命令歷史記錄*（*command history*））當看到我們所需要的命令時，按 Enter 執行，或使用左、右方向鍵做游標定位或退格鍵（Backspace）對命令內容進行編輯及刪除。（此功能亦稱為*命令列編輯*（*command-line editing*））

作者將在第 3 章中，討論更強大有關歷史記錄和編輯功能的命令。

命令 #4：grep

`grep` 是一個非常強大的命令，但現在要暫時隱藏它大部分功能，專注在列印與符合指定字串的行。（更多詳細資訊將在第 5 章介紹。）例如，以下命令顯示 *animals.txt* 中包含字串 `Nutshell` 的行：

```
$ grep Nutshell animals.txt
horse   Linux in a Nutshell     2009    Siever, Ellen
donkey  Cisco IOS in a Nutshell 2005    Boney, James
```

我們還可以使用 `-v` 選項，列印出不符合指定字串的行。請注意其結果，顯示不存在包含「Nutshell」的行：

```
$ grep -v Nutshell animals.txt
python  Programming Python      2010    Lutz, Mark
snail   SSH, The Secure Shell   2005    Barrett, Daniel
alpaca  Intermediate Perl       2012    Schwartz, Randal
robin   MySQL High Availability 2014    Bell, Charles
oryx    Writing Word Macros     1999    Roman, Steven
```

通常 `grep` 對於蒐集出現在檔案中搜尋文字的行內容很有用處。以下命令將列印名稱以 *.txt* 結尾的檔案中，包含字串 `Perl` 的行：

```
$ grep Perl *.txt
animals.txt:alpaca      Intermediate Perl       2012    Schwartz, Randal
essay.txt:really love the Perl programming language, which is
essay.txt:languages such as Perl, Python, PHP, and Ruby
```

範例中 grep 找到三個比對相符的行，一個在 *animals.txt* 中，兩個在 *essay.txt* 中。grep 讀取 stdin 並寫入 stdout，這非常適合管線。假設我們想知道在龐大目錄 */usr/lib* 中有多少個子目錄。沒有一個 Linux 命令可以提供這樣的答案，因此需要建立一個管線來處理。從 ls -l 命令開始：

```
$ ls -l /usr/lib
drwxrwxr-x  12 root root    4096 Mar  1  2020 4kstogram
drwxr-xr-x   3 root root    4096 Nov 30  2020 GraphicsMagick-1.4
drwxr-xr-x   4 root root    4096 Mar 19  2020 NetworkManager
-rw-r--r--   1 root root   35568 Dec  1  2017 attica_kde.so
-rwxr-xr-x   1 root root    684  May  5  2018 cnf-update-db
⋮
```

請注意 ls -l 在每一行的開頭用以 d 標記為目錄。使用 cut 分隔出第一欄，判斷它是不是 d：

```
$ ls -l /usr/lib | cut -c1
d
d
d
-
-
⋮
```

然後使用 grep 保留下含有 d 的行：

```
$ ls -l /usr/lib | cut -c1 | grep d
d
d
d
⋮
```

最後，用 wc 計算行數，我們就會得到由四個命令所組成的管線所產生的答案——*/usr/lib* 包含 145 個子目錄：

```
$ ls -l /usr/lib | cut -c1 | grep d | wc -l
145
```

命令 #5：sort

sort 命令將檔案的每一行，進行重新遞增排序（預設）：

```
$ sort animals.txt
alpaca  Intermediate Perl       2012    Schwartz, Randal
donkey  Cisco IOS in a Nutshell 2005    Boney, James
horse   Linux in a Nutshell     2009    Siever, Ellen
oryx    Writing Word Macros     1999    Roman, Steven
python  Programming Python      2010    Lutz, Mark
robin   MySQL High Availability 2014    Bell, Charles
snail   SSH, The Secure Shell   2005    Barrett, Daniel
```

或遞減（使用 `-r` 選項）：

```
$ sort -r animals.txt
snail   SSH, The Secure Shell   2005    Barrett, Daniel
robin   MySQL High Availability 2014    Bell, Charles
python  Programming Python      2010    Lutz, Mark
oryx    Writing Word Macros     1999    Roman, Steven
horse   Linux in a Nutshell     2009    Siever, Ellen
donkey  Cisco IOS in a Nutshell 2005    Boney, James
alpaca  Intermediate Perl       2012    Schwartz, Randal
```

`sort` 可以按照字母順序（預設）或數字（使用 `-n` 選項）對每一行進行排序。我們將示範對 *animals.txt* 中取得切割後的第三個欄位（出版年份）並進行管線處理：

```
$ cut -f3 animals.txt                       Unsorted
2010
2005
2012
2014
2009
2005
1999
$ cut -f3 animals.txt | sort -n             Ascending
1999
2005
2005
2009
2010
2012
2014
$ cut -f3 animals.txt | sort -nr            Descending
2014
2012
2010
2009
2005
2005
1999
```

要得知 *animals.txt* 中最新一本書的年份，透過 sort 的輸出，再由管線傳遞到 head 的輸入，並僅只列印第一行：

```
$ cut -f3 animals.txt | sort -nr | head -n1
2014
```

最大值和最小值

sort、head 兩者在處理數字資料時是強大的合作夥伴，每一行代表一個值。我們可以透過管線，將資料中的最大值列印輸出：

```
... | sort -nr | head -n1
```

並列印最小值：

```
... | sort -n | head -n1
```

再舉另一個例子，使用檔案 */etc/passwd*，列出可以在系統上執行程式的使用者[4]。我們將產生按字母順序排列的所有使用者的列表。並檢視前五行，會看到如下的內容：

```
$ head -n5 /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
smith:x:1000:1000:Aisha Smith,,,:/home/smith:/bin/bash
jones:x:1001:1001:Bilbo Jones,,,:/home/jones:/bin/bash
```

每一行由冒號分隔的字串所組成，第一個字串是使用者名稱，因此可以使用 cut 命令分隔出使用者名稱：

```
$ head -n5 /etc/passwd | cut -d: -f1
root
daemon
bin
smith
jones
```

並對它們進行排序：

```
$ head -n5 /etc/passwd | cut -d: -f1 | sort
bin
daemon
jones
root
smith
```

4 某些 Linux 系統會將使用者資訊儲存在別的地方。

要產生所有使用者名稱的排序列表，而不僅限於前五個，請將 head 替換為 cat：

```
$ cat /etc/passwd | cut -d: -f1 | sort
```

要檢測某個使用者帳號是否在我們的系統上，需透過 grep 與使用者名稱進行比對。如果沒有任何輸出，表示該帳號並不存在：

```
$ cut -d: -f1 /etc/passwd | grep -w jones
jones
$ cut -d: -f1 /etc/passwd | grep -w rutabaga          (不產生任何輸出)
```

-w 選項通知 grep 只比對完整的單字，而非部分單字，用來防止系統中也有包含「jones」的使用者名稱，例如：sallyjones2。

命令 #6：uniq

uniq 命令用來檢測檔案中相鄰重複的行。預設情況下，會刪除重複項目。我們將用一個包含大寫字母的檔案來做示範：

```
$ cat letters
A
A
A
B
B
A
C
C
C
C
$ uniq letters
A
B
A
C
```

請留意，我們透過 uniq 將前三行 A 減少為一個 A，但保留例子中最後一個 A，因為它與前三行不相鄰。

我們還可以使用 -c 選項，計算出現次數：

```
$ uniq -c letters
      3 A
      2 B
      1 A
      4 C
```

我們必須承認，第一次遇到 uniq 命令時，我們並不認為它有多大用處。但很快地，它就成為了我們最愛的工具之一。假設我們有一個學生大學課程的期末成績檔案，以 tab 字元作分隔，範圍從 A（最好）到 F（最差）：

```
$ cat grades
C       Geraldine
B       Carmine
A       Kayla
A       Sophia
B       Haresh
C       Liam
B       Elijah
B       Emma
A       Olivia
D       Noah
F       Ava
```

想要列印出現次數最多的成績（即使成績相同，只需要列印其中之一）。首先用 cut 分隔開等級，並對其進行排序：

```
$ cut -f1 grades | sort
A
A
A
B
B
B
B
C
C
D
F
```

接下來，使用 uniq 命令得到相鄰行：

```
$ cut -f1 grades | sort | uniq -c
      3 A
      4 B
      2 C
      1 D
      1 F
```

然後依照相反的數字順序對內容進行排序，將最常出現的成績移到第一行：

```
$ cut -f1 grades | sort | uniq -c | sort -nr
      4 B
      3 A
```

```
      2 C
      1 F
      1 D
```

並只保留最前面的第一行：

```
$ cut -f1 grades | sort | uniq -c | sort -nr | head -n1
      4 B
```

最後，由於我們只需要英文字母的成績等級，而非計算數字，再次使用 `cut` 分隔等級：

```
$ cut -f1 grades | sort | uniq -c | sort -nr | head -n1 | cut -c9
B
```

這就是答案，歸功於組合出很長的六個命令的管線。這種循序漸進的管線架構不僅僅是一種潛移默化的動作。這也是 Linux 進階專家實際工作的方式。第 8 章我們會專門介紹此技術。

檢測重複檔案

讓我們將先前學到的知識再與一個更大的範例結合起來。假設我們在一個充滿 JPEG 檔案的目錄中，想知道是否有重複的檔案：

```
$ ls
image001.jpg  image005.jpg  image009.jpg  image013.jpg  image017.jpg
image002.jpg  image006.jpg  image010.jpg  image014.jpg  image018.jpg
⋮
```

我們現在可以用管線來回答這個問題。此外還需要另一個命令 `md5sum` 的協助，它檢查檔案的內容，並計算出一個 32 個字元的字串，稱為**驗證碼**（*checksum*）：

```
$ md5sum image001.jpg
146b163929b6533f02e91bdf21cb9563  image001.jpg
```

依照數學理論，指定檔案所計算出來的驗證碼，其結果非常、非常可能是唯一的。因此，如果兩個檔案具有相同的驗證碼，則幾乎可以肯定它們是重複的。以下 `md5sum` 表示第一、第三個檔案是重複的：

```
$ md5sum image001.jpg image002.jpg image003.jpg
146b163929b6533f02e91bdf21cb9563  image001.jpg
63da88b3ddde0843c94269638dfa6958  image002.jpg
146b163929b6533f02e91bdf21cb9563  image003.jpg
```

當只有三個檔案時，重複的驗證碼很容易透過肉眼檢視出來，但是如果有三千個怎麼辦？管線就是救援的工具。首先計算所有檔案的驗證碼，使用 cut 分隔每一行的前 32 個字元，並對這些內容進行排序，使得任何重複內容彼此相鄰：

```
$ md5sum *.jpg | cut -c1-32 | sort
1258012d57050ef6005739d0e6f6a257
146b163929b6533f02e91bdf21cb9563
146b163929b6533f02e91bdf21cb9563
17f339ed03733f402f74cf386209aeb3
⋮
```

再加入 uniq 來計算重複的部分：

```
$ md5sum *.jpg | cut -c1-32 | sort | uniq -c
      1 1258012d57050ef6005739d0e6f6a257
      2 146b163929b6533f02e91bdf21cb9563
      1 17f339ed03733f402f74cf386209aeb3
      ⋮
```

如果沒有重複內容，則 uniq 產生的所有計算數量都將是 1。依照數字從高到低對結果進行排序，任何大於 1 的數字都將出現在輸出的最上方：

```
$ md5sum *.jpg | cut -c1-32 | sort | uniq -c | sort -nr
      3 f6464ed766daca87ba407aede21c8fcc
      2 c7978522c58425f6af3f095ef1de1cd5
      2 146b163929b6533f02e91bdf21cb9563
      1 d8ad913044a51408ec1ed8a204ea9502
      ⋮
```

接下來刪除非重複部分。驗證碼的前方有六個空格、一個數字和一個空格。我們使用 grep -v 刪除這些行[5]：

```
$ md5sum *.jpg | cut -c1-32 | sort | uniq -c | sort -nr | grep -v "      1 "
      3 f6464ed766daca87ba407aede21c8fcc
      2 c7978522c58425f6af3f095ef1de1cd5
      2 146b163929b6533f02e91bdf21cb9563
```

最後，我們得到一個重複驗證碼列表，依照出現次數排序，這是由六個命令組成的漂亮管線所產生的結果。如果沒有任何輸出，則表示沒有重複檔案。

5 從技術上來說，grep 已經刪除了所有非重複項目，因此在管線中我們其實不需要最後的 sort -nr 命令。

如果命令可以顯示重複的檔案名稱，會顯得更加有用，但其中操作所需要的功能是我們尚未討論（將在第 98 頁的「改進重複檔案檢測工具」小節中學習它們）。現在，透過使用 grep 搜尋來辨識具有指定驗證碼的檔案：

```
$ md5sum *.jpg | grep 146b163929b6533f02e91bdf21cb9563
146b163929b6533f02e91bdf21cb9563  image001.jpg
146b163929b6533f02e91bdf21cb9563  image003.jpg
```

並使用 cut 清理輸出的內容：

```
$ md5sum *.jpg | grep 146b163929b6533f02e91bdf21cb9563 | cut -c35-
image001.jpg
image003.jpg
```

總結

我們現在已經看到標準輸入（stdin）、標準輸出（stdout）和管線（pipe）的強大功能。透過他們將少量命令組合變成可用的工具集合，間接證實了合作能創造的成果大於單打獨鬥的總和。任何讀取標準輸入或寫入標準輸出的命令，都可以加入到管線之中[6]。隨著學習更多的命令，我們可以套用本章中的一般概念，來打造屬於自己的強大組合。

6 某些命令不使用標準輸入 / 標準輸出，因此無法從管線讀取或寫入。例如：mv、rm。然而，管線可能會以其他方式合併這些命令；我們將在第 8 章中看到相關範例。