debian

Debian 參考手冊

Osamu Aoki (青木修)

Debian 參考手冊（第 2.113 版）(2024-02-02 13:34:43 UTC) 旨在爲運行 Debian 系統的用戶提供全面的指引。通過爲非開發者編撰的 shell 指令例子來涵蓋系統管理的方方面面。

COLLABORATORS

	TITLE : Debian 参考手冊	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Osamu Aoki (青木修)	February 2, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

List of Tables

Abstract

這本書是自由且免費的；你可以在與 Debian 自由軟體引導方針（DFSG）兼容的任意版本的 GNU 通用公共許可證的條款下重新發布和/或修改本書。

序

Debian 參考手冊（第 2.113 版）(2024-02-02 13:34:43 UTC) 旨在爲運行 Debian 系統的使用者提供全面的指導。

本書的目標讀者：願意學習 shell 腳本，但是不準備爲了理解 GNU/Linux 系統是如何運作的而閱讀其所有 C 語言原始碼的人。

安裝說明，請見：

- Debian GNU/Linux 當前穩定系統安裝指南
- Debian GNU/Linux 當前測試系統安裝指南

免責聲明

所有擔保條款具有免責效力。所有商標均爲其各自商標所有者的財產。

Debian 系統本身是一個變化的事物。這導致其文件難於及時更新並且正確。雖然是以 Debian 系統當前的 測試版作為寫作該文件的基礎，但當你閱讀本文的時候，部分內容仍然可能已經過時。

請把本文檔作爲第二參考。本文檔不能夠代替任何官方指導手冊。文檔作者和文檔貢獻者對在本文檔中的錯誤、遺漏或歧義，不承擔責任後果。

什麼是 **Debian**

Debian 計畫 是由個人組成的團體，把創建自由的作業系統作爲共同目標。Debian 的發佈具有下列特徵。

- 承諾軟體自由：Debian 社群契約和 Debian 自由軟體引導方針（DFSG）
- 基於網際網路上無酬勞的志願者的貢獻：https://www.debian.org
- 大量預編譯的高品質軟體包
- 專注於穩定性和安全性，同時易於獲得安全更新
- 致力於使 `testing` 版存檔庫中的軟體包都能平順升級至最新版本
- 支援大量硬體架構

Debian 系統中的自由軟體來自GNU, Linux, BSD, X, ISC, Apache, Ghostscript, Common Unix Printing System , Samba, GNOME, KDE, Mozilla, LibreOffice, Vim, TeX, LaTeX, DocBook, Perl, Python, Tcl, Java, Ruby, PHP, Berkeley DB, MariaDB, PostgreSQL, SQLite, Exim, Postfix, Mutt, FreeBSD, OpenBSD, Plan 9 以及許多更加獨立的自由軟體項目。Debian 將上述各式各樣的自由軟體集成到一個系統裏面。

關於本文檔

引導原則

寫作本文檔時，遵循下列指導原則。

- 僅提供概覽，而忽略極端情況。(**Big Picture** 原則)
- 保持文字簡潔。(**KISS** 原則)
- 不重複造輪子。(使用鏈接指向已有參考)
- 專注於使用非圖形的工具和控制檯。(使用 **shell** 例子)
- 保持客觀。(使用 popcon 等等。)

提示
我試圖闡明作業系統底層和階層體系的各方面內容。

預備知識

警告
閱讀本文檔，你需要通過自己的努力去查找本文檔未提及的問題答案。本文檔僅僅提供有效的起點。

你必須自己從以下原始材料查找解決方案。

- Debian 網站 (https://www.debian.org) 上的通用資訊
- "`/usr/share/doc/`*`package_name`*" 目錄下的文檔
- Unix 風格的 **manpage**: "`dpkg -L` *`package_name`* `|grep '/man/man.*/'`"
- GNU 風格的 **info page**: "`dpkg -L` *`package_name`* `|grep '/info/'`"
- 錯誤報告：https://bugs.debian.org/*package_name*
- Debian Wiki （ https://wiki.debian.org/) 用於變化和特殊的議題
- 國際開放標準組織的的單一 UNIX 規範 UNIX 系統主頁上
- 免費的百科全書：維基百科 (https://www.wikipedia.org/)
- Debian 管理員手冊
- 來自 Linux 文件專案 (TLDP)的 HOWTO

注
軟體包的詳細說明文件，你需要安裝軟體包名用"`-doc`" 作爲後綴名的相應文件包。

排版約定

本文通過如下使用 bash(1) shell 指令例子的簡要方式來提供資訊。

```
# command-in-root-account
$ command-in-user-account
```

這些 shell 提示字元區分了所使用的帳戶。爲了可讀性，在本手冊中 shell 提示字元相關的環境變數被設置爲“PS1='\$'”和“PS2=' '。這與實際安裝的系統所使用的 shell 提示字元很有可能會不同。

所有指令範例都運行在英語語言環境下”LANG=en_US.UTF8”. 請不用期待 placeholder 字串：指令於 *root* 使用者和指令於一般使用者被翻譯於指令範例. 這是為了全版本統一

注
見”$PS1” 與”$PS2” 環境變數於 bash(1) 內的解譯。

要求系統管理員執行的操作，須用祈使句描述，如“在 shell 中輸入指令字串後，輸入 Enter 鍵。”

這些描述列或類似資訊在表格有一個名詞短語，後面會緊跟軟體包短描述，這些短語會省略掉前面的”a” 和”the”。它們也可以包含一個不定式短語作名詞短語，在聯機幫助的短指令描述約定後面不帶”to”。有些人可能覺得這看起來有點可笑，這裏故意保留這種風格是爲了讓文檔看起來儘可能的簡單。這些名詞短語在短指令描述約定裏並不會採用首字母大寫的方式。

注
無論專有名詞和指令名位於何處，保持其英文字母大小寫不變。

在文本段落中引用的片斷指令由雙引號括起來的打字機字體進行標記，就像”aptitude safe-upgrade”。

在文本段落中引用的來自調配文件的文本數據由雙引號括起來的打字機字體進行標記，就像”deb-src”。

指令和置於其後的圓括號內的手冊頁章節數（可選），由打字機字體進行標記，就像 bash(1)。我們鼓勵您這樣藉由輸入以下指令來獲得資訊。

```
$ man 1 bash
```

manpage 會在打字機字體後面括號中顯示 manpage 頁章節號，如 sources.list(5)。建議你藉由輸入以下命令來獲得幫助資訊。

```
$ man 5 sources.list
```

info page 頁是由雙引號之間的打字機字體來標註，如 info make。建議你藉由輸入以下的指令來獲得幫助資訊。

```
$ info make
```

文件名將由雙引號括起來的打字機字體進行標記，就像”/etc/passwd”。對於調配文件，你可以輸入下列的命令來獲得它的資訊。

```
$ sensible-pager ”/etc/passwd”
```

目錄名將由雙引號括起來的打字機字體進行標記，如”/etc/apt/”。你可以輸入下列的指令來瀏覽目錄的內容。

```
$ mc ”/etc/apt/”
```

軟體包名稱將由打字機字體進行標記，就像 vim。你可以輸入下列的指令來獲得它的資訊。

```
$ dpkg -L vim
$ apt-cache show vim
$ aptitude show vim
```

一個文檔可以通過文件名來表示它的位置,文件名將由雙引號括起來的打字機字體進行標記,例如”/usr/share/doc/base-
和” /usr/share/doc/base-passwd/users-and-groups.html”,或通過它的 URL 表示,如 https://www.debian.org。
你可以藉由輸入下列指令來閱讀文檔。

```
$ zcat ”/usr/share/doc/base-passwd/users-and-groups.txt.gz” | sensible-pager
$ sensible-browser ”/usr/share/doc/base-passwd/users-and-groups.html”
$ sensible-browser ”https://www.debian.org”
```

環境變數將由雙引號括起來的打字機字體進行標記，並帶有”$” 前綴，就像”$TERM”。你可以輸入下列指令來獲得它的當前值。

```
$ echo ”$TERM”
```

popcon 流行度

popcon數據被用來客觀地衡量每個包的流行度。它的下載時間爲 2024-01-25 09:08:50 UTC，包含了超過 196182 個二進位軟體包和 27 個架構的全部 233460 份提交。

注
請注意 amd64 不穩定的 archive 當前只包含 71664 個。popcon 數據包含許多舊系統安裝報告。

以“V:”開頭表示“votes”的 popcon 數值計算方式爲“1000 * （當前運行在 PC 上的包的 popcon 提交）/（總的 popcon 提交）”。

以“I:”開頭表示“安裝數”的 popcon 數值計算方式爲“1000 * （當前安裝在 PC 上的包的 popcon 提交）/（總的 popcon 提交）”。

注
流行度評比 popcon 數據不應視爲對包的重要性的絕對度量。有許多因素可以影響統計數據。例如，參與流行度評比的某些系統可能有像“/usr/bin”的目錄，掛載的時候帶“noatime”選項以提升系統性能，這樣的系統有效的禁用了“投票（vote）”功能。

包大小

包的尺寸數據同樣表明瞭對每個包的客觀衡量。它基於“apt-cache show”或“aptitude show”指令（當前在 amd64 的不穩定釋出的架構）報告的“安裝大小”。報告的尺寸的單位是 KiB（Kibibyte= 表示 1024 Bytes 的單位）。

注
包大小是一個小數值的包可能顯示了這個在“不穩定”釋出的包是一個虛擬包，它包含關於依賴關係的重要內容，會安裝其他的包。虛擬包使能平穩過度或分割一個包。

注
包大小後面跟着”(*)” 表明這個軟體包在不穩定版本中是缺失的同時使用了實驗性版本中的軟體包大小來替代。

給本文檔報告 Bug

發現此文件的任何疑義，請報告 debian-reference 套件的錯誤，使用 reportbug(1)。請以”diff -u” 提出純文字或原碼的更正後報告。

一些對新使用者的提醒

這是一些對新使用者的提醒：

- 備份你的資料
  - 參見節 10.2。
- 保護密碼和安全金鑰
- KISS 原則
  - 不要過度設計您的系統
- 閱讀您的日誌文件
  - 第一條錯誤資訊才是最重要的
- RTFM（閱讀手冊與指導）
- 問問題前，請先上網找資料
- 盡量使用一般使用者，除非不得已
- 不要胡亂折騰軟體包管理系統
- 別輸入你不懂的指令
- 不更改檔案權限，除非你確認過安全
- 在測試過你所做的修改之前不要關閉 root shell
- 保持擁有一個替代的啟動裝置 (USB 開機碟, CD, ⋯)

一些對新使用者的引導

從 Debian 郵件列表來的一些有趣引文，說不定可以幫助新使用者啓發。

- “這是 Unix。它給你足夠的繩索來吊死你自己。” --- Miquel van Smoorenburg <miquels at cistron.nl>
- “Unix 是使用者友好的⋯⋯它僅僅選擇誰是它的朋友。” --- Tollef Fog Heen <tollef at add.no>

維基百科文章”Unix 哲學” 列出了一些有趣的指導。

Chapter 1

GNU/Linux 教學

我想，當你學習一個電腦系統，就像學習一門新的外語。雖然教學和文檔是有幫助的，但你必須自己練習。爲了幫助你順利的開始，我詳細說明一些基本要點。

Debian GNU Linux中最強大的設計來自Unix作業系統，一個多人多工的作業系統。你必須學會利用這些特性以及 Unix 和 GNU/Linux 的相似性。

別迴避面對 Unix 的文檔，不要只是依賴於 GNU/Linux 文檔，這樣做會剝奪你瞭解許多有用的資訊。

注
如果你在任何類 Unix系統中使用過一段時間的命令列工具，你可能已經掌握了這份文檔中的內容。那請把它當做一個實戰檢驗和進修。

1.1 控制臺基礎

1.1.1 shell 提示字元 (prompt)

啟動系統之後，如果你沒有安裝 GUI (例如GNOME 或者 KDE)，那麼你會看到字元登入介面。假設你的主機名為 `foo`，那麼登入提示符將如下所示。

如果你安裝了一個 GUI 環境，那麼你仍然能夠用 Ctrl-Alt-F3 進入基於字元的登入提示符，同時你能透過 Ctrl-Alt-F2 回到 GUI 環境 (更多詳情請參閱下文節 1.1.6)。

```
foo login:
```

在登錄提示字元下，你輸入你的使用者名，例如 `penguin`，然後按 Enter 鍵，接下來輸入你的密碼並再次按 Enter 鍵。

注
遵循 Unix 傳統，Debian 系統下的使用者名和密碼是大小寫敏感的。使用者名通常由小寫字母組成。第一個使用者帳號通常在安裝期間進行創建。額外的使用者帳號由 root 使用者用 `adduser`(8) 創建。

系統以保存在"`/etc/motd`" 中的歡迎資訊 (Message Of The Day) 來開始，同時顯示一個指令提示字元。

```
Debian GNU/Linux 12 foo tty3

foo login: penguin
Password:
```

```
Linux foo 6.5.0-0.deb12.4-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.5.10-1~bpo12+1 (2023-11-23) ↩
     x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Last login: Wed Dec 20 09:39:00 JST 2023 on tty3
foo:~$
```

現在，你就在 shell 下。shell 解析你的指令。

1.1.2 GUI 下的 shell 提示符

如果你在安裝 Debian 的過程中，安裝了一個 GUI 環境，那麼你在啟動系統後將使用圖形登入介面。輸入你的使用者名稱和密碼可以登入到非特權使用者帳號。使用 Tab 鍵（跳格鍵）可以在使用者名稱和密碼之間移動，也可以使用滑鼠主要鍵點選。

要在 GUI（圖形使用者介面）環境下獲得 shell 提示符，你必須啟動一個 `x` 終端模擬器程式，例如 `gnome-terminal`(1)、`rxvt`(1) 或 `xterm`(1)。在 GNOME 桌面環境下，你可以按超級鍵（Windows 鍵），在搜尋提示裡輸入”terminal” 來開啟終端。

在其它一些桌面系統（如 `fluxbox`）下面，可能沒有明顯的開始選單入口。如果是這種情況，試下右擊桌面螢幕並希望能有彈出選單。

1.1.3 root 帳號

root 帳號也被稱作超級使用者或特權使用者。用這個帳號，你能夠履行下面的系統管理任務。

- 讀、寫和刪除系統上的任何文件，不顧它們的文件權限
- 設置系統上任何文件的所有者和權限
- 設置系統上任何非特權使用者的密碼
- 免使用者密碼登錄任何使用者

root 帳號擁有至高無上的權力，你要慎重和負責任的使用。

警告
從來不和其他人共享 root 密碼。

注
一個文件（包括硬體設備，如 CD-ROM 等，這些對 Debian 系統來說都只是一個文件）的權限可能會導致非 root 使用者無法使用或存取它。雖然在這種情況下，使用 root 帳戶是一個快速的方法，但正確的解決方法應該是對文件權限和使用者組的成員進行合適的設置（參見節 1.2.3）。

1.1.4 root shell 提示字元

這裏有一些基本的方法可以讓你在輸入 root 密碼後獲得 root 的 shell 提示字元。

- 在字元登入界面使用 `root` 作爲使用者名稱登入。
- 在任意使用者的 shell 提示字元下輸入“`su -l`”。
  - 這不會保存當前使用者的環境設定。
- 在任意使用者的 shell 提示字元下輸入“`su`”。
  - 這會保存當前使用者的一些環境設定。

1.1.5 GUI 系統管理工具

如果你的桌面選單沒有使用適當許可權啟動 GUI (圖形使用者介面) 的自動化管理工具，你可以在終端模擬器 (例如 `gnome-terminal`(1)、`rxvt`(1) 或 `xterm`(1)) 中 root 的 shell 提示符下啟動它。參見節 1.1.4和節 7.8。

警告
永遠不要在顯示管理器 (例如 `gdm3`(1)) 的提示符下輸入 `root` 來使用 root 賬戶啟動 GUI 顯示/會話管理器。
永遠不要在顯示關鍵資訊的 X Window 下運行不受信任的遠程 GUI 程序，因爲它可能會監聽你的 X 螢幕。

1.1.6 虛擬控制檯

在預設的 Debian 系統中，有 6 個可切換的類 VT100字元控制檯，可以直接在 Linux 主機上啓動 shell。除非你處於 GUI 環境下，否則你可以同時按下左 `Alt` 鍵和 `F1`—`F6` 之一的鍵在虛擬控制檯間切換。每一個字元控制檯都允許獨立登入帳號並提供多使用者環境。這個多使用者環境是偉大的 Unix 的特性，很容易上癮。

如果你處於 GUI 環境中，你可以透過 `Ctrl-Alt-F3` 鍵前往字元控制檯 3，也就是同時按下左 `Ctrl` 鍵、左 `Alt` 鍵和 `F3` 鍵。你可以按下 `Alt-F2` 回到 GUI 環境，它一般執行在虛擬控制檯 2。

你也可以使用命令列切換到另一個虛擬控制檯，例如切換到控制檯 3。

```
# chvt 3
```

1.1.7 怎樣退出命令列提示字元

在命令列輸入 `Ctrl-D`，即同時按下左側`-Ctrl-`鍵和 `d-`鍵，即可關閉 shell 活動。如果你正處於字元控制檯，你將會回傳到登錄提示行。儘管這些控制字元“control D” 使用了大寫字母，你並不需要按住 Shift-鍵。`Ctrl-D` 也可以簡寫爲 `^D`。或者，你也可以輸入”exit” 退出命令列。

如果你位於 `x` 終端模擬器 (1) 中，你可以使用這個關閉 `x` 終端模擬器視窗。

1.1.8 怎樣關閉系統

就像任何其他的現代作業系統一樣，Debian 會通過記憶體中的快取數據進行文件操作以提高性能，因此在電源被安全地關閉前需要適當的關機過程，通過將記憶體中的數據強制寫入硬盤來維持文件的完整性。如果軟體的電源控制可用，那麼關機過程中會自動關閉系統電源。(否則，你可能需要在關機過程之後按電源鍵幾秒鐘。)

在普通多使用者模式模式下，可以使用命令列關閉系統。

```
# shutdown -h now
```

在單使用者模式下，可以使用命令列關閉系統。

```
# poweroff -i -f
```

參見節 6.3.8。

1.1.9 恢復一個正常的控制檯

當做了一些滑稽的事（例如“cat 二進位文件”）後，螢幕會發狂，你可以在命令列輸入“reset”。你可能無法在螢幕上看到你輸入的指令。你也可以輸入“clear”來清理視窗。

1.1.10 建議新手的額外軟體包

不需任何桌面環境，就能執行 Debian 的最小安裝方式，但若能使用 mc 和 vim 併用 apt-get(8)，對初學者而言，仍是有用的。

```
# apt-get update
 ...
# apt-get install mc vim sudo aptitude
 ...
```

如果你已經安裝了這些軟體包，那麼不會有新的軟體包被安裝。

軟體包	流行度	大小	說明
mc	V:49, I:212	1542	文本模式的全螢幕文件管理器
sudo	V:680, I:838	6550	給普通使用者授予部分 root 權限的程序
vim	V:91, I:370	3743	Unix 文本編輯器 Vi 的改進版，一個程式設計師的文本編輯器（標準版）
vim-tiny	V:55, I:974	1722	Unix 文本編輯器 Vi 的改進版，一個程式設計師的文本編輯器（精簡版）
emacs-nox	V:3, I:16	35109	GNU 項目的 Emacs，基於 Lisp 的擴展文本編輯器
w3m	V:14, I:188	2837	文本模式的 www 瀏覽器
gpm	V:10, I:12	521	文本控制檯 Unix 式樣的貼上拷貝（後台）

Table 1.1: 有趣的文本模式程序包列表

閱讀一些資訊文檔，也是一個好的主意。

軟體包	流行度	大小	說明
doc-debian	I:865	187	Debian 項目文檔，（Debian 常見問題）和其它文檔
debian-policy	I:15	4379	Debian 策略手冊和相關文檔
developers-reference	V:0, I:5	2601	Debian 開發者引導方針和資訊
debmake-doc	I:0	11701	Debian 維護者手冊
debian-history	I:0	4692	Debian 項目歷史
debian-faq	I:863	790	Debian 常見問題

Table 1.2: 軟體包資訊文檔列表

你可以用下面的指令安裝這些包。

```
# apt-get install package_name
```

1.1.11 額外使用者帳號

如果你不想用你自己的主使用者帳號來進行下面的練習操作，你可以使用下面的方式創建一個練習使用者帳號，比如說，創建一個使用者名爲 fish 的賬號。

```
# adduser fish
```

回答所有問題。

這將創建一個名爲 fish 的新帳號。在你練習完成後，你可以使用下面的指令刪除這個使用者帳號和它的使用者主目錄。

```
# deluser --remove-home fish
```

1.1.12 sudo 調配

對於典型的單使用者工作站，例如運行在筆記本電腦上的桌面 Debian 系統，通常簡單地調配 sudo(8) 來使爲非特權使用者（例如使用者 penguin）只需輸入使用者密碼而非 root 密碼就能獲得管理員權限。

```
# echo "penguin  ALL=(ALL) ALL" >> /etc/sudoers
```

另外，可以使用下列指令使非特權使用者（例如使用者 penguin）無需密碼就獲得管理員權限。

```
# echo "penguin  ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers
```

這些技巧只對你管理的單使用者工作站中那個唯一的使用者有用。

警告
在多使用者工作站中不要建立這樣的普通使用者帳號，因爲它會導致非常嚴重的系統安全問題。

注意
在上述例子中，使用者 penguin 的密碼及帳號要有和 root 帳號密碼同樣多的保護。
在這種情況下，管理員權限被賦予那些有權對工作站進行系統管理任務的人。永遠不要讓你的公司行政管理部門或你的老闆進行管理（例如給予他們權限），除非他們獲得了授權並有這樣的能力。

注
爲了對受限的設備和文件提供存取權限，你應該考慮使用 羣組提供的近用限制而不是使用 root 權限，來自 sudo(8)。
隨着越來越細緻周密的調配，sudo(8) 可以授予一個共享系統上的其它使用者有限的管理權限而不共享 root 密碼。這可以幫助對有多個管理員的主機進行責任追究，你可以瞭解到是誰做什麼。另一方面，你可能不想任何人有這樣的權限。

1.1.13 Play time

現在你已經準備好使用 Debian 系統了，只要你使用非特權使用者帳號就不會有風險。

這是因爲 Debian 系統（即使是預設安裝）會設置適當的文件權限來防止非特權使用者對系統造成破壞。當然，可能仍然有一些漏洞可以利用，但關心這些問題的人不應該閱讀這一節，而應該去閱讀 Debian 安全手冊。

我們使用下面的方式，把 Debian 系統當作一個 類 Unix 系統來學習。

- 節 1.2 (基本概念)
- 節 1.3 (生存方式)
- 節 1.4 (基本方式)
- 節 1.5 (shell 機制)
- 節 1.6 (文本處理方式)

1.2 類 Unix 檔案系統

在 GNU/Linux 和其他類 Unix作業系統中，文件被組織到目錄中。所有的文件和目錄排放在以“/”爲根的巨大的樹裏。叫它樹是因爲如果你畫出檔案系統，它看起來就像一棵樹，只是它是顛倒過來的。

這些文件和目錄可以分散在多個設備中。mount(8) 用於把某個設備上找到的檔案系統附着到巨大的文件樹上。相反的，umount(8) 把它再次分離。在當前的 Linux 核心裏 mount(8) 帶某些參數，可以把文件樹的一部分綁定到另外的地方，或者可以把檔案系統掛載爲共享的、私有的、從設備、或不可綁定的。對每個檔案系統支援的掛載選項可以在/usr/share/doc/linux-doc-*/Documentation/filesystems/找到。

Unix 系統上叫做目錄，某些其他系統上叫做文件夾。請同樣留意，在任何 Unix 系統上，沒有的驅動器的概念，例如“A:”。這只有一個檔案系統，並且所有東西都包含在內。這相對於 Windows 來說是一個巨大的優點。

1.2.1 Unix 文件基礎

下面是一些 Unix 文件基礎。

- 文件名是 區分大小寫的。也就是說，”MYFILE” 和”MyFile” 是不同的文件。
- 根目錄意味着檔案系統的根，簡單的稱爲“/”，不要把它跟 root 使用者的家目錄“/root”混淆了。
- 每個目錄都有一個名字，它可以包含任意字母或除了/”以外的符號。根目錄是個特例。它的名字是“/”(稱作“斜線”或“根目錄”)，並且它不能被重命名。
- 每個文件或目錄都被指定一個全限定文件名，絕對文件名，或路徑，按順序給出必須經過的目錄從而到達相應目錄。這三個術語是同義的。
- 所有的全限定文件名以“/”目錄開始，並且在每個目錄或文件名之間有一個“/”。第一個“/”是最頂層目錄，其他的“/”用於分隔跟着的子目錄。直到到達最後的入口，即實際文件的名稱。這些話可能會令人困惑。用下面這個全限定文件名作爲例子:“/usr/share/keytables/us.map.gz”。不過，人們也把它的基名“us.map.gz”單獨作爲文件名。
- 根目錄有很多分支，例如“/etc/”和“/usr/”。這些子目錄依次分出更多的子目錄，例如“/etc/systemd/”和“/usr/local/”。這整體叫做“目錄樹”。你可以把一個絕對文件名想象成從“/”這棵樹的基到某個分支 (一個文件) 的結尾的一條路徑。你也聽到人們談論目錄樹，就好像它是一個包含所有直系後代的“家庭”樹的一個圖，這個圖叫做根目錄 (“/”): 因此子目錄有父目錄，並且一條路徑顯示了一個文件完整的祖先。也有相對路徑從其他地方開始，而不是從根目錄。你應該還記得目錄“../”指向父目錄。這個術語也適用於其他類似目錄的結構，如分層數據結構。
- 對於一個實體設備, 是沒有一個特定的目錄路徑名來對應的組成部分. 這不同於RT-11, CP/M,OpenVMS,MS-DOS,AmigaOS, 以及微軟的 Windows，這些系統存在一個路徑包含了一個設備名字，比如”C:\”。(儘管如此, 路徑條目確實存在引用了物理設備作爲正常的檔案系統的一部分. 參考節 1.2.2。)

注
雖然你可以在文件名中使用任意的字幕或者符號, 但是在實際情況下這樣做是一個壞主意. 最好避免使用一些在命令列裏面含有特殊意義的字元, 比如空格, 製表符, 換行符, 和其它的特殊字元: { } ( ) [ ] ’ ` ” \ / >< | ; ! #&^ * % @ $. 如果你想有一個區分度良好的命名, 比較好的選擇是利用時期, 連字元和下劃線. 你也可以每個單詞的首字母大寫, 這叫大駝峯命名法, 比如這樣”LikeThis”. 經驗豐富的 Linux 使用者會趨向於在文件名中不使用空格.

注
這個”root” 可能既表示” 超級使用者 root” 又表示” 根目錄”(/root) . 應該根據上下文確定它的用法.

注
單詞 **path** 不僅表示包含全限定文件名, 也可能表示指令搜索的路徑. 通常路徑真實的意思是需要通過上下文來明確.

關於檔案階層的最佳詳細實踐在檔案系統階層標準 (”/usr/share/doc/debian-policy/fhs/fhs-2.3.txt.gz” 和 hier (7)). 你應該記住以下的一些標準作爲開始學習的步驟.

目錄	目錄用途
/	根目錄
/etc/	系統範圍的調配文件
/var/log/	系統日誌文件
/home/	所有非特權使用者的使用者目錄

Table 1.3: 重要目錄的用途列表

1.2.2 檔案系統深入解析

按照 **UNIX** 系统的傳統，Debian GNU / Linux 的文件系统是在硬體數據儲存設備，諸如硬碟或其他的儲存設備上，與硬體的互動，如控制台和遠端序列終端都是以統一的方式呈現在“/ dev /”下面。

每個檔案、目錄、命名管道（一種兩個程式間共享資料的方法）或 Debian GNU/Linux 系統上的物理裝置都有一個叫做 inode的資料結構，描述了其相關特性，例如擁有它的使用者（所有者），它屬於的群組，最後一次存取時間，等等。把所有東西都表示在檔案系統中的想法是來源於 Unix，現代的 Linux 核心則將這個思路進行了擴充。現在，甚至有關計算機上正在執行的程序的資訊都可以在檔案系統中找到。

這個對物理實體和內部行程的統一和抽象是非常強大的，因爲這允許我們用同樣的指令對許多完全不同的設備進行同樣的操作。甚至可以通過向鏈接到運行行程的特殊文件寫入數據來改變核心的運行方式。

提示
如果你需要識別文件樹和物理實體之間的對應關係，不帶參數運行 mount(8)。

1.2.3 檔案系統權限

類 Unix系統的檔案系統權限被定義給三類受影響的使用者。

- 擁有這個文件的使用者（**u**）
- 這個文件所屬組的其他使用者（**g**）
- 所有其餘的使用者（**o**），同樣稱爲“世界”和“所有人”

對文件來說，每個對應權限允許下列動作。

- 可讀（**r**）權限允許所有者檢查文件的內容。
- 可寫（**w**）權限允許所有者修改文件內容。
- 可執行（**x**）權限允許所有者把文件當做一個指令運行。

對於目錄來說，每個對應權限允許下列動作。

- 可讀（**r**）權限允許所有者列出目錄內的內容。
- 可寫（**w**）權限允許所有者添加或刪除目錄裏面的文件。
- 可執行（**x**）權限允許所有者存取目錄裏的文件。

在這裏，一個目錄的可執行權限意味着不僅允許讀目錄裏的文件，還允許顯示他們的屬性，例如大小和修改時間。

`ls`(1) 用於顯示文件和目錄的權限資訊（更多）。當運行時帶有“`-l`”選項，它將按給定順序顯示下列資訊。

- 文件類型（第一個字母）
- 文件的存取權限（9 個字元，三個字元組成一組按照使用者、組、其他的順序表示）
- 鏈接到文件的硬鏈接數
- 文件所有者的使用者名
- 這個文件所屬的組名
- 以字元（字節）爲單位的文件大小
- 文件的日期和時間（mtime）
- 文件的名字

字元	說明
`-`	普通文件
`d`	目錄
`l`	符號鏈接
`c`	字元裝置節點
`b`	塊裝置節點
`p`	命名管道
`s`	套接字

Table 1.4: “`ls -l`”輸出的第一個字元列表

`chown`(1) 用於 root 帳號修改文件的所有者。`chgrp`(1) 用於文件的所有者或 root 帳號修改文件所屬的組。`chmod`(1) 用於文件的所有者或 root 帳號修改文件和文件夾的存取權限。操作一個 `foo` 文件的基本語法如下。

```
# chown newowner foo
# chgrp newgroup foo
# chmod  [ugoa][+-=][rwxXst][,...] foo
```

例如，你可以按照下面使一個目錄樹被使用者 `foo` 所有，並共享給組 `bar`。

```
# cd /some/location/
# chown -R foo:bar .
# chmod -R ug+rwX,o=rX .
```

有三個更加特殊的權限位。

- **Set_User_ID** 位（**s** 或 **S** 替換使用者的 **x**）
- **Set_Group_ID** 位（**s** 或 **S** 替換組的 **x**）
- 粘置位（**t** 或 **T** 替代其他使用者的 **x**）

這裏“ls -l”對這些位的輸出是大寫的，如果這些輸出裏隱藏了可執行位，則它未設置。

給一個可執行文件設置 **Set-User-ID** 位將允許一個使用者以他自己的 ID 運行這個可執行文件（例如 **root**）。類似的，給一個可執行文件設置了 **Set-Group-ID** 位將允許一個使用者以文件的組 ID 運行該文件。（例如 **root** 組）。由於這些設置可能引起安全風險，使能它們的時候需要格外留意。

在一個目錄上設置“**Set-Group-ID**”將打開類 BSD 的文件創建計劃，所有在目錄裏面創建的文件將屬於目錄所屬的組。

給一個目錄設定“粘滯位”將保護該目錄內的檔案不被其所有者之外的一個使用者刪除。為了保護一個在像“/tmp”這樣所有人可寫或同組可寫的目錄下檔案內容的安全，不僅要去除可寫許可權，還要給其所在目錄設定粘滯位。否則，該檔案可以被任意對其所在目錄有寫許可權的使用者刪除並建立一個同名的新檔案。

這裏有少量有趣的文件權限例子。

```
$ ls -l /etc/passwd /etc/shadow /dev/ppp /usr/sbin/exim4
crw------T 1 root root   108, 0 Oct 16 20:57 /dev/ppp
-rw-r--r-- 1 root root     2761 Aug 30 10:38 /etc/passwd
-rw-r----- 1 root shadow   1695 Aug 30 10:38 /etc/shadow
-rwsr-xr-x 1 root root   973824 Sep 23 20:04 /usr/sbin/exim4
$ ls -ld /tmp /var/tmp /usr/local /var/mail /usr/src
drwxrwxrwt 14 root root  20480 Oct 16 21:25 /tmp
drwxrwsr-x 10 root staff  4096 Sep 29 22:50 /usr/local
drwxr-xr-x 10 root root   4096 Oct 11 00:28 /usr/src
drwxrwsr-x  2 root mail   4096 Oct 15 21:40 /var/mail
drwxrwxrwt  3 root root   4096 Oct 16 21:20 /var/tmp
```

chmod(1) 有另一種數值模式來描述檔案許可權。這種數字模式使用 3 到 4 位八進位制（底為 8）數。

數字	說明
第一個可選數字	**set user ID** (=4), **set group ID** (=2) 和 **sticky bit** (=1) 之和
第二個數字	**read** (=4), **write** (=2), 和 **execute** (=1) 權限之和，**user** 使用者
第三個數字	同上，**group**
第四個數字位	同上，**other**

Table 1.5: chmod(1) 指令文件權限的數字模式

這聽起來很複雜實際上相當簡單。如果你把“ls -l”指令輸出的前幾列（2-10），看成以二進位制（底為 2）表示檔案的許可權（“-”看成 0，“rwx”看成 1），你應該可以理解用數字模式值的最後 3 位數字對檔案許可權的八進位制表示。

嘗試下列例子

```
$ touch foo bar
$ chmod u=rw,go=r foo
$ chmod 644 bar
$ ls -l foo bar
-rw-r--r-- 1 penguin penguin 0 Oct 16 21:39 bar
-rw-r--r-- 1 penguin penguin 0 Oct 16 21:35 foo
```

提示
如果你需要在 shell 指令碼中存取“ls -l”顯示的資訊，你需要使用相關指令，如 test(1)，stat(1) 和 readlink(1)。shell 內建指令，如“[”或“test”，可能也會用到。

1.2.4 控制新建檔案的許可權：umask

什麼許可權將應用到新建檔案受 shell 內建指令 umask 的限制。參見 dash(1)，bash(1)，和內建命令 (7)。

```
(file permissions) = (requested file permissions) & ~(umask value)
```

umask 值	建立的檔案許可權	建立的目錄許可權	用法
0022	-rw-r--r--	-rwxr-xr-x	僅所屬使用者可寫
0002	-rw-rw-r--	-rwxrwxr-x	僅所屬組可寫

Table 1.6: **umask** 值舉例

Debian 預設使用使用者私人組（UPG）。每當一個新使用者新增到系統的時候都會建立一個 UPG。UPG 的名字和建立它的使用者相同，這個使用者是這個 UPG 的唯一成員。自從每個使用者都有自己的私人組之後，把 umask 設定成 0002 變得更安全了。（在某些 Unix 變體中，把所有普通使用者設定到一個叫 **users** 的組是非常常見的做法，在這種情況下，出於安全考慮把 umask 設為 0022 是一個好主意）

提示
通過把“umask 002”寫入 ~/.bashrc 檔案開啟 UPG。

1.2.5 一組使用者的許可權（組）

警告
在做重啟或者類似行為前，確保儲存沒有儲存的修改。

為了使組許可權應用到一個特定使用者，這個使用者需要透過使用“sudo vigr”編輯 /etc/group 以及使用“sudo vigr -s”編輯 /etc/gshadow 成為該組的成員。你需要重啟之後重新登入（或執行“kill -TERM -1”）[1]以啟用新的組配置。

注
或者，你可以通過新增一行“auth optional pam_group.so”到“/etc/pam.d/common-auth”以及調配“/etc/security/group.conf”，使得在身份驗證過程動態新增使用者到組。（參見章 4。）

在 Debian 系統中，硬體裝置是另一種檔案。如果你從一個使用者帳號存取某些裝置出現問題，例如 CD-ROM 和 USB 記憶棒，你需要使這個使用者成為相關組的成員。

一些著名的由系統提供的組允許其成員不需要 root 許可權存取某些特定的檔案和裝置。

組	可存取檔案和裝置的描述
dialout	完全及直接的存取串列埠埠（“/dev/ttyS[0-3]”）
dip	有限的存取串列埠，建立到信任點的撥號 **IP** 連線
cdrom	CD-ROM, DVD+/-RW 驅動器
audio	音訊裝置
video	視訊裝置
scanner	掃描器
adm	系統監控日誌
staff	一些用於初級管理工作的目錄：“/usr/local”，“/home”

Table 1.7: 關於檔案存取的由系統提供的著名組列表

[1]在現代環境下，透過 GUI（圖形使用者介面）選單登出登入，在這裡不一定生效。

提示
你需要屬於 dialout 組使得可以重調配調變解調器，撥號到任意地方，等等。但如果 root 使用者在“/etc/ppp/peers/”為受信任點建立了預定義組態檔案的話，你只需要屬於 dip 組，就可以通過使用 pppd(8)、pon(1)，以及 poff(1) 指令建立撥號 **IP** 連結到這些受信任的點。

某些著名的由系統提供的組允許它們的成員不帶 root 許可權執行特定的指令。

組	可存取指令
sudo	不帶它們的密碼執行 sudo
lpadmin	執行指令以從印表機資料庫新增、修改、移除印表機

Table 1.8: 著名的由系統提供用於特定指令執行的組列表

由系統提供的使用者和組的完整列表,參見由 base-passwd 包提供的“/usr/share/doc/base-passwd/users-and-gr
中，當前版本的“使用者和組”。

使用者和組系統的管理指令,參見 passwd(5),group(5),shadow(5),newgrp(1),vipw(8),vigr(8),以及 pam_group(8)。

1.2.6 時間戳

GNU/Linux 文件有三種類型的時間戳。

類型	含義（歷史上 **Unix** 的定義）
mtime	文件修改時間 (ls -l)
ctime	文件狀態修改時間 (ls -lc)
atime	文件最後被存取的時間 (ls -lu)

Table 1.9: 時間戳類型列表

注
ctime 不是文件創建時間。

注
atime 在 GNU/Linux 系統上的真實值可能和歷史上 Unix 的定義有所不同。

- 覆蓋一個文件，將會改變該文件所有的 **mtime**, **ctime**, 和 **atime** 屬性。
- 改變文件的所有者或者權限，將改變文件的 **ctime** 和 **atime** 屬性。
- 在歷史上的 Unix 系統中，讀取一個檔案將改變檔案的 **atime** 屬性。
- 讀一個檔案，將改變檔案的 **atime** 屬性；在 GNU/Linux 系統上，這僅發生在其檔案系統使用“strictatime”引數掛載的情況下。
- 如果 GNU/Linux 系統的檔案系統使用”relatime” 選項掛載，第一次讀檔案，或者隨後讀檔案，將改變該檔案的 **atime** 屬性. (從 Linux 2.6.30 開始的預設行為)
- 如果 GNU/Linux 系統的檔案系統使用”noatime” 掛載，則讀一個檔案，不會改變這個檔案的 **atime** 屬性。

注
為了在正常的使用場景中能夠提升檔案系統的讀取效率，新增了"noatime" 和"relatime" 這兩個載入選項。如使用了"strictatime" 選項，即使簡單的檔案讀操作都伴隨著更新 **atime** 屬性這個耗時的寫操作。但是 **atime** 屬性除了 mbox(5) 檔案以外卻很少用到。詳情請看 mount(8)。

使用 touch(1) 指令修改已存在檔案的時間戳。

對於時間戳，在非英語區域 (“fr_FR.UTF-8”)，ls 命令輸出本地化字串。

```
$ LANG=C  ls -l foo
-rw-rw-r-- 1 penguin penguin 0 Oct 16 21:35 foo
$ LANG=en_US.UTF-8  ls -l foo
-rw-rw-r-- 1 penguin penguin 0 Oct 16 21:35 foo
$ LANG=fr_FR.UTF-8  ls -l foo
-rw-rw-r-- 1 penguin penguin 0 oct. 16 21:35 foo
```

提示
參考節 9.3.4 自定義“ls -l”輸出。

1.2.7 連結

有兩種方法把一個檔案“foo”連結到一個不同的檔名“bar”。

- 硬連結
  - 對現有檔案重複名稱
  - “ln foo bar”
- 符號連結或 symlink
  - 通過名字指向另一個檔案的特殊檔案
  - “ln -s foo bar”

請參閱下面的示例，rm 指令結果中連結數的變化和細微的差別。

```
$ umask 002
$ echo "Original Content" > foo
$ ls -li foo
1449840 -rw-rw-r-- 1 penguin penguin 17 Oct 16 21:42 foo
$ ln foo bar     # hard link
$ ln -s foo baz  # symlink
$ ls -li foo bar baz
1449840 -rw-rw-r-- 2 penguin penguin 17 Oct 16 21:42 bar
1450180 lrwxrwxrwx 1 penguin penguin  3 Oct 16 21:47 baz -> foo
1449840 -rw-rw-r-- 2 penguin penguin 17 Oct 16 21:42 foo
$ rm foo
$ echo "New Content" > foo
$ ls -li foo bar baz
1449840 -rw-rw-r-- 1 penguin penguin 17 Oct 16 21:42 bar
1450180 lrwxrwxrwx 1 penguin penguin  3 Oct 16 21:47 baz -> foo
1450183 -rw-rw-r-- 1 penguin penguin 12 Oct 16 21:48 foo
$ cat bar
Original Content
$ cat baz
New Content
```

硬連結可以在同一個檔案系統內建立，並共用同一個 inode 號，由 `ls`(1) 帶“`-i`”選項顯示。

符號連結總是名義上具有“`rwxrwxrwx`”的檔案存取許可權，如上面例子所示，實際的有效存取許可權由它所指向的檔案確定。

注意
除非你有非常好的理由，否則不要建立一個複雜的符號連結或硬連結通常是個好主意。符號連結的邏輯組合可能導致檔案系統噩夢般的無限迴圈。

注
通常使用符號連結比使用硬連結更合適，除非你有一個好理由使用硬連結。

“`.`”目錄連結到它所在的目錄，因此任何新建目錄的連結數從 2 開始。“`..`”目錄連結到父目錄，因此目錄的連結數隨著新的子目錄的建立而增加。

如果你剛從 Windows 遷移到 Linux，你很快將清楚 Unix 的檔名連結相較於 Windows 最相近的“快捷方式”是多麼精心設計的。由於它是在檔案系統中實現的，應用無法看到連結檔案跟原始檔案之間的區別。在硬連結這種情況，這真的是毫無差別。

1.2.8 命名管道（先進先出）

命名管道是一個像管道一樣的檔案。你把內容放進了檔案，它從另一端出來。因此，它被稱為 FIFO，即先進先出：你從管道這端先放進去的東西會從另一端先出來。

如果對一個命名管道進行寫入操作，寫入的過程不會被終止，直到寫入的資訊從管道中被讀取出來。讀取過程將會持續到沒有資訊可以讀取為止。管道的大小始終是零，它不儲存資料，它只是連線兩個過程，像 shell 提供的”`1|2`”語法功能一樣。然而，一旦管道有了名稱，這兩個程序就可以不必在同一個指令列，甚至由同一個使用者執行。管道是 UNIX 的一個非常有影響力的創新。

嘗試下列例子

```
$ cd; mkfifo mypipe
$ echo ”hello” >mypipe & # put into background
[1] 8022
$ ls -l mypipe
prw-rw-r-- 1 penguin penguin 0 Oct 16 21:49 mypipe
$ cat mypipe
hello
[1]+  Done                    echo ”hello” >mypipe
$ ls mypipe
mypipe
$ rm mypipe
```

1.2.9 套接字

套接字被廣泛應用於所有的網際網路通訊，資料庫和作業系統本身。它類似於命名管道（FIFO）並且允許程序之間甚至不同計算機之間進行資訊交換。對於套接字，這些程序不需要在同一時間執行，也不需要是同一個父程序的子程序。它是程序間通訊（IPC）的一個節點。資訊的交換可能會通過網路發生在不同主機之間。最常見的兩種是 網際網路套接字 和 UNIX 域套接字 。

提示
通過”`netstat -an`” 指令可以很方便的檢視系統已經打開了那些套接字。

1.2.10 設備文件

設備文件包括系統的物理設備和虛擬設備，如硬盤、顯卡、顯示屏、鍵盤。虛擬設備的一個例子是控制檯，用“/dev/console”來描述。

設備文件有兩種類型。

- 字元設備
  - 每次存取一個字元
  - 一個字元等於一個字節
  - 如鍵盤、串口…
- 塊設備
  - 通過更大的單元-塊，進行存取
  - 一個塊 > 一個字節
  - 如硬盤等…

你可以讀寫塊設備文件，儘管該文件可能包含二進位數據，讀取後顯示出無法理解的亂碼。向文件寫入數據，有時可以幫助定位硬體連接故障。比如，你可以將文本文件導入打字機設備“/dev/lp0”，或者將調製解調指令發送到合適的串口“/dev/ttyS0”。但是，除非這些操作都小心完成，否則可能會導致一場大災難。所以要特別小心。

注
常規存取打字機，使用 lp(1)。

設備的節點數可以通過執行 ls(1) 得到，如下所示。

```
$ ls -l /dev/sda /dev/sr0 /dev/ttyS0 /dev/zero
brw-rw---T  1 root disk     8,  0 Oct 16 20:57 /dev/sda
brw-rw---T+ 1 root cdrom   11,  0 Oct 16 21:53 /dev/sr0
crw-rw---T  1 root dialout  4, 64 Oct 16 20:57 /dev/ttyS0
crw-rw-rw-  1 root root     1,  5 Oct 16 20:57 /dev/zero
```

- ”/dev/sda” 的主設備號是 8，次設備號是 0。它可以被 disk 羣組的使用者讀寫。
- ”/dev/sr0” 的主設備號是 11，次設備號是 0。它可以被 cdrom 羣組的使用者讀寫。
- ”/dev/ttyS0” 的主設備號是 4，次設備號是 64。它可以被 dailout 羣組的使用者讀寫。
- ”/dev/zero” 的主設備號是 1，次設備號是 5。它可以被任意使用者讀寫。

在現代 Linux 系統中，處在”/dev” 之下的檔案系統會自動被 udev() 機制填充。

1.2.11 特別設備文件

還有一些特別的設備文件。

這些特別設備文件經常和 shell 數據重導向聯合使用（參考節 1.5.8）。

設備文件	操作	響應描述
/dev/null	讀取	回傳“文件結尾字元（EOF）“
/dev/null	寫入	無回傳（一個無底的數據轉存深淵）
/dev/zero	讀取	回傳”\0 空字元”（與 ASCII 中的數字 0 不同）
/dev/random	讀取	從真隨機數產生器回傳隨機字元，提供真熵（緩慢）
/dev/urandom	讀取	從能夠安全加密的僞隨機數產生器回傳隨機字元
/dev/full	寫入	回傳磁盤已滿（ENOSPC）錯誤

Table 1.10: 特別設備文件列表

1.2.12 procfs 和 sysfs

procfs和sysfs兩個僞檔案系統，分別掛載於”/proc” 和”/sys” 之上，將核心中的數據結構暴露給使用者空間。或者說，這些條目是虛擬的，他們打開了深入瞭解作業系統運行的方便之門。

目錄”/proc” 爲每個正在運行的行程提供了一個子目錄，目錄的名字就是行程的 PID。需要讀取行程資訊的系統工具，如 ps()，可以從這個目錄結構獲得資訊。

”/proc/sys” 之下的目錄，包含了可以更改某些核心運行參數的接口。（你也可以使用專門的 sysctl() 指令修改，或者使用其預加載/調配文件”/etc/sysctl.conf”。）

當人們看到這個特別大的文件”/proc/kcore” 時，常常會驚慌失措。這個文件於你的的電腦記憶體大小相差不多。它被用來調試核心。它是一個虛擬文件，指向系統記憶體，所以不必擔心它的大小。

”/sys” 以下的目錄包含了核心輸出的數據結構，它們的屬性，以及它們之間的鏈接。它同時也包含了改變某些核心運行時參數的界面。

參考”proc.txt(.gz)”,”sysfs.txt(.gz)”,以及其他相關的 Linux 核心文檔（”/usr/share/doc/linux-doc-*/Doc
這些文件由 linux-doc-* 軟體包提供。

1.2.13 tmpfs

tmpfs是一個臨時檔案系統，它的文件都保存在虛擬記憶體中。必要時，位於記憶體頁快取的 tmpfs 數據可能被交換到硬碟中的交換分區。

系統啓動早期階段，”/run” 目錄掛載爲 tmpfs。這樣即使”/” 掛載爲只讀，它也是可以被寫入的。它爲過渡態文件提供了新的存儲空間，同時也替代了Filesystem Hierarchy Standar2.3 版中說明的目錄位置：

- ”/var/run” → ”/run”
- ”/var/lock” → ”/run/lock”
- ”/dev/shm” → ”/run/shm”

參考”tmpfs.txt(.gz)”, 文件位於 Linux 核心文檔（”/usr/share/doc/linux-doc-*/Documentation/filesystem
目錄之下，由軟體包 linux-doc-* 提供。

1.3 Midnight Commander (MC)

Midnight Commander (MC) 是一個 Linux 終端或其它終端環境下的 GNU 版” 瑞士刀” 。它為新手們提供了一個選單式樣的終端使用體驗，這更易於學習運用標準的 Unix 指令。

你可能需要按照下面的指令來安裝標題為” mc ” 的 Midnight Commander 包.

```
$ sudo apt-get install mc
```

使用 mc(1) 指令那個來瀏覽 Debian 系統。這是最好的學習方式。請使用游標鍵和 Enter 鍵來翻看一些感興趣的內容。

- ”/etc” 及其子目錄
- ” /var/log ” 及其子目錄
- ” /usr/share/doc ” 及其子目錄
- ” /usr/sbin ” 和” /usr/bin ”

1.3.1 自定義 MC

為了在退出 MC 的時候更改目錄並 cd 到其它目錄，我建議修改”~/.bashrc” 包含一個由 mc 包提供的指令碼。

```
. /usr/lib/mc/mc.sh
```

檢視 mc(1) (在”-P” 選項裡) 的原因。(如果你不能理解我這裡說所講的，你可以稍後回頭再看)

1.3.2 啟動 MC

MC 可以這樣啟動起來。

```
$ mc
```

MC 通過選單覆蓋了所有的檔案操作，因此而讓使用者更省心省力。只需要按 F1 就可以跳轉到幫助介面。你只需要按游標鍵和功能鍵就可以使用 MC。

注
某些終端比如 gnome-terminal(1)，功能鍵的按鍵觸發訊息可能會被終端程式擷取。在 gnome-terminal 裡，可以透過” 首選項” → “通用” -> ” 快捷鍵” 選單設定來停用這些特徵。

如果你遇到字元編碼問題，顯示出來都是亂碼，通過新增”-a” 到 MC 指令列或許有助於避免此類問題。

如果這樣不能解決 MC 中的顯示問題，可以參考節 9.5.6.

1.3.3 MC 文件管理

預設的兩個目錄面板裡包含了檔案列表。另一個有用的模式是設定右邊視窗為” 資訊” 來讀取檔案存取許可權資訊。接下來是一些必要的快捷鍵。背景程式 gpm(8) 執行的時候，你也可以在字元指令列裡用滑鼠來操作。(在 MC 裡進行復制和貼上操作的時候一定要按住 shift 鍵。)

快捷鍵	鍵綁定功能
F1	幫助清單
F3	內部檔案檢視器
F4	內部編輯器
F9	啟用下拉選單
F10	退出 Midnight Commander
Tab	在兩個視窗間移動
Insert 或 Ctrl-T	用於多檔案操作的標記檔案，如副本
Del	刪除檔案 (注意---設定 MC 為安全刪除模式)
游標鍵	自我解釋

Table 1.11: MC 快捷鍵綁定

1.3.4 MC 指令列技巧

- cd 指令在選中的螢幕中改變目錄。
- Ctrl-Enter or Alt-Enter 拷貝檔名到指令列。使用 cp(1) 和 mv(1) 兩個指令來進行處理。
- Alt-Tab 顯示檔名自動補全提示。
- 通過新增 MC 指令參數可以指定開始目錄；例如，”mc /etc /root”。
- Esc + n-key → Fn (i.e., Esc + 1 → F1, etc.; Esc + 0 → F10)
- 先按 Esc 鍵和同時按 Alt 是一樣；例如, 輸入 Esc + c 和同時 Alt-C 是一樣的。Esc 被稱為 meta 鍵，有時候也稱之為”M-”。

1.3.5 MC 內部編輯器

這個內建編輯器有一個有意思的貼上方案。摁 F3 開始選擇起始點，再摁 F3 選擇終點並高亮選擇區。此刻你可以移動你的游標，使用 F6 將選區移動到當前游標下，F5 則將選區複製到當前游標下。 F2 儲存檔案。 F10 退出。多數游標鍵以直觀的方式工作。

MC 編輯器可以直接以下面的指令方式啟動。

```
$ mc -e filename_to_edit
```

```
$ mcedit filename_to_edit
```

這不是一個多視窗編輯器，但是能通過複用終端來達到同樣的效果。在兩個視窗間複製，需要用到 Alt-F < n > 來切換虛擬終端並使用”File→Insert file” 或者”File→Copy to file” 來移動文字。

內部編輯器可以被外部編輯器替代。

同樣，許多程式使用環境變數 $EDITOR 或 $VISUAL 來決定編輯器的使用。如果你準備使用 vim(1) 或者 nano(1) 來開始，你或許需要將下面的程式碼加入”~/.bashrc” 來對 mcedit 進行設定。

```
export EDITOR=mcedit
export VISUAL=mcedit
```

如果可能的話我推薦用”vim”。

如果你使用 vim(1) 並不順手，你可以在大部分系統中繼續使用 mcedit(1) 來進行工作。

1.3.6 MC 內部檢視器

MC 是一個非常智慧的檢視器。這是一個在文件中搜索文字的好工具。我經常使用它在/usr/share/doc 目錄中查詢檔案。這是瀏覽大量 Linux 資訊的最快方式。這個檢視器可以通過下列指令中的任何一個來直接啟動。

```
$ mc -v path/to/filename_to_view
```

```
$ mcview path/to/filename_to_view
```

1.3.7 自動啟動 MC

在檔案中輸入回車，用適當的程式來處理檔案的內容 (檢視節 9.4.11)。這是 MC 一個非常方便的用法。

為讓這些檢視器和虛擬檔案特徵生效，可檢視的檔案不能夠被設定為可執行。使用 chmod(1) 或通過 MC 檔案選單改變他們的狀態。

檔案型別	對 **Enter** 鍵的響應
可執行檔案	執行指令
幫助文件	管道內容檢視器軟體
html 檔案	管道內容網頁瀏覽器
"`*.tar.gz`" 和"`*.deb`" 文件	瀏覽其內容就像檢視子目錄一樣

Table 1.12: MC 中對 Enter 鍵的響應

1.3.8 MC 中的虛擬檔案系統

MC 能夠跨因特網訪問檔案。在選單按 `F9`, "`Enter`" 和"`h`" 來啟用 Shell 檔案系統。按"`sh://[user@]machine[:options]/[`的形式輸入 URL, 就會看起來像本地使用 `ssh` 一樣來檢索遠端目錄。

1.4 類 Unix 工作環境基礎

雖然 MC 差不多可以讓你做任何事情，但學會從 shell 提示下使用指令列工具也是非常重要的，可以讓你變得熟悉類 Unix 工作環境。

1.4.1 登入 shell

因登入 shell 可以被一些系統初始化程式使用，請謹慎的把登入 shell 保持為 `bash`(1)，並避免把它轉換為 `chsh`(1)。

如果你想使用不同的 shell 互動提示符，從 GUI（圖形使用者介面）的終端模擬器來設定；或者從 `~/.bashrc` 啟動，比如說，在裡面放置"`exec /usr/bin/zsh -i -l`" 或"`exec /usr/bin/fish -i -l`"。

軟體包	流行度	大小	**POSIX shell**	說明
`bash`	V:837, I:999	7175	Yes	Bash: GNU Bourne Again SHell (事實上的標準)
`bash-completion`	V:32, I:932	1454	N/A	bash shell 程式設計補全
`dash`	V:883, I:997	191	Yes	Debian Almquist Shell, 擅長 shell 指令碼
`zsh`	V:40, I:74	2463	Yes	Z shell: 有許多增強的標準 shell
`tcsh`	V:6, I:21	1355	No	TENEX C Shell: 一個 Berkeley csh 的增強版本
`mksh`	V:7, I:12	1566	Yes	Korn shell 的一個版本
`csh`	V:1, I:6	339	No	OpenBSD C Shell, Berkeley csh 的一個版本
`sash`	V:0, I:5	1157	Yes	有內建指令的 Stand-alone shell (並不意味著標準的"`/usr/bin/sh`")
`ksh`	V:1, I:11	61	Yes	Korn shell的真正的 AT&T 版本
`rc`	V:0, I:1	178	No	AT&T Plan 9 rc shell 的一個實現
`posh`	V:0, I:0	190	Yes	Policy-compliant Ordinary SHell 策略相容的普通 shell(`pdksh` 衍生實現)

Table 1.13: shell 程式列表

提示
雖然類 POSIX 共享基本語法，但他們在 shell 變數和 glob 擴充等基本事情上，行為可以不同。細節請查閱他們的文件。

在本教學中，互動式的 shell 總是指 `bash`.

1.4.2 定製 bash

您可客製化 bash(1) 行為，使用”~/.bashrc”。

嘗試下列例子。

```
# enable bash-completion
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

# CD upon exiting MC
. /usr/lib/mc/mc.sh

# set CDPATH to a good one
CDPATH=.:/usr/share/doc:~:~/Desktop:~
export CDPATH

PATH="${PATH+$PATH:}/usr/sbin:/sbin"
# set PATH so it includes user's private bin if it exists
if [ -d ~/bin ] ; then
  PATH="~/bin${PATH+:$PATH}"
fi
export PATH

EDITOR=vim
export EDITOR
```

提示
可以找到其他 bash 客製化技巧，諸如節 9.3.6，在章 9之內。

提示
bash-completion 軟體包能夠讓 bash 進行指令補全.

1.4.3 特殊按鍵

在類 Unix環境，有一些具有特殊含義的按鍵。請注意，普通的 Linux 字元控制台，只有左手邊的 Ctrl 和 Alt 鍵可以正常運作。其中有幾個值得記住的按鍵。

提示
Ctrl-S 的終端功能可能被 stty(1) 禁用。

1.4.4 滑鼠操作

Debian 系統針對文字的滑鼠操作混合 2 種風格，外加一些新的方法：

- 傳統的 Unix 滑鼠操作方式：
  - 使用 3 個按鈕（單擊）

快捷鍵	描述 **key binding**
Ctrl-U	刪除光標前到行首的字元
Ctrl-H	刪除光標前的一個字元
Ctrl-D	終止輸入 (如果你在使用 shell，則退出 shell)
Ctrl-C	終止一個正在執行的程式
Ctrl-Z	通過將程序移動到後臺來暫停程序
Ctrl-S	停止螢幕輸出
Ctrl-Q	重啟螢幕輸出
Ctrl-Alt-Del	重啓/關閉系統，參見 inittab(5)
Left-Alt-key (或 Windows-key)	Emacs 和相似 UI 的元鍵 (meta-key)
Up-arrow	開始在 bash 中的指令歷史搜索
Ctrl-R	開始在 bash 中的增量指令歷史搜索
Tab	在 bash 命令列中補全文件名
Ctrl-V Tab	在 bash 命令列中輸出 Tab 而不是進行補全

Table 1.14: bash 的按鍵綁定列表

- 使用主要鍵
- 使用圖形化應用程式像是 xterm 和 text 應用程式於 Linux console

• 現代 GUI (圖形使用者介面) 滑鼠操作方式：

- 拖曳並點擊
- 使用 PRIMARY 和剪貼簿
- 使用於現代 GUI 應用程式像是 gnome-terminal

操作	回應
左擊並拖動滑鼠游標	主要鍵的選擇作為選擇範圍
單擊左鍵	主要鍵的位置作為選擇範圍的開頭
單擊右鍵 (傳統方式)	主要鍵的位置作為選擇範圍的結尾
單擊右鍵 (現代方式)	前後文相依清單 (cut/copy/paste)
點選中鍵或者 Shift-Ins	在游標處插入主要鍵的選擇
Ctrl-X	剪下主要鍵的選擇到剪貼簿
Ctrl-C (在終端是 Shift-Ctrl-C)	複製主要鍵的選擇到剪貼簿
Ctrl-V	貼上剪下板的內容到游標處

Table 1.15: 游標右鍵操作及相關按鍵操作列表於 Debian

於此，在 terminal 中主要選取反白文字的區間，使用 Shift-Ctrl-C 去複製，以避免中止正在執行的程式

在現代滾輪滑鼠上的中央滾輪，被認為是中間鍵，並可以被當做中間鍵使用。在 2 鍵滑鼠系統的情況下，同時按左鍵和右鍵就相當於按中間鍵。

為了在 Linux 字元控制台中使用游標，你必須讓 gpm(8) 在背景執行。

1.4.5 文件內容查看

less(1) 命令是一個增強版的分頁程式 (檔案內容檢視器)。它按照指定的命令引數或標準輸出來讀取檔案。在用 less 命令檢視的時候如果需要幫助可以按 “h”。它的功能比 more(1) 命令更豐富，透過在指令碼的開頭執行”eval $(lesspipe)” 或”eval $(lessfile)” 它的功能還能變得更加強大。詳細請參考”/usr/share/doc/less/LESSOPEN”。”-R” 選項可以實現原始的字元輸出還可以啟用 ANSI 顏色轉義序列。詳細請參考 less(1)。

提示
在 less 命令中，輸入”h” 來檢視幫助螢幕，輸入”/” 或”?” 來搜尋字串，輸入”-i” 來改變大小寫敏感性。

1.4.6 文字編輯器

在使用類 Unix 系統過程中，各種類似於Vim 或 Emacs的工具，你應該精通其中的一個。

我認為習慣於使用 Vim 指令是一個明智的選擇，因為 Linux/Unix 系統裡一般都附帶了 Vi 編輯器。(實際上最初的 `vi` 以及後來的 `nvi` 這類工具程式很常見。因為在 Vim 裡提供了 `F1` 幫助鍵，在同類工具中它的功能更強大，所以我選擇 Vim 而不是其它新出的一些工具。)

假設你不是用 Emacs 就是用XEmacs 作為你的編輯器，其實還有更好的選擇，尤其是在程式設計的時候。Emacs 還有很多其他的特點，包括新手導讀，目錄編輯器，郵件客戶端等等。當編寫指令碼或程式的時候，它能自動識別當前工作模式所對應的格式，讓使用更加便利。一些人甚至堅持認為 Linux 系統裡最需要配備的就是 Emacs。花十分鐘來學習 Emacs 可以為後面的工作剩下更多時間。在此強烈推薦學習使用 Emacs 時候直接使用 GNU Emacs 參考手冊。

在實踐應用中所有這些程式都會有一個教學，輸入"`vim`" 和 F1 鍵就可以啟動 Vim。建議你最好閱讀一下前面的 35 行。移動游標到"`|tutor|`" 並按 `Ctrl-]` 就可以看到線上教學。

注
好的編輯器，像 Vim 和 Emacs，可以處理 UTF-8 及其它不常用編碼格式的文字。有個建議就是在 GUI (圖形使用者介面) 環境下使用 UTF-8 編碼，並安裝要求的程式和字型。編輯器裡可以選擇獨立於 GUI (圖形使用者介面) 環境的編碼格式。關於多位元組文字可以查閱參考文件。

1.4.7 設置預設文本編輯器

Debian 有許多不同的編輯器。我們建議安裝上面提到的 `vim` 軟體包。

Debian 通過指令“`/usr/bin/editor`”提供了對系統預設編輯器的統一存取，因此其它程序 (例如 `reportbug`(1)) 可以調用它。你可以通過下列指令改變它。

```
$ sudo update-alternatives --config editor
```

對於新手，我建議使用“`/usr/bin/vim.basic`”代替“`/usr/bin/vim.tiny`”，因爲它支援語法突顯 (syntax highlighting)。

提示
許多程式使用環境變數“`$EDITOR`”或“`$VISUAL`”來決定使用那個編輯器 (參見節 1.3.5和節 9.4.11)。出於 Debian 系統的一致性考慮，它們被設置到“`/usr/bin/editor`”。(在歷史上，“`$EDITOR`”是“`ed`”，“`$VISUAL`”是“`vi`”。)

1.4.8 使用 vim

最近的 `vim`(1) 用完全的"`nocompatible`" 選項啟動自己，進入到 普通模式。[2]

請使用"`vimtutor`" 程式來學習 `vim`，透過一個互動式的指導課程。

`vim` 程式基於 模式輸入的按鍵來改變它的行為。在 插入-模式和 替代-模式下，輸入的按鍵大部分進入了緩衝區。移動游標大部分在 普通-模式下完成。互動選擇在 可視-模式下完成。在普通-模式下輸入"`:`" ，改變它的 模式進入到 `Ex`-模式。`Ex`-接受命令。

提示
Vim 和 **Netrw** 軟體包可以一起使用。Netrw 同時支援在本地和網路讀寫檔案，瀏覽目錄。用"`vim .`" (一個點作為引數) 來嘗試 Netrw，在"`:help netrw`" 讀取它的文件。

`vim` 的高階配置，參見節 9.2。

2即使舊的 `vim` 也能夠啟動完全的"`nocompatible`" 模式，透過使用"`-N`" 選項啟動。

模式	按鍵	操作
普通	`:help|only`	顯示 help file
普通	`:e filename.ext`	開啟新的緩衝區來編輯 `filename.ext`
普通	`:w`	由當前緩衝區覆寫原有檔案
普通	`:w filename.ext`	寫入當前緩衝區到 `filename.ext`
普通	`:q`	退出 `vim`
普通	`:q!`	強制退出 `vim`
普通	`:only`	關閉所有其它分割開啟的視窗
普通	`:set nocompatible?`	檢查 `vim` 是否在完全的 `nocompatible` 模式
普通	`:set nocompatible`	設定 `vim` 到完全的 `nocompatible` 模式
普通	`i`	進入 插入模式
普通	`R`	進入 替代模式
普通	`v`	進入 可視模式
普通	`V`	進入 可視行模式
普通	Ctrl-V	進入 可視塊模式
除了 `TERMINAL-JOB` 外	`ESC`-鍵	進入 普通模式
普通	`:term`	進入 `TERMINAL-JOB` 模式
`TERMINAL-NORMAL`	`i`	進入 `TERMINAL-JOB` 模式
`TERMINAL-JOB`	Ctrl-`W N` (或者 Ctrl-\ Ctrl-`N`)	進入 `TERMINAL-NORMAL` 模式
`TERMINAL-JOB`	Ctrl-`W :`	在 `TERMINAL-NORMAL` 模式裡進入 `Ex`-模式

Table 1.16: 基本的 Vim 按鍵列表

1.4.9 記錄 shell 活動

shell 指令的輸出有可能滾動出了螢幕，並可能導致你無法再查看到它。將 shell 活動記錄到文件中再來回顧它是個不錯的主意。當你執行任何系統管理任務時，這種記錄是必不可少的。

> 提示
> 新版本的 Vim (version>=8.2) 能夠被用來清晰的記錄 shell 活動，使用 `TERMINAL-JOB`-模式。參見節 1.4.8。

記錄 shell 活動的基本方法是在 `script`(1) 下運行 shell。

嘗試下列例子

```
$ script
Script started, file is typescript
```

在 `script` 下使用任何 shell 指令。

按 `Ctrl-D` 來退出 `script`。

```
$ vim typescript
```

參見節 9.1.1。

1.4.10 基本的 Unix 指令

讓我們來學習基本的 Unix 指令。在這裏，我指的是一般意義上的“UNIX”。任何 UNIX 複製所衍生的系統通常都會提供等價的指令。Debian 系統也不例外。如果有一些指令不像你想的那樣起作用，請不要擔心。如果 shell 中使用了別名，其對應的指令輸出會不同。這些例子並不意味着要以這個順序來執行。

建議嘗試使用非特權使用者帳號來使用下列的指令。

指令	說明
pwd	顯示當前/工作目錄的名稱
whoami	顯示當前的使用者名
id	顯示當前使用者的身份（名稱、uid、gid 和相關組）
file *foo*	顯示“*foo*”文件的文件類型
type -p *commandname*	顯示“*commandname*”指令的文件所處位置
which *commandname*	同上
type *commandname*	顯示“*commandname*”指令的相關資訊
apropos *key-word*	查找與“*key-word*”有關的指令
man -k *key-word*	同上
whatis *commandname*	用一行解釋“*commandname*”指令
man -a *commandname*	顯示“*commandname*”指令的解釋（Unix 風格）
info *commandname*	顯示“*commandname*”指令相當長的解釋（GNU 風格）
ls	顯示目錄內容（不包含以. 點號開頭的文件和目錄）
ls -a	顯示目錄內容（包含所有文件和目錄）
ls -A	顯示目錄內容（包含幾乎所有文件和目錄，除了“..”和“.”）
ls -la	顯示所有的目錄內容，并包含詳細的資訊
ls -lai	顯示所有的目錄內容，并包含 inode 和詳細的資訊
ls -d	顯示當前目錄下的所有目錄
tree	使用樹狀圖顯示目錄內容
lsof *foo*	列出處於開啟狀態的檔案”*foo*”
lsof -p *pid*	列出被某程序開啟的檔案: ”*pid*”
mkdir *foo*	在當前目錄中建立新目錄“*foo*”
rmdir *foo*	刪除當前目錄中的“*foo*”目錄
cd *foo*	切換到當前目錄下或變量“$CDPATH”中的“*foo*”目錄
cd /	切換到根目錄
cd	切換到當前使用者的家目錄
cd */foo*	切換到絕對路徑爲“*/foo*”的目錄
cd ..	切換到上一級目錄
cd *~foo*	切換到使用者“*foo*”的家目錄
cd -	切換到之前的目錄
</etc/motd pager	使用預設的分頁顯示程式來顯示“/etc/motd”的內容
touch *junkfile*	建立一個空文件“*junkfile*”
cp *foo bar*	將一個現有文件“*foo*”複製到一個新文件“*bar*”
rm *junkfile*	刪除文件“*junkfile*”
mv *foo bar*	將一個現有文件“*foo*”重命名成“*bar*”（“*bar*”必須不存在）
mv *foo bar*	將一個現有文件“*foo*”移動到新的位置“*bar/foo*”（必須存在“*bar*”目錄）
mv *foo bar/baz*	移動一個現有文件“*foo*”到新位置並重命名爲“*bar/baz*”（必須存在“bar”目錄，且不存在“bar>/*baz*”目錄）
chmod 600 *foo*	使其他人無法讀寫現有文件“*foo*”（並且所有人都無法執行該文件）
chmod 644 *foo*	使其他人對現有文件“*foo*”可讀但不可寫（並且所有人都無法執行該文件）
chmod 755 *foo*	使其他人對“*foo*”可讀而不可寫（並且所有人都能執行該文件）
find . -name *pattern*	使用 shell “*pattern*”查找匹配的文件名（速度較慢）
locate -d . *pattern*	使用 shell “*pattern*”查找匹配的文件名（速度較快，使用定期生成的數據庫）
grep -e ”*pattern*” *.html	在當前目錄下以“.html”結尾的所有文件中，查找匹配“*pattern*”的文件並顯示
top	全螢幕顯示行程資訊，輸入“q”退出
ps aux | pager	顯示所有正在運行的行程的資訊（BSD 風格）
ps -ef | pager	顯示所有正在運行的行程的資訊（Unix system-V 風格）
ps aux | grep -e ”[e]xim4*”	顯示所有正在運行“exim”和“exim4”的行程
ps axf | pager	顯示所有正在運行的行程的資訊（ASCII 風格）
kill *1234*	傳送程式中止訊號給 ID 爲“*1234*”的行程
gzip *foo*	使用 Lempel-Ziv 編碼（LZ77）將“*foo*”壓縮爲“*foo*.gz”
gunzip *foo*.gz	將“*foo*.gz”解壓爲“*foo*”
bzip2 *foo*	使用 Burrows-Wheeter 塊排序壓縮算法和 Huffman 編碼將“*foo*”壓縮爲“*foo*.bz2”（壓縮效果比 gzip 更好）
bunzip2 *foo*.bz2	將“*foo*.bz2”解壓爲“*foo*”
xz *foo*	使用 Lempel-Ziv-Markov 鏈算法將“*foo*”壓縮爲“*foo*.xz”（壓縮效果比 bzip2 更好）

注
Unix 有一個慣例，以“.”開頭的文件將被隱藏。它們一般爲包含了調配資訊和使用者首選項的文件。
對於 cd 指令，參見 builtins(7)。
基本的 Debian 系統的預設分頁顯示程式是 more(1)，它無法往回滾動。通過指令“apt-get install less”安裝 less 軟體包後，less(1) 會成爲預設的分頁顯示程式，它可以通過方向鍵往回滾動。
"[" 和"]" 在正規表達式"ps aux | grep -e "[e]xim4*"" 指令中，可以避免 grep 在結果中排除它自己，正規表達式中的"4*" 意思是空或字元"4"，這樣可以讓 grep 既找到"exim" 也找到"exim4"。雖然"*" 可以用於指令名稱匹配和正規表達式中，但是它們的含義是不一樣的。欲詳細瞭解正規表達式可以參考 grep(1)。

作爲訓練，請使用上述的指令來遍歷目錄並探究系統。如果你有任何有關控制臺指令的問題，請務必自行閱讀手冊。

嘗試下列例子

```
$ man man
$ man bash
$ man builtins
$ man grep
$ man ls
```

手冊的風格可能讓人有點難以習慣，因爲它們都相當簡潔，尤其是比較老舊、非常傳統的那些手冊。但是，一旦你習慣了它，你會欣賞它們的簡潔。

請注意，許多類 Unix 指令（包含來自 GNU 和 BSD 的）都可以顯示簡短的幫助資訊，你可以使用下列的其中一種方式來查看它（有時不帶任何參數也可以）。

```
$ commandname --help
$ commandname -h
```

1.5 簡單 shell 指令

現在，你對如何使用 Debian 系統已經有一些感覺了。讓我們更深入瞭解 Debian 系統的指令執行機制。在這裏，我將爲新手做一般的講解。精確的解釋參見 bash(1)。

一般的指令由有序的組件構成。

1. 初始化此程式之環境變數值（可選）
2. 指令名
3. 參數（可選）
4. 重導向（可選：>, >>, <, << 等等）
5. 控制運算子（可選：&&，||，換行符號, ;, &, (,)）

1.5.1 指令執行和環境變數

一些環境變數的值會改變部分 Unix 指令的行爲。

環境變數的預設值由 PAM 系統初始化，其中一些會被某些應用程式重新設定。

- PAM（可插拔身份驗證模組）系統的模組,比如 pam_env 模組,可以透過 /etc/pam.conf”、”/etc/environment” 和”/etc/default/locale” 設定環境變數。
- 顯示管理器（例如 gdm3）可以透過”~/.profile” 給 GUI（圖形使用者介面）會話重新設定環境變數。
- 使用者特有的程式初始化時，可以重新設定在”~/.profile”、”~/.bash_profile” 和”~/.bashrc” 中設定的環境變數。

1.5.2 “$LANG”變量

預設的語言環境是在”$LANG” 環境變數中定義，它在安裝的時候配置為”LANG=xx_YY.UTF-8”，或者在接下來的 GUI (圖形使用者介面) 中配置，例如在 GNOME 中是，” 設定” → ” 區域 & 語言” → ” 語言” / ” 格式”。

注
目前建議最好用變數”$LANG” 來配置系統環境變數，只有在逼不得已的情況下才用 $LC_*” 開頭的變數。

“$LANG”變數的完整的語言環境值由 3 部分組成：“xx_YY.ZZZZ”。

語言環境值	說明
xx	ISO 639 語言代碼 (小寫) 例如“en”
YY	ISO 3166 國家代碼 (大寫) 例如“US”
ZZZZ	編碼，總是設置爲“UTF-8”

Table 1.18: 語言環境值的 3 個部分

locale 推薦	語言 (地區)
en_US.UTF-8	英語 (美國)
en_GB.UTF-8	英語 (大不列顛)
fr_FR.UTF-8	法語 (法國)
de_DE.UTF-8	德語 (德國)
it_IT.UTF-8	意大利語 (意大利)
es_ES.UTF-8	西班牙語 (西班牙)
ca_ES.UTF-8	加泰隆語 (西班牙)
sv_SE.UTF-8	瑞典語 (瑞典)
pt_BR.UTF-8	葡萄牙語 (巴西)
ru_RU.UTF-8	俄語 (俄國)
zh_CN.UTF-8	漢語 (中華人民共和國)
zh_TW.UTF-8	漢語 (中國臺灣)
ja_JP.UTF-8	日語 (日本)
ko_KR.UTF-8	韓語 (韓國)
vi_VN.UTF-8	越南語 (越南)

Table 1.19: locale 推薦的列表

使用 shell 命令列按順序執行下列典型的指令。

```
$ echo $LANG
en_US.UTF-8
$ date -u
Wed 19 May 2021 03:18:43 PM UTC
$ LANG=fr_FR.UTF-8 date -u
mer. 19 mai 2021 15:19:02 UTC
```

這裡，date(1) 程式執行時使用了不同的語言環境值。

- 第一個指令，“$LANG”設定為系統的預設語言環境值“en_US.UTF-8”。
- 第二個指令，“$LANG”設置爲法語的 UTF-8 locale 值“fr_FR.UTF-8”。

大多數的指令在執行時並沒有預先定義環境變數。對於上面的例子，你也可以選擇如下的方式。

```
$ LANG=fr_FR.UTF-8
$ date -u
mer. 19 mai 2021 15:19:24 UTC
```

提示
提交一個 BUG 報告的時候，如果使用的是非英語的環境，在"LANG=en_US.UTF-8" 語言環境環境下對命令進行執行和檢查會更好一些。

對於語言環境調配的細節，參見節 8.1。

1.5.3 "$PATH" 變數

當你在 Shell 裡輸入指令的時候，Shell 會在"$PATH" 變數所包含的目錄列表裡進行搜尋，"$PATH" 變數的值也叫作 Shell 的搜尋路徑。

在預設的 Debian 安裝過程中，所使用的使用者帳號的"$PATH" 環境變數可能不包括"/usr/sbin" 和"/usr/sbin" 目錄。例如，ifconfig 指令就需要指定完整的路徑"/usr/sbin/ifconfig"。(類似地，ip 指令是在"/usr/bin" 目錄下)

可以在 Bash 指令碼檔案"~/.bash_profile" 或"~/.bashrc" 中改變"$PATH" 環境變數的值。

1.5.4 "$HOME" 變數

很多指令在使用者目錄中都存放了使用者指定的調配，然後通過調配的內容來改變它的執行方式，使用者目錄通常用"$HOME" 變數來指定。

"$HOME" 變數的值	程式執行環境
/	初始程式執行的程式 (背景程式 daemon)
/root	root 使用者許可權 Shell 執行的程式
/home/*normal_user*	普通使用者許可權 Shell 執行的程式
/home/*normal_user*	普通使用者 GUI 桌面選單執行的程式
/home/*normal_user*	用 root 使用者許可權來執行程式"sudo program"
/root	用 root 使用者許可權執行程式"sudo -H program"

Table 1.20: "$HOME" 變數值列表

提示
Shell 擴充"~/" 為轉入當前使用者的主目錄，也就是"$HOME/"。Shell 擴充"~foo/" 為 foo 的目錄，也就是"/home/foo/"。

如果 $HOME 對你的程式不可用，參見節 12.1.5。

1.5.5 指令列選項

一些指令附帶參數。這些參數以"-" 或"--" 開頭，通常稱之為選項，用來控制指令的執行方式。

```
$ date
Thu 20 May 2021 01:08:08 AM JST
$ date -R
Thu, 20 May 2021 01:08:12 +0900
```

這裡的指令參數"-R" 改變 date(1) 指令輸出為 RFC2822 標準的日期字元格式。

1.5.6 Shell 萬用字元

經常有這種情況你期望指令成串自動執行而不需要挨個輸入，將檔名擴展為 **glob**，(有時候被稱為 萬用字元)，以此來滿足這方面的需求。

shell glob 模式	匹配規則之描述
*	不以”.” 開頭的檔名 (段)
.*	以”.” 開頭的檔名 (段)
?	一個字元
[…]	包含在括號中的任意字元都可以作為一個字元
[a-z]	”a” 到”z” 之間的任意一個字元都可以作為一個字元
[^…]	除了包含在括號中的任意字元 (” 1^ 2” 除外)，其它字元都可以作為一個字元

Table 1.21: Shell glob 模式

嘗試下列例子

```
$ mkdir junk; cd junk; touch 1.txt 2.txt 3.c 4.h .5.txt ..6.txt
$ echo *.txt
1.txt 2.txt
$ echo *
1.txt 2.txt 3.c 4.h
$ echo *.[hc]
3.c 4.h
$ echo .*
. .. .5.txt ..6.txt
$ echo .*[^.]*
.5.txt ..6.txt
$ echo [^1-3]*
4.h
$ cd ..; rm -rf junk
```

參見 glob(7)。

注
與 shell 通用的檔名匹配方式不同，使用” -name ” 選項的 find (1)，其 shell 模式” * ”，匹配以” . ” 開始的檔名. (新POSIX 的特性)

注
BASH 可以使用內建的 shopt 選項如” dotglob ”，” noglob ”,” nocaseglob ”,” nullglob ”,” extglob ” 定製全域性行為, 使用 bash (1) 檢視詳細說明。

1.5.7 指令的回傳值

每個指令都會回傳它的退出狀態 (變量:“$?”) 作爲回傳值。

嘗試下列例子。

```
$ [ 1 = 1 ] ; echo $?
0
$ [ 1 = 2 ] ; echo $?
1
```

指令的退出狀態	數字回傳值	邏輯回傳值
success	zero, 0	**TRUE**
error	non-zero, -1	**FALSE**

Table 1.22: 指令的退出代碼

注
請注意，**success** 是邏輯 **TRUE**，0（zero）則是它的值。這有些不直觀，需要在這裡提一下。

1.5.8 典型的順序指令和 shell 重導向

讓我們試著記住下面 Shell 指令裡部分指令列所使用的指令習語。

指令常見用法	說明
`command &`	在子 shell 的後臺 中執行 `command`
`command1 | command2`	通過管道將 `command1` 的標準輸出作爲 `command2` 的標準輸入（並行執行）
`command1 2>&1 | command2`	通過管道將 `command1` 的標準輸出和標準錯誤作爲 `command2` 的標準輸入（並行執行）
`command1 ; command2`	執行 `command1` 與 `command2` 循序地進行
`command1 && command2`	執行 `command1`；若成功，則續執行 `command2` 循序地進行 (若 `command1` 與 `command2` 都成功，則送回成功的訊息)
`command1 || command2`	執行 `command1`；若不成功，接著執行 `command2` 循序地 (若 `command1` 或 `command2` 成功，則送回成功的訊息)
`command > foo`	將 `command` 的標準輸出重導向到文件 `foo`（覆蓋）
`command 2> foo`	將 `command` 的標準錯誤重導向到文件 `foo`（覆蓋）
`command >> foo`	將 `command` 的標準輸出重導向到文件 `foo`（附加）
`command 2>> foo`	將 `command` 的標準錯誤重導向到文件 `foo`（附加）
`command > foo 2>&1`	將 `command` 的標準輸出和標準錯誤重導向到文件 `foo`
`command < foo`	將 `command` 的標準輸入重導向到文件 `foo`
`command << delimiter`	將 `command` 的標準輸入重導向到下面的命令列，直到遇到“`delimiter`”（here document）
`command <<- delimiter`	將 `command` 的標準輸入重導向到下面的命令列，直到遇到“`delimiter`”（here document，命令列中開頭的製表符會被忽略）

Table 1.23: Shell 指令常見用法

Debian 系統是一個多工的作業系統。後臺任務讓使用者能夠在一個 shell 中執行多個程式。後臺行程的管理涉及 shell 的內建指令：`jobs`、`fg`、`bg` 和 `kill`。請閱讀 bash(1) 中的章節：“SIGNALS”、“JOB CONTROL”和“`builtins`(1)”。

嘗試下列例子

```
$ </etc/motd pager
```

```
$ pager </etc/motd
```

```
$ pager /etc/motd
```

```
$ cat /etc/motd | pager
```

儘管 4 個 shell 重導向的例子都會顯示相同的結果，但最後一個例子毫無意義地運行了額外的 `cat` 指令浪費了資源。

shell 允許你使用 `exec` 通過任意一個文件描述符來打開文件。

```
$ echo Hello >foo
$ exec 3<foo 4>bar  # open files
$ cat <&3 >&4       # redirect stdin to 3, stdout to 4
$ exec 3<&- 4>&-    # close files
$ cat bar
Hello
```

預定義（Predefined）的文件描述符 0-2。

設備	說明	文件描述符
stdin	標準輸入	0
stdout	標準輸出	1
stderr	標準錯誤	2

Table 1.24: 預定義的文件描述符 (file descriptors)

1.5.9 指令別名

你可以爲經常使用的指令設置一個別名。

嘗試下列例子

```
$ alias la='ls -la'
```

現在，“la”是“ls -al”的簡寫形式，並同樣會以長列表形式列出所有的文件。

你可以使用 alias 來列出所有的別名（參見 bash(1) 中的“SHELL BUILTIN COMMANDS”）。

```
$ alias
...
alias la='ls -la'
```

你可以使用 type 來確認指令的準確路徑或類型（參見 bash(1) 中的“SHELL BUILTIN COMMANDS”）。

嘗試下列例子

```
$ type ls
ls is hashed (/bin/ls)
$ type la
la is aliased to ls -la
$ type echo
echo is a shell builtin
$ type file
file is /usr/bin/file
```

ls 在最近被使用過，而“file”沒有，因此“ls”標記為“hashed”（被錄入雜湊表），即 shell 有一個內部的記錄用來快速存取“ls”所處的位置。

提示
參見節 9.3.6。

1.6 類 Unix 的文本處理

在類 Unix 的工作環境中，文本處理是通過使用管道組成的標準文本處理工具鏈完成的。這是另一個重要的 Unix 創新(設計哲學)。

1.6.1 Unix 文本工具

這裏有一些在類 Unix 系統中經常使用到的標準文本處理工具。

- 沒有使用正規表達式：
  - `cat`(1) 連接文件並輸出全部的內容。
  - `tac`(1) 連接文件並反向輸出。
  - `cut`(1) 選擇行的一部分並輸出。
  - `head`(1) 輸出文件的開頭。
  - `tail`(1) 輸出文件的末尾。
  - `sort`(1) 對文本文件的行進行排序。
  - `uniq`(1) 從已排序的文件中移除相同的行。
  - `tr`(1) 轉換或刪除字元。
  - `diff`(1) 對文件的行進行對比。
- 預設使用基礎正則表示式（**BRE**）：
  - `ed`(1) 是一個原始行編輯器。
  - `sed`(1) 是一個流編輯器。
  - `grep`(1) 匹配滿足 pattern 的文字。
  - `vim`(1) 是一個螢幕編輯器。
  - `emacs`(1) 是一個螢幕編輯器。（有些擴展的 **BRE**）
- 使用擴展的正規表達式（**ERE**）：
  - `awk`(1) 進行簡單的文本處理。
  - `egrep`(1) 匹配滿足多個 pattern 的文字。
  - `tcl`(3tcl) 可以進行任何你想得到的文本處理：參見 `re_syntax`(3)。經常與 `tk`(3tk) 一起使用。
  - `perl`(1) 可以進行任何你想得到的文本處理。參見 `perlre`(1)。
  - `pcregrep` 軟體包中的 `pcregrep`(1) 可以匹配滿足 Perl 相容正規表達式（PCRE）模式的文字。
  - `python`(1) 帶有 `re` 模組可以處理每個預期的文件作業。見”`/usr/share/doc/python/html/index.html`”。

如果你不確定這些指令究竟做了什麼，請使用“`man command`”來自己把它搞清楚吧。

注
排序的順序和表示式的範圍取決於語言環境。如果你想要獲得一個命令的傳統行為，可以使用“`LANG=C`”或 **C.UTF-8** 語言環境代替原來的 **UTF-8** 語言環境（參見節 8.1）。

注
Perl 正規表達式（`perlre`(1)）、Perl 兼容正則表達式（PCRE）和 Python 的 `re` 模組提供的正規表達式與一般的 **ERE** 相比多了許多通用的擴展。

BRE	ERE	正規表達式的描述
\ . [ ] ^ $ *	\ . [ ] ^ $ *	通用的元字元
\+ \? \(\) \{ \} |		BRE 獨有的“\”跳脫元字元
	+ ? ( ) { } |	ERE 獨有的不需要“\”轉義的元字元
c	c	匹配非元字元 (即文字字元) “c”
\c	\c	匹配一個字面意義上的字元“c”，即使“c”本身是元字元
.	.	匹配任意字元，包括換行符
^	^	字串的開始位置
$	$	字串的結束位置
\<	\<	單詞的開始位置
\>	\>	單詞的結束位置
[abc…]	[abc…]	匹配在“abc...”中的任意字元
[^abc…]	[^abc…]	匹配除了“abc...”中的任意字元
r*	r*	匹配零個或多個“r”
r\+	r+	匹配一個或多個“r”
r\?	r?	匹配零個或一個“r”
r1|r2	r1|r2	匹配一個“r1”或“r2”
\(r1|r2\)	(r1|r2)	匹配一個“r1”或“r2“，並作爲括號內的正規表達式

Table 1.25: BRE 和 ERE 中的元字元

1.6.2 正規表達式

正規表達式被使用在許多文字處理工具中。它們類似 shell 的萬用字元，但更加複雜和強大。

正規表達式描述要匹配的模式，它是由文字字元和元字元 **(metacharacters)** 構成的。

元字元僅僅是帶有特殊含義的字元。它們有兩種主要的形式，**BRE** 和 **ERE** ，使用哪種取決於上述的文本工具。

常見的 **emacs** 表示為基本的 **BRE** 但擴充為”+” 與”?” 做為 **metacharacters** 當成 **ERE**。因此，不需要以”\” 在正規的表示 emacs 表示中做為跳脫字元。

grep(1) 可以使用正規表達式來進行文本搜索。

嘗試下列例子

```
$ egrep 'GNU.*LICENSE|Yoyodyne' /usr/share/common-licenses/GPL
GNU GENERAL PUBLIC LICENSE
GNU GENERAL PUBLIC LICENSE
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
```

提示
參見節 9.3.6。

1.6.3 替換表達式

對於替換表達式，一些字元有特殊的含義。

但對 Perl 替換字串來說，應使用“$&”而非“&”，應使用“$n”而非“\n”。

嘗試下列例子

```
$ echo zzz1abc2efg3hij4 | \
sed -e 's/\(1[a-z]*\)[0-9]*\(.*\)$/=&=/'
zzz=1abc2efg3hij4=
```

替換表達式	替換表達式替換的文本
&	正規表達式所匹配的內容（在 emacs 中使用 \& ）
\n	前 n 個括號的正規表達式匹配的內容（“n”是數字）

Table 1.26: 替換表達式

```
$ echo zzz1abc2efg3hij4 | \
sed -E -e 's/(1[a-z]*)[0-9]*(.*)$/=&=/'
zzz=1abc2efg3hij4=
$ echo zzz1abc2efg3hij4 | \
perl -pe 's/(1[a-z]*)[0-9]*(.*)$/=$&=/'
zzz=1abc2efg3hij4=
$ echo zzz1abc2efg3hij4 | \
sed -e 's/\(1[a-z]*\)[0-9]*\(.*\)$/\2===\1/'
zzzefg3hij4===1abc
$ echo zzz1abc2efg3hij4 | \
sed -E -e 's/(1[a-z]*)[0-9]*(.*)$/\2===\1/'
zzzefg3hij4===1abc
$ echo zzz1abc2efg3hij4 | \
perl -pe 's/(1[a-z]*)[0-9]*(.*)$/$2===$1/'
zzzefg3hij4===1abc
```

請特別注意這些括號正規表達式的格式，以及這些被匹配的文本如何被替換到不同的文本處理工具中的。

這些正規表達式在一些編輯器中也可以用來移動游標和替換文字。

在 shell 指令列行末的反斜槓“\”會跳脫一個換行字元（作為空白字元），並將游標移動到下一行的行首。

請閱讀所有相關手冊來學習這些指令。

1.6.4 正規表達式的全域性替換

ed(1) 指令可以在“file”中將所有的“FROM_REGEX”替換成“TO_TEXT”。

```
$ ed file <<EOF
,s/FROM_REGEX/TO_TEXT/g
w
q
EOF
```

sed(1) 指令可以在“file”中將所有的“FROM_REGEX”替換成“TO_TEXT”。

```
$ sed -i -e 's/FROM_REGEX/TO_TEXT/g' file
```

vim(1) 指令可以通過使用 ex(1) 指令在“file”中將所有的“FROM_REGEX”替換成“TO_TEXT”。

```
$ vim '+%s/FROM_REGEX/TO_TEXT/gc' '+update' '+q' file
```

提示
上面的“c”標誌可以確保在每次替換時都進行互動式的確認。

多個檔案（“file1”，“file2”和“file3”）可以使用 vim(1) 或 perl(1) 通過正規表達式進行類似的處理。

```
$ vim '+argdo %s/FROM_REGEX/TO_TEXT/gce|update' '+q' file1 file2 file3
```

提示
上面的“e”標誌是為了防止“No match”錯誤中斷替換。

```
$ perl -i -p -e 's/FROM_REGEX/TO_TEXT/g;' file1 file2 file3
```

在 perl(1) 例子中, "-i" 是在每一個目標檔案的原處編輯，"-p" 是表示迴圈所有給定的檔案。

提示
使用參數“-i.bak”代替“-i”，可以在檔名後新增“.bak”再儲存。對於複雜的替換，這使得從錯誤中恢復變得容易。

注
ed(1) 和 vim(1) 使用 **BRE**； perl(1) 使用 **ERE**。

1.6.5 從文字檔案的表格中提取資料

下面有一個文字檔案“DPL”，裡面含有 2004 年以前 Debian 專案的領導者名字和起始日期，並以空格分隔。

```
Ian     Murdock   August  1993
Bruce   Perens    April   1996
Ian     Jackson   January 1998
Wichert Akkerman  January 1999
Ben     Collins   April   2001
Bdale   Garbee    April   2002
Martin  Michlmayr March   2003
```

提示
參見 “Debian 簡史” 獲得最新的 Debian 領導階層歷史。

Awk 經常被用來從這種型別的檔案中提取資料。

嘗試下列例子

```
$ awk '{ print $3 }' <DPL                    # month started
August
April
January
January
April
April
March
$ awk '($1=="Ian") { print }' <DPL           # DPL called Ian
Ian     Murdock   August  1993
Ian     Jackson   January 1998
$ awk '($2=="Perens") { print $3,$4 }' <DPL # When Perens started
April 1996
```

Shell（例如 Bash）也可以用來分析這種檔案。

嘗試下列例子

```
$ while read first last month year; do
    echo $month
  done <DPL
... same output as the first Awk example
```

內建指令 read 使用“$IFS”(內部域分隔符) 中的字元來將行分隔成多個單詞。

如果你將“$IFS”改變為“:”，你可以很好地使用 shell 來分析“/etc/passwd”。

```
$ oldIFS=”$IFS”   # save old value
$ IFS=’:’
$ while read user password uid gid rest_of_line; do
    if [ ”$user” = ”bozo” ]; then
      echo ”$user’s ID is $uid”
    fi
  done < /etc/passwd
bozo’s ID is 1000
$ IFS=”$oldIFS”   # restore old value
```

(如果要用 Awk 做到相同的事，使用“FS=’:’”來設定域分隔字元。)

IFS 也被 shell 用來分割參數擴展、指令替換和算術擴充套件的結果。這不會出現在雙引號或單引號中。IFS 的預設值為 空格、*tab* 和換行字元。

請謹慎使用 shell 的 IFS 技巧。當 shell 將指令碼的一部分解釋為對它的輸入時，會發生一些奇怪的事。

```
$ IFS=”:,”                        # use ”:” and ”,” as IFS
$ echo IFS=$IFS,   IFS=”$IFS”     # echo is a Bash builtin
IFS=  , IFS=:,
$ date -R                         # just a command output
Sat, 23 Aug 2003 08:30:15 +0200
$ echo $(date -R)                 # sub shell --> input to main shell
Sat  23 Aug 2003 08 30 36 +0200
$ unset IFS                       # reset IFS to the default
$ echo $(date -R)
Sat, 23 Aug 2003 08:30:50 +0200
```

1.6.6 用於管道指令的小片段指令碼

下面的指令碼作為管道的一部分，可以做一些細緻的事情。

使用 find(1) 和 xargs(1)，單行 shell 指令碼能夠在多個檔案上迴圈使用，可以執行相當複雜的任務。參見節 10.1.5 和節 9.4.9.

當使用 shell 互動模式變得太麻煩的時候，請考慮寫一個 shell 腳本 (參見節 12.1).

指令碼片段（在一行內輸入）	指令效果
`find /usr -print`	找出"/usr" 下的所有檔案
`seq 1 100`	顯示 1 到 100
`| xargs -n 1 command`	把從管道過來的每一項作為參數，重複執行指令
`| xargs -n 1 echo`	把從管道過來的，用空格隔開的項，分隔成多列
`| xargs echo`	把從管道過來的所有列合併為一列
`| grep -e regex_pattern`	從管道過來，包含有 *regex_pattern* 的列，提取出來
`| grep -v -e regex_pattern`	把從管道過來，不包含有 *regex_pattern* 的列，提取出來
`| cut -d: -f3 -`	把從管道過來，用":" 分隔的第三行提取出來 (passwd 檔案等。)
`| awk '{ print $3 }'`	把用空格隔開的第三行提取出來
`| awk -F'\t' '{ print $3 }'`	把用 tab 鍵隔開的第三行提取出來
`| col -bx`	刪除退格鍵，擴展 tab 鍵為空格鍵
`| expand -`	擴展 tab 鍵為空格鍵
`| sort| uniq`	排序並刪除重複列
`| tr 'A-Z' 'a-z'`	將大小字母轉換為小寫字母
`| tr -d '\n'`	將多列連線為一列（移除換行字元）
`| tr -d '\r'`	刪除換行 CR 字元
`| sed 's/^/# /'`	在每列行首增加一個"#" 符
`| sed 's/\.ext//g'`	刪除".ext"
`| sed -n -e 2p`	顯示第二列
`| head -n 2 -`	顯示最前面兩列
`| tail -n 2 -`	顯示最後兩列

Table 1.27: 管道指令的小片段指令碼列表

Chapter 2

Debian 軟體包管理

注
這一章假定最新的穩定版的代號為: bookworm 。
在本文件中，APT 系統的資料來源總稱為源列表。能夠在"/etc/apt/sources.list" 檔案、"/etc/apt/sources.list.d/*.list" 檔案或"/etc/apt/sources.list.d/*.source" 檔案的任何地方定義。

2.1 Debian 軟體包管理的前提

2.1.1 Debian 軟體包管理

Debian 是一個志願者組織，它建立一致的自由軟體的預編譯二進位制包並從檔案庫中分發它們。

許多遠端映象站提供了 HTTP 和 FTP 的方式來存取 Debian 檔案庫。也可以使用 CD-ROM/DVD 。

目前 Debian 的軟體包管理系統是 高階軟體包工具 (APT)，它能夠使用所有這些資源。

Debian 軟體包管理系統，當使用適當時，可以讓使用者從檔案庫安裝統一設定的二進位制軟體包到系統中。現在，有 71664 個可用於 amd64 架構的軟體包。

Debian 軟體包管理系統有豐富的歷史，有許多可供選擇的前端使用者程式和後端存取方式。現在，我們推薦下列的這些。

- apt(8) 用於所有的互動式命令列操作，包含軟體包的安裝、移除和版本升級。
- apt-get(8) 用於從指令碼中呼叫 Debian 軟體包管理系統。它在 apt 不可用時也可作為一個備選選項 (常見於較舊的 Debian 系統)。
- aptitude(8) 使用一個互動式的文字介面來管理已安裝的軟體包和搜尋可用的軟體包。

2.1.2 軟體包調配

下面是 Debian 系統軟體包調配的一些要點。

- Debian 尊重系統管理員的手動調配。換句話說，軟體包調配系統不會為了方便而去更改那些調配。
- 每個軟體包都帶有自己的調配指令碼，它使用標準使用者介面 debconf(7) 來幫助軟體包初始化安裝過程。
- Debian 開發者通過軟體包調配指令碼，盡力使你能有一個完美的升級體驗。

軟體包	流行度	大小	說明
dpkg	V:912, I:999	6387	用於 Debian 的底層軟體包管理系統（基於檔案的）
apt	V:866, I:999	4313	使用命令列管理軟體包的 APT 前端: apt/apt-get/apt-cache
aptitude	V:48, I:260	4377	使用全屏控制檯互動式管理軟體包的 APT 前端: aptitude(8)
tasksel	V:35, I:980	347	用來安裝選擇的任務的 APT 前端: tasksel(8)
unattended-upgrades	V:184, I:287	301	用於 APT 的增強軟體包，會自動安裝安全更新
gnome-software	V:150, I:259	3041	GNOME 軟體中心 (圖形化的 APT 前端)
synaptic	V:45, I:372	7627	圖形化的軟體包管理工具（GTK 的 APT 前端）
apt-utils	V:372, I:998	1063	APT 實用程式: apt-extracttemplates(1)、apt-ftparchive(1) 和 apt-sortpkgs(1)
apt-listchanges	V:350, I:870	398	軟體包歷史更改提醒工具
apt-listbugs	V:6, I:9	477	在每次 APT 安裝前列出嚴重的 bug
apt-file	V:17, I:69	89	APT 軟體包搜尋工具——指令列介面
apt-rdepends	V:0, I:5	39	遞迴列出軟體包依賴

Table 2.1: Debian 軟體包管理工具列表

- 系統管理員可以使用軟體包工具的全部功能。但在預設的安裝中會禁用那些具有安全風險的。
- 如果你手動激活了一些具有安全隱患的服務，你有責任遏制風險。
- 高深的配置可以由系統管理員手動啟用。這可能會對用於系統配置的通用流行幫助程式造成干擾。

2.1.3 基本的注意事項

警告
不要從任何的混合套件中安裝軟體包。它可能會打破軟體包的一致性，這需要你要深厚的系統管理知識，例如 ABI 編譯器、庫版本和直譯器特性等等。

Debian 系統管理員中的新手應該保持在只進行安全更新的 **stable** 版本。直到你十分了解 Debian 系統前，你應當遵循下列的預防措施。

- 在 原始檔中不要包含 **testing** 或 **unstable**。
- 在 原始檔裡不要在標準的 Debian 中混合使用其它非 Debian 的檔案庫，例如 Ubuntu。
- 不要建立“/etc/apt/preferences”。
- 不瞭解會造成的全部影響，就不要通過組態檔案改變軟體包管理工具的預設行為。
- 不要使用“dpkg -i *random_package*”安裝任何軟體包。
- 絕不使用“dpkg --force-all -i *random_package*”安裝任何軟體包。
- 不要刪除或修改“/var/lib/dpkg/”中的檔案。
- 不要讓從原始碼直接安裝的程式覆蓋系統檔案。
  - 如果需要的話，將它們安裝到“/usr/local”或“/opt”中。

對 Debian 軟體包管理系統，違背上面的預防措施，會導致不相容影響，可能會使你的系統無法使用。

負責有關鍵任務的伺服器的嚴謹的 Debian 系統系統管理員，應該使用額外的預防措施。

- 沒有在安全的條件下使用你特定的調配進行徹底地測試，就不要從 Debian 安裝任何軟體包（包含安全更新）。
  - 你作為系統管理員要對你的系統負責到底。
  - Debian 系統長久的穩定史並無法保證什麼。

2.1.4 持續升級的生活

注意
對於你的生產伺服器，建議使用帶有安全更行的 `stable` 套件。對於你只進行有限管理的桌面 PC 也是同樣如此。

儘管我在上面進行了警告，我知道本文件的許多讀者希望可以執行更新的 `testing` 或 `unstable` 版。

菩薩使用下面的內容拯救一個人，使他從掙扎於持續升級地獄的因果報應中脫困，並讓他達到 Debian 的極樂世界。

這個列表面向 自己管理的桌面環境。

- 使用 `testing` 版，實際上，它是自動滾動釋出的，由 Debian 檔案庫的 QA 質量架構來管理，比如：Debian 持續整合、只上傳原始碼實踐 和 庫轉換跟蹤。在 `testing` 版中的軟體包被更新得足夠頻繁來提供全部最新的特性。
- 在源列表裡面設定 `testing` 版相應的程式碼名為套件名（在 `bookworm`-作為-`stable` 版的釋出週期時,是"`trixie`"）。
- 大概在主版本釋出一個月後，僅僅在你自己評估了形勢後，才手動更新 原始檔裡的這個程式碼名到新的版本號。對於這個更新，Debian 使用者和開發者郵件列表也是好的資訊來源。

使用 `unstable` 版是不推薦的。`unstable` 版對開發者 除錯軟體包合適，但對普通的桌面使用而言，會有使你暴露在不必要的風險中的傾向。儘管 Debian 系統的 `unstable` 版在大多數時候看起來都非常穩定，但會有一些軟體包問題，並且它們中的一部分是不容易解決的。

這裡有一些基本預防措施意見，確保簡單快速地從 Debian 軟體包的 bug 中恢復。

- 通過將 Debian 系統的 `stable` 套件安裝到另一個分割槽，可以使系統能夠進行雙啟動
- 製作安裝 CD 便於用於 救援啟動
- 考慮安裝 `apt-listbugs`，這可以在升級之前檢查 Debian Bug 跟蹤系統（BTS）的資訊
- 對軟體包系統的基礎設施有足夠的瞭解來解決問題

注意
如果你無法做到這些預防措施中的任何一個，那你可能還沒做好使用 `testing` 和 `unstable` 版的準備。

2.1.5 Debian 檔案庫基礎

提示
Debian 檔案庫官方政策的定義參見 Debian 政策文件，第 2 章——Debian 檔案庫。

讓我們從系統使用者的角度來看看 Debian 檔案庫。

對於系統使用者，是使用 APT 系統來訪問 Debian 檔案庫。

APT 系統定義它的資料來源作為源列表，在 `sources.list`(5) 裡面描述。

對於使用典型的 HTTP 訪問的 `bookworm` 系統，單行格式的源列表如下：

```
deb http://deb.debian.org/debian/ bookworm main non-free-firmware contrib non-free
deb-src http://deb.debian.org/debian/ bookworm main non-free-firmware contrib non-free

deb http://security.debian.org/debian-security bookworm-security main non-free-firmware  ↩
    contrib non-free
deb-src http://security.debian.org/debian-security bookworm-security main non-free-firmware ↩
     contrib non-free
```

可替代的，相等的使用 deb822 格式的源列表如下：

```
Types: deb deb-src
URIs: http://deb.debian.org/debian/
Suites: bookworm
Components: main non-free-firmware contrib non-free

Types: deb deb-src
URIs: http://security.debian.org/debian-security/
Suites: bookworm-security
Components: main non-free-firmware contrib non-free
```

原始檔的關鍵點如下。

- 單行格式
  - 它的定義檔案在”/etc/apt/sources.list” 檔案和”/etc/apt/sources.list.d/*.list” 檔案裡面。
  - 每一行定義了 APT 系統的資料來源。
  - “deb” 的那行定義了二進位制軟體包。
  - “deb-src” 的那行定義了原始碼軟體包。
  - 第一個參數是 Debian 檔案庫的根 URL 。
  - 第二個引數是發行版名稱，可以使用套件名或代號。
  - 第三個和之後的參數是 Debian 檔案庫的有效檔案庫範圍名稱。
- Deb822 格式
  - 它的定義檔案在”/etc/apt/sources.list.d/*.source” 檔案裡。
  - 由空格隔開的每個多行塊，定義了 APT 系統的資料來源。
  - ”Types:” 章節定義列表型別，即”deb” 和”deb-src”。
  - ”URIs:” 章節定義 Debian 檔案庫 URI 的根地址。
  - ”Suites:” 章節定義了發行版名稱列表，名稱可以使用套件名或代號。
  - ”Components:” 章節定義 Debian 檔案庫中有效檔案庫名稱列表。

如果只是用 aptitude，它不訪問原始碼相關的元資料，“deb-src” 定義可以安全地省略。這可以加速檔案庫元資料的更新。

URL 可以是”https://”, ”http://”, ”ftp://”, ”file://”,……

”#” 開頭的行是註釋，被忽略。

這裡，我傾向於使用代號 “bookworm” 或”trixie” 來代替套件名 “stable” 或”testing” ，以避免下一個 stable 版本釋出時出現意外。

提示
如果在上述的例子中，使用了 “sid” 代替 “bookworm” ，那麼源列表中用於安全更新的 “deb: http://security.debian.org/ …” 這行或它的 deb822 等價內容就不需要了。因為沒有用於 “sid”（unstable）的安全更新的檔案庫。

檔案庫 URL	套件名	程式碼名	倉庫用途
http://deb.debian.org/debian/	stable	bookworm	在擴充套件檢查後，類似靜態的 stable 釋出
http://deb.debian.org/debian/	testing	trixie	在表面檢查和短期等待後的動態 testing 釋出
http://deb.debian.org/debian/	unstable	sid	在最少的檢查和不等待的動態 unstable 釋出
http://deb.debian.org/debian/	experimental	N/A	開發者預釋出實驗版本（可選，只適用於開發者）
http://deb.debian.org/debian/	stable-proposed-updates	bookworm-proposed-updates	用於下一個穩定版 stable 的點版本（小版本）釋出的更新（可選）
http://deb.debian.org/debian/	stable-updates	bookworm-updates	stable-proposed-updates 套件的子集，需要緊急更新，比如說時區（可選的）
http://deb.debian.org/debian/	stable-backports	bookworm-backports	大部分從 testing 版中隨機收集和重新編譯的軟體包（可選）
http://security.debian.org/debian-security/	stable-security	bookworm-security	釋出的安全更新（重要）
http://security.debian.org/debian-security/	testing-security	trixie-security	這個沒有積極支援，不被安全團隊使用

Table 2.2: Debian 檔案庫站點列表

在 bookworm 釋出後，下面是配置檔案所使用的 Debian 檔案庫站點的 URL 和套件名或代號的列表。

注意
只有帶有安全更新的純淨的 **stable** release 版本可以提供最佳的穩定性。執行大多數 **stable** release 版本的軟體包之中混合一些來自 **testing** 或 **unstable** release 版本的軟體包會比執行純淨的 **unstable** release 版本冒更大的風險，這是因為庫版本的不匹配導致的。如果在 **stable** release 版本下你真的需要一些程式的最新版本，請使用來自stable-updates和 backports （參見節 2.7.4 ）的軟體包。使用這些軟體包時必須額外小心。

注意
在"deb" 行中，你只需列出 stable，testing 或者 unstable 套件中的一個即可，如果你在"deb" 行中混合了 stable，testing 和 unstable 套件，APT 程式的執行速度將會變慢並且只有最新的檔案庫是有用的。只有在"/etc/apt/preferences" 檔案帶有明確目標的時候，混合的列表才是有意義的。（檢視節 2.7.7 ）。

提示
對於使用 stable 套件的 Debian 系統而言，在 源列表中包含帶有“http://security.debian.org/”的內容是不錯的主意。它會啟用安全更新。

注
Debian 安全團體將會修正 stable 檔案庫的安全缺陷。這些行為是十分嚴格可靠的。testing 檔案庫中的缺陷，不一定會被 Debian 測試安全團體修正。由於一些原因，這些行為相對 stable 檔案庫沒有那麼嚴格，您可能需要等待已修正的 unstable 軟體包移植到 testing。unstable 檔案庫的缺陷，交由各個維護者修改。經常維護的 unstable 軟體包通常處於相當好的狀況，因為它利用了上流最新的安全修正。有關 Debian 怎樣處理安全缺陷，請參見 Debian 安全常問問題 。

上述軟體包的數量是 amd64 架構的。main 區域提供 Debian 系統（參見節 2.1.6 ）。

區域	軟體包數量	軟體包元件標準
`main`	70312	遵從 Debian 自由軟體引導方針（DFSG），並且不依賴於 `non-free`
`non-free-firmware`	39	不符合 Debian 自由軟體指導方針（DFSG），正常的系統安裝過程中必需要用到的韌體
`contrib`	360	遵從 Debian 自由軟體引導方針（DFSG），但依賴於 `non-free`
`non-free`	953	不遵從 Debian 自由軟體指導方針（DFSG），並且不在 `non-free-firmware`

Table 2.3: Debian 歸檔列表

通過把你的瀏覽器指向檔案庫 URL，這些 URL 在 `dists` 或 `pool` 之後是各不相同的，Debian 檔案庫能夠被有規劃的組織。

發行版可以用套件或代號來指定。發行版在許多文件中也被當做是套件的同義詞。套件和代號的關係總結如下。

時間	**suite = `stable`**	**suite = `testing`**	**suite = `unstable`**
在 `bookworm` 發佈後	codename = `bookworm`	codename = `trixie`	codename = `sid`
在 `trixie` 發佈後	codename = `trixie`	codename = `forky`	codename = `sid`

Table 2.4: 套件和代號的關係

代號的歷史參見 Debian FAQ: 6.2.1 Which other codenames have been used in the past?

在較嚴格的 Debian 檔案術語，“部分 section”這一詞特指按應用領域來分類的軟體包類別。（但是，主要部分 (”main section”) 這一詞有時會用來描述 Debian 檔案區中，名為”main 主要” 的區域。）

Debian 開發者（DD）每次上傳軟體包到 `unstable` 檔案庫（通過 incoming 處理），都必須確保上傳的軟體包與最新的 `unstable` 檔案庫中的最新軟體包相容。

如果 DD 故意打破重要的庫升級等的這種相容性，這通常會在 Debian 開發者郵件列表等進行公告。

在 Debian 檔案庫維護指令碼將軟體包從 `unstable` 檔案庫移動到 `testing` 檔案庫前，檔案庫維護指令碼不僅檢查時間（約 2-10 天）和軟體包的 RC bug 報告的狀態，還嘗試確保它們可以和最新的 `testing` 檔案庫中的軟體相容。這個過程使得 `testing` 檔案庫非常正確可用。

通過由釋出團隊領導的逐步凍結檔案庫的過程，並進行一些手動干預，使 `testing` 檔案庫完全一致，無缺陷。然後，將舊的 `testing` 檔案庫的程式碼名稱分配給新的 `stable` 檔案庫，併為新的 `testing` 檔案庫建立新的程式碼名稱。新的 `testing` 檔案庫最初的內容和新發布的 `stable` 檔案庫的內容完全相同。

`unstable` 和 `testing` 檔案庫都可能會遭受由以下幾個因素導致的臨時的小故障。

- 損壞的軟體包被上傳到檔案庫（多見於 `unstable` ）
- 延遲接受新的軟體包到檔案庫（多見於 `unstable` ）
- 檔案庫時間同步問題（`testing` 和 `unstable` ）
- 手動干預檔案庫，例如移除軟體包（多見於 `testing` ）等。

因此，如果你決定使用這些檔案庫，你應該能夠修復或忍受這些型別的小故障。

注意
在新的 `stable` 版本釋出後的幾個月，大多數桌面使用者應該使用帶有安全更新的 `stable` 檔案庫，即使他們通常使用 `unstable` 或 `testing` 檔案庫。在這個過渡期中，`unstable` 和 `testing` 檔案庫不適合大多數人。你使用 `unstable` 檔案庫的系統是很難保持良好的工作狀態的，因為它會遭受核心軟體包的大量升級狂潮。`testing` 檔案庫不大有用，因為它包含有和沒有安全支援的 `stable` 檔案庫相同的內容（Debian testing 安全公告 2008-12）。一個月左右的時間後，如果你仔細點的話，`unstable` 或 `testing` 檔案庫或許可以使用。

提示
跟蹤 testing 檔案庫時，由一個已移除的軟體包引起的問題通常可以安裝 unstable 檔案庫中相同的軟體包（已修復 bug）來解決。

檔案庫的定義參見 Debian 政策文件。

- 部分
- "優先順序"
- "基本系統"
- "極重要的軟體包"

2.1.6 Debian 是 100% 的自由軟體

Debian 是 100% 的自由軟體，因為：

- Debian 預設只安裝自由軟體，這尊重了使用者的自由。
- Debian 在 main 中只提供自由軟體。
- Debian 建議只執行來自 main 的自由軟體。
- 在 main 中的軟體包，沒有依賴或推薦在 non-free 或 non-free-firmware 或 contrib 中的軟體包。

有人想知道下列的兩個事實是否互相矛盾。

- "Debian 將始終是 100% 的自由軟體"。（Debian 社群契約中的第一條）
- Debian 伺服器上有一些 non-free-firmware、non-free 和 contrib 軟體包。

因為下列原因，這並不矛盾。

- Debian 系統具有 100% 的自由，並且它的軟體包位於 Debian 伺服器的 main 區域。
- Debian 系統之外的軟體包位於 Debian 伺服器的 non-free、non-free-firmware 和 contrib 區域。

在 Debian 社群契約的第 4 條和第 5 條對這進行了明確的解釋：

- 我們將優先考慮我們的使用者及自由軟體
  - 我們由我們的使用者及自由軟體社群的需要所導向。我們將優先考慮他們的利益。我們將在多種計算環境中支援我們的使用者的操作需要。我們不反對在 Debian 系統上使用非自由軟體，我們也不會嘗試向建立和使用這部分軟體的使用者索取費用。我們允許他人，在沒有我們的資金的參與下，製造包括 Debian 以及商業軟體的增值套件。為了達成這些目標，我們將提供整合的、高質量的、100% 自由的軟體，而不附加任何可能阻止在這些方面使用的法律限制。
- 哪些作品不符合我們的自由軟體規範
  - 我們知道，某些我們的使用者需要使用不符合 Debian 自由軟體指導方針的作品。我們為這些作品，在我們的檔案庫中留出了"non-free"、"non-free-firmware" 和"contrib" 目錄。在這些目錄下的軟體包，並不屬於 Debian 系統儘管它們已被配置成可以在 Debian 下使用。我們鼓勵光碟製造商閱讀這些目錄下的軟體的許可證，以判斷他們是否可以在光碟中發行這些軟體。所以，儘管非自由軟體並非 Debian 系統的一部分，我們仍支援它們的使用，並且我們為非自由軟體提供了公共資源 (諸如我們的缺陷跟蹤系統以及郵件列表)。Debian 官方媒介可以包括韌體，韌體不是 Debian 系統的一部分，這是一個例外，能夠讓 Debian 用於需要這些韌體的硬體上。

注
在目前的 Debian 社群契約（Debian Social Contract） 1.2 版本第 5 條條款的實際文字和上面的文字有稍微不同。在不改變 Debian 社群契約實際內容下，這個文字調整讓本使用者文件在邏輯上保持一致。

使用者應該瞭解使用 non-free 、non-free-firmware 和 contrib 中的軟體包所需要冒的風險：

- 使用類似的軟體包會失去自由
- 失去 Debian 對軟體包的支援（這些軟體包無法存取原始碼，Debian 不能進行完全的支援。）
- 汙染你 100% 自由的 Debian 系統

Debian 自由軟體引導方針為 Debian 設立了自由軟體標準。Debian 對軟體包中的軟體做了最廣泛的解釋，包含文件、韌體、圖示和圖形資料。這使得 Debian 的自由軟體標準非常嚴格。

典型的 non-free、non-free-firmware 和 contrib 軟體包包含了下列型別的自由分發的軟體包：

- 在GNU Free Documentation License下的文件包，包含不變的部分，比如 GCC 和 Make 的。（大多數都可以在 non-free/doc 找到.)
- 包含沒有原始碼的二進位制資料的韌體軟體包，例如在節 9.10.5 中作為 non-free-firmware 列出的軟體包。（多見於 non-free-firmware/kernel 部分。）
- 遊戲和字型軟體包，對商業使用和/或內容修改進行了限制。

請注意，non-free、non-free-firmware 和 contrib 軟體包的數量少於 main 軟體包的 2%。允許訪問 non-free、non-free-firmware 和 contrib 並不會模糊軟體包的來源。使用 aptitude(8) 的全屏互動式介面可以提供完全的可見性和完全的控制，可以讓你決定安裝來自某個部分的軟體包，來使你的系統保持自由。

2.1.7 軟體包依賴關係

Debian 系統通過其控制檔案欄位中的版本化二進位制依賴宣告機制來提供一致的二進位制軟體包集合。下面有一些它們的簡單定義。

- “依賴”
  - 絕對的依賴，所有在這裡列出的軟體包都必須同時或提前安裝。
- ” 預依賴”
  - 類似於 Depends，但列出的軟體包必須提前完成安裝。
- ” 推薦”
  - 這裡表示一個強，但不是絕對的依賴關係。大多數使用者不會想要這個包，除非在這裡列出的所有包都已經安裝。
- ” 建議”
  - 較弱的依賴。這個軟體包的大多數使用者可能會從安裝所列的軟體包中受益，但沒有它們也可以有適當的功能。
- ” 增強”
  - 這裡表明一個像建議的弱依賴關係，不裝也沒關係。
- ” 破損”
  - 表明一個軟體包不相容一些版本規範。一般的解決方法就是升級列出的所有軟體包。
- ” 衝突”

  - 這表明了絕對的不相容。為了安裝這個軟體包必須移除所有列出的軟體包。
- ”替代”
  - 這表明這個檔案安裝的檔案會替代所列的軟體包的檔案。
- ”提供”
  - 表明這個軟體包會提供所列的軟體包所有的檔案和功能。

注
請注意，同時將“Provides”、“Conflicts”和“Replaces”定義到一個虛擬的軟體包是一個明智的調配。這確保了在任何一個時間只能安裝一個提供該虛擬包的真正軟體包。

包含原始碼依賴關係的官方定義位於 the Policy Manual: Chapter 7 - Declaring relationships between packages。

2.1.8 包管理的事件流

這是 APT 提供的軟體包管理的簡單事件流摘要。

- 更新 (“`apt update`”、“`aptitude update`”或“`apt-get update`”):
  1. 從遠端檔案庫獲得檔案庫元資料
  2. 重建和更新 APT 使用的本地元資料
- 升級 (“`apt upgrade`”和“`apt full-upgrade`”,或“`aptitude safe-upgrade`”和“`aptitude full-upgrade`”, 或“`apt-get upgrade`”和“`apt-get dist-upgrade`”):
  1. 選擇候選版本，它所安裝的軟體包通常都是最新的可用版本 (例外參見節 2.7.7)
  2. 解決軟體包依賴關係
  3. 如果候選版本與已安裝的版本不同，會從遠端檔案庫獲得所選擇的二進位制軟體包
  4. 解包所獲得的二進位制軟體包
  5. 執行 **preinst** 指令碼
  6. 安裝二進位制檔案
  7. 執行 **postinst** 指令碼
- 安裝 (“`apt install ...`”、“`aptitude install ...`”或者”`apt-get install …`”):
  1. 選擇命令列中列出的包
  2. 解決軟體包依賴關係
  3. 從遠端伺服器獲得已選二進位制包
  4. 解包所獲得的二進位制軟體包
  5. 執行 **preinst** 指令碼
  6. 安裝二進位制檔案
  7. 執行 **postinst** 指令碼
- 移除 (”`apt remove …`”, “`aptitude remove …`”或“`apt-get remove …`”):
  1. 選擇命令列中列出的包
  2. 解決軟體包依賴關係
  3. 執行 **prerm** 指令碼
  4. 移除已安裝的檔案，除了組態檔案
  5. 執行 **postrm** 指令碼

- 清除（"apt purge"，"aptitude purge …" 或 "apt-get purge …"）：
  1. 選擇命令列中列出的包
  2. 解決軟體包依賴關係
  3. 執行 **prerm** 指令碼
  4. 移除已安裝的檔案，包含組態檔案
  5. 執行 **postrm** 指令碼

這裡，為了大局，我特意省略了技術細節。

2.1.9 對包管理問題的第一個迴應

你應該閱讀優良的官方文件。第一個閱讀的文件是 Debian 特定的"/usr/share/doc/*package_name*/README.Debian"。同時也應該查詢"/usr/share/doc/*package_name*/"中的其它文件。如果你設定 shell 為節 1.4.2，輸入下列指令。

```
$ cd package_name
$ pager README.Debian
$ mc
```

你可能需要安裝以"-doc"字尾命名的對應文件軟體包來獲得詳細的資訊。

如果你在使用一個特定的軟體包時出現了問題，一定要首先檢查 Debian bug 跟蹤系統（BTS）網站。

網站	指令
Debian bug tracking system (BTS) 主頁	sensible-browser "https://bugs.debian.org/"
已經知道軟體包名稱的 bug 報告	sensible-browser "https://bugs.debian.org/*package_name*"
知道 bug 編號的 bug 報告	sensible-browser "https://bugs.debian.org/*bug_number*"

Table 2.5: 解決特定軟體包問題的主要網站

使用 Google 搜尋，在關鍵字中包含"site:debian.org"，"site:wiki.debian.org"，"site:lists.debian.org"等等。

當你要傳送一份 bug 報告時，請使用 reportbug(1) 指令。

2.1.10 如何挑選 Debian 軟體包

當遇到 2 個以上的類似的軟體包時，先前沒有經過反覆的嘗試，你不知道安裝哪一個的時候，應該用常識來判斷。我認為以下幾點是首選的軟體包應該具有的特徵。

- 重要性：是 > 否
- 型別：main > contrib > non-free
- 優先順序：需要 > 重要 > 標準 > 可選 > 額外
- 任務：在任務下有軟體包的列表資訊，例如"桌面環境"
- 軟體包是被與之有依賴關係的軟體包所選擇的（例如 gcc 依賴 gcc-10）
- 流行度：在投票或者安裝指數上有著更高的分數
- 更新日誌：維護者經常的更新
- BTS (缺陷跟蹤系統)：沒有 RC 級別的缺陷（沒有危險、重大嚴重的缺陷）

- BTS (缺陷跟蹤系統): 有維護者對缺陷報告反饋
- BTS (缺陷跟蹤系統): 有著更多的近期修復的 bug 數目
- BTS (缺陷跟蹤系統): 遺留的非嚴重（non-wishlist）缺陷數量較少

Debian 是一個使用分散式開發模式的志願專案，它的檔案庫包含了許多不同關注點和不同質量的軟體包。你必須做出自己的選擇。

2.1.11 怎樣和不一致的要求協作

無論你決定使用哪個 Debian 系統套件，你仍然希望執行在那個套件裡不存在的程式版本。即使你在其它 Debian 套件裡面，或者在其它非 Debian 的資源裡面，找到這個程式的二進位制軟體包，它們的要求可能和你目前的 Debian 系統不一致。

在節 2.7.7 裡描述的 **apt-pinning** 等技術，雖然你能夠用它來調整軟體包管理系統來安裝這類不不同步的二進位制軟體包，但這樣的調整方法只有有限的使用場景，應為它們可能破壞那些程式和你的系統。

在單獨安裝這類不同步的軟體包之前，你需要查詢所有存在的和你目前 Debian 系統相容的安全技術替代方案。

- 使用相應的沙盒化的上游二進位制軟體包來安裝這樣的程式（參見節 7.6）。
  - 許多常見的 GUI（圖形使用者介面）程式，比如 LibreOffice 和 GNOME 應用，會有 Flatpak、Snap 或 AppImage 軟體包存在。
- 建立一個 chroot 或類似的環境來在裡面執行這樣的程式（參見節 9.11）。
  - CLI 命令能夠在和它相容的 chroot 下輕鬆執行（參見節 9.11.4）。
  - 不重啟機器，能夠輕鬆的嘗試多個完整的桌面環境（參見節 9.11.5）。
- 自己構建需要的二進位制軟體包版本，和你目前 Debian 系統相容。
  - 這是一個 不輕鬆的任務 (參見節 2.7.13)。

2.2 基礎軟體包管理操作

在 Debian 系統中有許多基於 APT 的軟體包管理工具可以在 Debian 系統上進行基於倉庫的軟體包管理操作。在這裡，我們將介紹兩種基本的軟體包管理工具：`apt`、`apt-get` / `apt-cache` 和 `aptitude`。

對於涉及軟體包安裝或更新軟體包元資料的軟體包管理操作，你必須有 root 許可權。

2.2.1 `apt` vs. `apt-get` / `apt-cache` vs. `aptitude`

儘管 `aptitude` 是作者主要使用的一個非常好的可互動工具，但你應該知道下列警示：

- 不建議在新版本釋出後在 `stable` Debian 系統上使用 `aptitude` 指令來進行跨版本的系統升級。
  - 建議使用"`apt full-upgrade`" 或“`apt-get dist-upgrade`”來進行這個操作。參見 Bug #411280。
- `aptitude` 指令有時候會為了 `testing` 或 `unstable` Debian 系統升級清除大量軟體包。
  - 這個情況嚇壞了許多的系統管理員。請不要驚慌。
  - 這似乎大多數是由元軟體包的依賴或推薦的軟體包版本偏差造成的，例如 `gnome-core`。
  - 要解決這個問題，可以在 `aptitude` 命令選單中選擇“取消待執行的動作”，退出 `aptitude`，並使用“`apt full-upgrade`”。

`apt-get` 和 `apt-cache` 是最基礎的基於 APT 的軟體包管理工具。

- `apt-get` 和 `apt-cache` 只提供指令列使用者介面。
- `apt-get` 是進行跨版本的主系統升級等操作的最合適工具。
- `apt-get` 提供了一個強大的軟體包依賴解析器。
- `apt-get` 對硬體資源的要求不高。它消耗更少的記憶體並且執行速度更快。
- `apt-cache` 提供了一個 標準的正規表達式來搜尋軟體包名稱和描述。
- `apt-get` 和 `apt-cache` 可以使用 `/etc/apt/preferences` 來管理軟體包的多個版本，但這非常繁瑣。

`apt` 指令是一個高階的命令列介面用於套件管理。基本上，它是 `apt-get`、`apt-cache` 及其相似指令的封裝。本意為一個終端使用者介面，且預設開啟一些適合互動用途的選項。

- `apt` 工具在使用者使用 `apt install` 安裝軟體包時提供了一個友好的進度條。
- 在成功安裝下載的軟體包後，`apt` 將預設刪除快取的 `.deb` 軟體包。

提示
建議使用者使用新的 `apt`(8) 命令用於 互動式的使用場景，而在 shell 指令碼中使用 `apt-get`(8) 和 `apt-cache`(8) 命令。

`aptitude` 指令是最通用的基於 APT 的軟體包管理工具。

- `aptitude` 提供了一個全螢幕的互動式文字使用者介面。
- `aptitude` 同樣也提供了一個指令使用者介面。
- `aptitude` 是用於日常軟體包管理（例如檢查已安裝的軟體包和搜尋可用的軟體包）的最合適工具。
- `aptitude` 對硬體資源的要求更高。它消耗更多的記憶體並且執行速度更慢。
- `aptitude` 提供一個增強的正規表達式來搜尋所有的軟體包元資料。
- `aptitude` 可以管理軟體包的多個版本，並且不使用 `/etc/apt/preferences`，這會十分直觀。

2.2.2 指令列中的基礎軟體包管理操作

下面是使用 `apt`(8), `aptitude`(8) 和 `apt-get`(8) / `apt-cache`(8) 的命令列基本軟體包管理操作。

`apt` / `apt-get` 和 `aptitude` 能夠混用，沒有大問題。

“`aptitude why` *`regex`*”可以透過“`aptitude -v why` *`regex`*”列出更多的資訊。類似的資訊可以透過”`apt rdepends` *`package`*” 或“`apt-cache rdepends` *`package`*”獲取。

當 `aptitude` 指令在指令列模式下啟動後遇到了一些問題（例如軟體包衝突），你可以在之後的提示中按下“e”鍵切換到全螢幕的互動模式。

注
雖然 `aptitude` 命令提供了豐富的功能，例如增強的軟體包解析器，但它的複雜程度導致了（或可能導致）一些退步，例如 Bug #411123、Bug #514930 及 Bug #570377。如有疑問，請使用 `apt`,`apt-get` 和 `apt-cache` 命令來替代 `aptitude` 命令。

你可以在“`aptitude`”後面使用的指令選項。

更多內容參見 `aptitude`(8) 和位於“`/usr/share/doc/aptitude/README`”的“aptitude 使用者手冊”。

apt 語法	**aptitude** 語法	**apt-get** / **apt-cache** 語法	說明
apt update	aptitude update	apt-get update	更新軟體包檔案庫元資料
apt install foo	aptitude install foo	apt-get install foo	安裝“foo”軟體包的候選版本以及它的依賴
apt upgrade	aptitude safe-upgrade	apt-get upgrade	安裝已安裝的軟體包的候選版本並且不移除任何其它的軟體包
apt full-upgrade	aptitude full-upgrade	apt-get dist-upgrade	安裝已安裝的軟體包的候選版本，並且需要的話會移除其它的軟體包
apt remove foo	aptitude remove foo	apt-get remove foo	移除“foo”軟體包，但留下組態檔案
apt autoremove	N/A	apt-get autoremove	移除不再需要的自動安裝的軟體包
apt purge foo	aptitude purge foo	apt-get purge foo	清除“foo”軟體包的組態檔案
apt clean	aptitude clean	apt-get clean	完全清除本地倉庫的軟體包檢索檔案
apt autoclean	aptitude autoclean	apt-get autoclean	清除本地倉庫中過時軟體包的軟體包檢索檔案
apt show foo	aptitude show foo	apt-cache show foo	顯示“foo”軟體包的詳細資訊
apt search 正則表示式	aptitude search *regex*	apt-cache search *regex*	搜尋匹配 *regex* 的軟體包
N/A	aptitude why *regex*	N/A	解釋匹配 *regex* 的軟體包必須被安裝的原因
N/A	aptitude why-not *regex*	N/A	解釋匹配 *regex* 的軟體包不必安裝的原因
apt list --manual-installed	aptitude search '~i!~M'	apt-mark showmanual	列出手動安裝的軟體包

Table 2.6: 使用 apt(8), aptitude(8) 和 apt-get(8) / apt-cache(8) 的命令列基本軟體包管理操作

指令選項	說明
-s	模擬指令的結果
-d	僅下載，不進行安裝/更新
-D	在自動安裝和刪除前，顯示簡要的說明

Table 2.7: aptitude(8) 中重要的指令選項

2.2.3 aptitude 的互動式使用

要使用互動式的軟體包管理，你可以像下面那樣以互動模式啟動 aptitude 。

```
$ sudo aptitude -u
Password:
```

這將更新檔案庫資訊的本地副本,並以選單的形式全螢幕顯示軟體包列表。aptitude 將它的調配放在“~/.aptitude/config”

提示
如果你想用 root 的調配而非使用者的,可以在上面的例子中使用“sudo -H aptitude …”代替“sudo aptitude …”。

提示
當 aptitude 以互動模式啟動時，會自動設定待執行的動作。如果您不喜歡，您可以通過選單:“動作” → “取消待執行的動作”來取消它。

2.2.4 aptitude 的按鍵繫結

在全螢幕模式下瀏覽軟體包狀態和設定動作的按鍵如下。

快捷鍵	鍵綁定功能
F10 or Ctrl-t	選單
?	顯示按鍵幫助（更加完整的清單）
F10 → 幫助 → 使用者手冊	顯示使用者手冊
u	更新軟體包檔案庫資訊
+	標記該軟體包以便升級或安裝
-	標記該軟體包以便移除（保留組態檔案）
_	標記該軟體包以便清除（移除組態檔案）
=	將軟體包設為保持狀態
U	標記所有可升級包（動作如同 **full-upgrade**）
g	開始 下載並 安裝所選擇包
q	退出該介面並儲存變更
x	退出該介面並清除變更
Enter	檢視軟體包的資訊
C	檢視軟體包的變更記錄
l	變更軟體包的顯示限制
/	搜尋匹配的第一個軟體包
\	重複上一個搜尋

Table 2.8: aptitude 的按鍵繫結

可以透過命令列指定檔名稱，也可以透過按“l”或“/”之後在選單提示下輸入下列所述的 aptitude 正則表示式。aptitude 正則表示式可以使用“~n”開頭後接軟體包名稱的字串來精確匹配軟體包名稱。

提示
你需要在視覺化介面中按下“U”鍵讓所有的已安裝軟體包升級到可用版本。否則只有選中的軟體包和一些與之有依賴關係的軟體包才能被升級到可用版本。

2.2.5 aptitude 軟體包檢視

aptitude(8) 全螢幕互動模式下，軟體包列表裡的軟體包會像下面的例子那樣顯示。

```
idA   libsmbclient                              -2220kB 3.0.25a-1  3.0.25a-2
```

該行的從左到右的含義如下。

- “狀態”標籤 (第一個字母)
- “動作”標籤 (第二個字母)
- “自動”標籤 (第三個字母)
- 軟體包名稱
- 該“動作”對磁碟空間的變化
- 軟體包當前版本
- 軟體包可用版本

提示
您可以在幫助選單中找到完整的標籤列表，按“?”即可在幫助選單底部顯示。

可用版本的選擇是依據當前的本地首選項 (參見 apt_preferences(5) 和節 2.7.7)。

軟體包檢視的幾種型別都可以在“檢視 ”選單下找到。

檢視	檢視描述
Package View	參見表格 2.10 (預設)
Audit Recommendations	列出推薦安裝但還沒有安裝的軟體包
Flat Package List	不分類地列出軟體包 (用於正規表達式)
Debtags Browser	列出由 debtags 進行分類的軟體包
原始碼軟體包檢視	列出由原始碼軟體包分組的軟體包

Table 2.9: aptitude 檢視

注
請幫助我們改進用 debtags 標記的軟體包！

標準“軟體包檢視”分類軟體包的方法與帶有一些額外功能的 dselect 有點像。

提示
軟體集檢視可以用來為你的任務選出最佳的軟體包。

2.2.6 aptitude 搜尋方式選項

aptitude 提供了幾個可以使用正規表達式來搜尋軟體包的選項。

- shell 指令列：

分類	檢視描述
Upgradable Packages	按照 section → area → 軟體包的順序顯示列出軟體包
New Packages	同上
Installed Packages	同上
Not Installed Packages	同上
Obsolete and Locally Created Packages	同上
Virtual Packages	列出同樣功能的軟體包
Tasks	列出一個特定任務所需的不同功能的軟體包

Table 2.10: 標準軟體包檢視的分類

  - “aptitude search *'aptitude_regex'*” 列出安裝狀態、軟體包名稱和匹配軟體包的剪短描述
  - “aptitude show *'package_name'*” 列出軟體包的詳細描述
- 全螢幕互動模式:
  - ”l” 可以限制匹配軟體包的檢視
  - ”/” 搜尋匹配的軟體包
  - ”\” 向後搜尋匹配的軟體包
  - ”n” 查詢下一個
  - ”N” 查詢上一個

提示
字串 *package_name* 被看作軟體包名稱的精確字串匹配，除非它是以”~” 開頭的正規表達式。

2.2.7 aptitude 正規表達式

aptitude 常規表達式是類 mutt 的拓展 **ERE**（參見節 1.6.2），aptitude 具體的特殊匹配規則擴充如下。

- 正規表達式使用的是 **ERE**，就跟 egrep(1)、awk(1) 和 perl(1) 這些典型的類 Unix 文字工具中所使用的“^”、“.*”、“$”等是相同的。
- 依賴關係 *type* 是一種特定的軟體包相互關係（depends、predepends、recommends、suggests、conflicts、replaces、provides）。
- 預設的 *type* 依賴關係是“depends”。

提示
當 *regex_pattern* 為空字串時，請立即在指令後面新增”~T”。

下面是一些快捷方式。

- ”~P*term*” == ”~Dprovides:*term*”
- ”~C*term*” == ”~Dconflicts:*term*”
- ”…~W term” == ”(…|term)”

使用者熟悉 mutt 的快速選擇，因為 mutt 的靈感來源於表示式語法。參見“使用者手冊”“/usr/share/doc/aptitude/READ
中的“SEARCHING, LIMITING, AND EXPRESSIONS”。

注
lenny 版本的 aptitude(8) 中，新的長格式語法，例如“?broken”，在正規表達式中可以用來等效為它舊的短格式“~b”。現在空格字元“ ”被認為是除了波浪字元“~”外的另一個正規表達式終止字元。新的長格式語法參見“使用者手冊”。

擴充的比對規則描述	正規表達式
匹配軟體包名稱	~n*regex_name*
匹配描述	~d*regex_description*
匹配軟體集名稱	~t*regex_task*
匹配 debtag	~G*regex_debtag*
匹配維護者	~m*regex_maintainer*
匹配軟體包的 section	~s*regex_section*
匹配軟體包版本	~V*regex_version*
匹配檔案庫	~A{bookworm,trixie,sid}
匹配來源	~O{debian,…}
匹配優先順序	~p{extra,important,optional,required,standard}
匹配必要的軟體包	~E
匹配虛擬軟體包	~v
匹配新的軟體包	~N
匹配待執行的動作	~a{install,upgrade,downgrade,remove,purge,hold,keep}
匹配已安裝軟體包	~i
匹配帶有 **A** 標籤的已安裝軟體包（自動安裝的軟體包）	~M
匹配不帶有 **A** 標籤的已安裝軟體包（管理員選擇的軟體包）	~i!~M
匹配已安裝並且是可升級的軟體包	~U
匹配已刪除但未清除的軟體包	~c
匹配已移除，已清除或可移除的軟體包	~g
匹配破壞依賴關係的軟體包	~b
匹配破壞 *type* 依賴關係的軟體包	~B*type*
匹配 *pattern* 軟體包的 *type* 依賴關係	~D[*type*:]*pattern*
匹配 *pattern* 軟體包破壞的 *type* 依賴關係	~DB[*type*:]*pattern*
匹配依賴於 *pattern* 軟體包的 *type* 依賴的軟體包	~R[*type*:]*pattern*
匹配依賴於 *pattern* 軟體包破壞的 *type* 依賴的軟體包	~RB[*type*:]*pattern*
匹配其它已安裝軟體包所依賴的軟體包	~R~i
匹配沒有被其它已安裝軟體包所依賴的軟體包	!~R~i
匹配其它已安裝軟體包所依賴或建議安裝的軟體包	~R~i|~Rrecommends:~i
匹配 *pattern* 過濾版本之後的軟體包	~S filter *pattern*
匹配所有軟體包（真）	~T
不匹配軟體包（假）	~F

Table 2.11: aptitude 正規表達式

2.2.8 aptitude 的依賴解決

如果通過選單“F10 → 選項 → 首選項 → 正在處理依賴關係”進行相應的設定，則在 aptitude 中選擇一個軟體包時，不僅會將其“Depends:”列表中的軟體包選上，“Recommends:”列表中的軟體包也會被選上。在 aptitude 下，這些自動安裝的軟體包在不再需要時會自動移除。

aptitude 指令中控制“自動安裝”行為的標籤也可以通過 apt 軟體包中的 apt-mark(8) 指令來設定。

2.2.9 軟體包活動日誌

你可以在日誌檔案裡查詢到軟體包活動歷史。

檔案	內容
/var/log/dpkg.log	dpkg 級的軟體包活動日誌
/var/log/apt/term.log	通用 APT 活動日誌
/var/log/aptitude	aptitude 指令活動日誌

Table 2.12: 軟體包活動日誌檔案

事實上，很難從這些日誌上快速獲得有用的資訊。較簡便的方法參見節 9.3.9 。

2.3 aptitude 操作範例

下面是一些 aptitude(8) 的操作範例。

2.3.1 查詢感興趣的軟體包

你可以根據 aptitude 這個包管理工具中的軟體包描述或者是任務面板下的列表資訊，來查詢你所需要的軟體包。

2.3.2 通過正規表達式匹配軟體包名稱來列出軟體包

下面的指令列出了通過正規表達式匹配軟體包名稱來列出軟體包。

```
$ aptitude search '~n(pam|nss).*ldap'
p libnss-ldap - NSS module for using LDAP as a naming service
p libpam-ldap - Pluggable Authentication Module allowing LDAP interfaces
```

這種方式查詢精確的軟體包名稱很方便。

2.3.3 使用正規表達式匹配瀏覽

在” 新扁平軟體包列表” 中使用”l” 提示檢視，正規表達式”~dipv6” 可以限制性地匹配軟體描述，並互動式地展示資訊。

2.3.4 完整地清理已刪除軟體包

您能清除所有已移除軟體包的剩餘組態檔案。

檢查以下指令的結果。

```
# aptitude search '~c'
```

如果您確認所列出的軟體包應當被完整刪除，請執行以下指令。

```
# aptitude purge '~c'
```

您可能想要在互動模式中做類似的操作進行細粒度的控制。

在" 新軟體包檢視" 使用"l" 提示並輸入正則匹配式"~c" ，這將僅匹配軟體包，比如，" 移除但不清空調配" 。所有符合匹配的軟體包可以在頂層標題上使用"[" 顯示。

當您在頂層標題如“未安裝的包”中輸入“_”，當前標題下的軟體包只有匹配正則式才會被清除。您還可以使用“=”來互動式地排除軟體包以避免刪除它們。

這種技術方便易用且適用於許多其他的指令鍵。

2.3.5 調整自動/手動安裝狀態

下面是調整軟體包的自動/手動安裝狀態的方法（在使用非 aptitude 軟體包管理器之後）。

1. 用 root 以互動模式執行 aptitude。
2. 用"u" 指令更新可用的軟體包列表，"U" 指令標記所有可升級的軟體包以執行升級，"f" 指令清除新軟體包列表，"g" 指令執行所有可升級的軟體包以執行升級。
3. 按下“l”，並輸入“~i(~R~i|~Rrecommends:~i)”來限制軟體包的顯示，按下“M”將“已安裝軟體包”的狀態改為自動安裝。
4. 按下“l”，並輸入“~prequired|~pimportant|~pstandard|~E”來限制軟體包的顯示，按下“m”將“已安裝軟體包”的狀態改為手動安裝。
5. 按下“l”，並輸入“~i!~M”來限制軟體包的顯示，在“已安裝軟體包”上按下“[”來陳列無用的軟體包，按下“-”將它們移除。
6. 按下“l”，並輸入“~i”來限制軟體包的顯示，之後在“軟體集”上按下“m”將那些軟體包標記為手動安裝。
7. 退出 aptitude。
8. 用 root 使用者執行"apt-get -s autoremove|less" 指令，來檢視有那些軟體包是不再需要的。
9. 在互動模式下重啟 aptitude 程式，用“m”指令標記所需要的軟體包。
10. 用 root 使用者重新執行"apt-get -s autoremove|less" 這個指令來複查移除的包中是不是隻含有自己所希望移除的軟體包。
11. 用 root 使用者執行"apt-get autoremove|less" 指令來自動移除不再需要的軟體包。

在你所需要執行的"Tasks" 上，執行"m" 指令是一個可選的操作，目的就是為了防止大量軟體包被解除安裝的情況出現。

2.3.6 全面的系統升級

注
當你遷移到新的發行版的時候，雖然正如下面所描述的那樣，Debian 是可升級的，但是你還是應該考慮純淨的安裝新的系統。這給了你機會去移除廢棄的軟體包同時還可以接觸到最新軟體包的完美集合體。當然，在做遷移之前，你也應該對你的系統做完整的備份，並把它移到安全的地方去（檢視節 10.2)。“我”也建議用不同的分割槽做另外一個啟動項，來實現平穩的升級。

你可以透過改變 源列表的內容使之指向新的發行版所在地址的方法來進行系統的全面升級，然後執行”apt update; apt dist-upgrade” 命令。

在 bookworm-作為-stable 釋出迴圈中, 從 stable 升級到 testing 或者 unstable,你應該用”trixie” 或者”sid” 替換源列表檔案裡的”bookworm” 示例，參考節 2.1.5。

事實上，由於一些軟體包版本變遷的問題，你可能會遇到一些困難，主要是由於軟體包的依賴問題。升級之後的差異越大，你越有可能遇到麻煩。在新版本發行後，系統從舊的 stable 過渡到新的 stable，你可以檢視 Release Notes 然後按照裡面的步驟去做，來儘可能的減少麻煩。

在它正式釋出之前，你決定要從先前的 stable 遷移到將要釋出的 testing，這裡沒有 Release Notes 可以幫到你。在前一個 stable 釋出以後， stable 發行版跟將要釋出的 testing 發行版之間的差異可能變得相當大同時也使得升級系統變得更加的複雜。

在全面升級系統的時候，你應該謹慎的操作，同時你也應該從郵件列表中獲得最新的資料然後根據你的常識作出正確的判斷。

1. 檢視先前的“發行說明”。
2. 備份整個系統 (尤其是資料和調配資訊)。
3. 當 bootloader 壞了的時候，手邊應該有可以引導電腦啟動的儲存介質。
4. 事先通知系統上的使用者。
5. 用 script(1) 記錄升級的過程。
6. 用“unmarkauto”指令來保留你想要的軟體包，例如”aptitude unmarkauto vim” 這個指令是用來防止移除 vim 這個軟體的。
7. 為了減少軟體包之間可能會發生的衝突，應該儘量減少要安裝的軟體包的數目，例如，移除桌面環境這個軟體包。
8. 移除”/etc/apt/preferences” 檔案 (停用 **apt-pinning**)。
9. 試著一步步的升級: oldstable → stable → testing → unstable.
10. 升級 源列表檔案，使其指向新的檔案庫然後執行”aptitude update” 命令。
11. 可選的安裝選項，首先是新的 **core packages**，例如“aptitude install perl”。
12. 執行”apt-get -s dist-upgrade” 指令來評估升級造成的影響。
13. 最後執行”apt-get dist-upgrade” 指令。

注意
在 stable 版本升級的時候，跳過主要的 Debian 發行版是不明智的。

注意
在先前的“發行手冊”裡，GCC, Linux Kernel, initrd-tools, Glibc, Perl, APT tool chain 等等，有一些關於系統全面升級的重要注意事項。

關於 unstable 版本的日常升級，檢視節 2.4.3。

指令	操作
COLUMNS=120 dpkg -l *package_name_pattern*	列出已安裝軟體包的列表用於錯誤報告
dpkg -L *package_name*	顯示一個已安裝軟體包的內容
dpkg -L *package_name* | egrep '/usr/share/man/man.*/.+'	列出一個已安裝軟體包的 man 手冊頁
dpkg -S *file_name_pattern*	列出匹配檔名的已安裝軟體包
apt-file search *file_name_pattern*	列出檔案庫中匹配檔名的軟體包
apt-file list *package_name_pattern*	列出檔案庫中匹配的軟體包的內容
dpkg-reconfigure *package_name*	重新調配軟體包
dpkg-reconfigure -plow *package_name*	通過最詳細的方式來重新調配軟體包
configure-debian	以全螢幕選單的形式重新調配軟體包
dpkg --audit	部分安裝軟體包的審計系統
dpkg --configure -a	調配所有部分安裝的軟體包
apt-cache policy *binary_package_name*	顯示一個二進位制軟體包的可用版本、優先順序和檔案庫資訊
apt-cache madison *package_name*	顯示一個軟體包的可用版本和檔案庫資訊
apt-cache showsrc *binary_package_name*	顯示一個二進位制軟體包的原始碼軟體包資訊
apt-get build-dep *package_name*	安裝構建軟體包所需要的軟體包
aptitude build-dep *package_name*	安裝構建軟體包所需要的軟體包
apt-get source *package_name*	（從標準檔案庫）下載原始碼
dget *dsc* 檔案的 *URL*	（從其它檔案庫）下載原始碼軟體包
dpkg-source -x *package_name_version-debian.revision*.dsc	從原始碼軟體包集合（“*.orig.tar.gz”和“*.debian.tar.gz”/”*.diff.gz”）中構建程式碼樹
debuild binary	從本地的原始碼樹中構建軟體包
make-kpkg kernel_image	從核心原始碼樹中構建一個核心軟體包
make-kpkg --initrd kernel_image	從啟用了 initramfs 的核心程式碼樹中構建一個核心軟體包
dpkg -i *package_name_version-debian.revision_arch*.deb	安裝一個本地的軟體包到系統中
apt install /path/to/*package_filename*.deb	安裝一個本地軟體包到系統中，同時，嘗試自動地解析依賴性
debi *package_name_version-debian.revision_arch*.dsc	安裝本地軟體包到系統中
dpkg --get-selections '*' >selection.txt	儲存 dpkg 級別的軟體包選擇狀態資訊
dpkg --set-selections <selection.txt	使用 dpkg 設定軟體包選擇狀態
echo *package_name* hold | dpkg --set-selections	使用 dpkg 將一個軟體包的包選擇狀態設定為 **hold**（相當於”aptitude hold < 包名 >”）

Table 2.13: 高階軟體包管理操作

2.4 高階軟體包管理操作

2.4.1 指令列中的高階軟體包管理操作

下面列出了一些其它的軟體包管理操作，這些操作對於 aptitude 過於高階或缺失所需的功能。

注
對於一個支援多架構的軟體包，你可能需要為一些指令指定架構名稱。例如，使用“dpkg -L libglib2.0-0:amd64”來列出 amd64 架構的 libglib2.0-0 軟體包的內容。

注意
系統管理員應該小心使用低階的軟體包工具（例如“dpkg -i …”和“debi …”），它們不會自動處理所需的軟體包依賴。dpkg 的指令列選項“--force-all”和類似的選項（參見 dpkg(1)）只適用於高手。沒有完全理解它們的效果卻使用它們會破壞你的整個系統。

請注意以下幾點。

- 所有的系統調配和安裝指令都需要以 root 執行。
- 不同於使用正規表達式的 aptitude（參見節 1.6.2），其它的軟體包管理指令使用類似於 shell glob 的萬用字元（參見節 1.5.6）。
- apt-file(1) 由 apt-file 軟體包提供，並且需要先執行“apt-file update”。
- configure-debian(8) 由 configure-debian 軟體包提供，它執行 dpkg-reconfigure(8) 作為後端。
- dpkg-reconfigure(8) 使用 debconf(1) 作為後端來執行軟體包指令碼。
- ”apt-get build-dep”、”apt-get source” 和”apt-cache showsrc” 命令需要 源列表中存在”deb-src” 條目。
- dget(1)、debuild(1) 和 debi(1) 需要 devscripts 軟體包。
- 參見節 2.7.13 裡使用”apt-get source” 的打包（重打包）過程。
- make-kpkg 指令需要 kernel-package 軟體包（參見節 9.10）。
- 通用打包參見節 12.9.

2.4.2 驗證安裝的軟體包檔案

已經安裝 debsums 軟體包的，能使用 debsums(1) 指令通過”/var/lib/dpkg/info/*.md5sums” 檔案中的 MD5sum 值，驗證已安裝的檔案。參見節 10.3.5來獲得 MD5sum 是怎樣工作的資訊。

注
因為 MD5sum 資料庫可能被侵入者篡改，debsums(1) 作為安全工具使用有限。這種工具用於校驗管理者造成的本地修改或媒體錯誤造成的損壞是很不錯的。

2.4.3 預防軟體包故障

許多使用者更想使用 Debian 系統的 **testing**（或 **unstable**）版本，因為它有新的功能和軟體包。但這會使得系統更容易遇到嚴重的軟體包問題。

安裝軟體包 apt-list bugs 可以避免您的系統遭遇嚴重 bugs，在通過 APT 系統升級時，它會自動檢查 Debian BTS 裡的嚴重 bug。

安裝 apt-listchanges 軟體包，在使用 APT 系統升級時它會在“NEWS.Debian”中提供重要新聞。

2.4.4 搜尋軟體包元資料

儘管近來瀏覽 Debian 網站 https://packages.debian.org/ 是搜尋軟體包元資料更加簡單的方法，但我們依舊來看看更傳統的方法。

`grep-dctrl`(1)、`grep-status`(1) 和 `grep-available`(1) 指令被用來搜尋具有 Debian 軟體包控制檔案格式的任何檔案。

“`dpkg -S` *`file_name_pattern`*”能夠被用來搜尋包含該檔案的軟體包名稱，其匹配的名稱是由 dpkg 安裝的。但它會忽略維護者的指令碼建立的檔案。

如果你需要對 dpkg 元資料進行更復雜的搜尋，你需要在“`/var/lib/dpkg/info/`”目錄下執行“`grep -e regex_pattern *`”命令。這會使你在軟體包指令碼和安裝查詢文字中搜索提及的單詞。

如果你想遞迴查詢軟體包依賴，你應該使用 `apt-rdepends`(8)。

2.5 Debian 軟體包內部管理

讓我們來學習 Debian 軟體包管理的內部工作原理。這應該能夠幫助你獨立解決一些軟體包問題。

2.5.1 檔案庫元資料

每個發行版的元資料檔案都儲存在 Debian 映象站的“`dist/`*`codename`*”下面，例如“`http://deb.debian.org/debian/`”。檔案庫的結構可以通過網路瀏覽器來瀏覽。其中有 6 種關鍵的元資料。

檔案	位置	內容
`Release`	發行版的頂層	檔案庫描述和完整性資訊
`Release.gpg`	發行版的頂層	”`Release`” 檔案的簽名檔案，使用檔案庫金鑰簽名
`Contents-`*`architecture`*	發行版的頂層	列出在相關架構中所有軟體包的全部檔案
`Release`	每個發行版/區域/架構組合的頂部	歸檔描述使用 `apt_preferences`(5) 的規則
`Packages`	每個發行版/區域/二進位制架構組合的頂部	連線 `debian/control` 獲得二進位制包
`Sources`	每個發行版/區域/原始碼組合的頂部	連線 `debian/control` 獲得原始碼包

Table 2.14: Debian 檔案庫元資料的內容

為了減少網路流量，在最近的檔案庫中，這些元資料儲存為壓縮了的差分檔案。

2.5.2 頂層“Release”檔案及真實性

> 提示
> 頂層“`Release`”檔案用於簽署 **secure APT** 系統下的歸檔檔案。

每個 Debian 檔案庫的網址都有一個這樣的“`Release`”檔案，例如“`http://deb.debian.org/debian/dists/unstable`”，內容如下。

```
Origin: Debian
Label: Debian
Suite: unstable
Codename: sid
```

```
Date: Sat, 14 May 2011 08:20:50 UTC
Valid-Until: Sat, 21 May 2011 08:20:50 UTC
Architectures: alpha amd64 armel hppa hurd-i386 i386 ia64 kfreebsd-amd64 kfreebsd-i386 mips ↩
     mipsel powerpc s390 sparc
Components: main contrib non-free
Description: Debian x.y Unstable - Not Released
MD5Sum:
 bdc8fa4b3f5e4a715dd0d56d176fc789 18876880 Contents-alpha.gz
 9469a03c94b85e010d116aeeab9614c0 19441880 Contents-amd64.gz
 3d68e206d7faa3aded660dc0996054fe 19203165 Contents-armel.gz
...
```

注
在節 2.1.5 裡，你能夠發現我使用”suite” 和”codename” 的邏輯。“發行版” 被用來同時談及”suite” 和”codename”. 所有由檔案庫提供的歸檔”area” 名，會被列在”Components” 下.

頂層檔案”Release” 的完整性，是由叫 secure apt 的加密架構來驗證，在 apt-secure(8) 中進行描述。

- 加密簽名檔案”Release.gpg” 是由頂層授權檔案”Release” 和加密的 Debian 檔案庫公鑰建立。
- 公開的 Debian 檔案庫公鑰能夠透過安裝 debian-archive-keyring 軟體包來安裝到本地。
- **secure APT** 系統自動驗證下載的頂層檔案”Release” 的完整性。加密驗證過程用到了”Release.gpg” 檔案和本地安裝的 Debian 檔案庫公鑰.
- 所有”Packages” 和”Sources” 檔案的完整性是由在頂層”Release” 檔案裡的 MD5sum 值來驗證。所有軟體包檔案的完整性由”Packages” 和”Sources” 檔案裡的 MD5sum 值來驗證. 參見 debsums(1) 和節 2.4.2.
- 因加密簽名驗證比計算 MD5sum 值消耗更多的 CPU，使用 MD5sum 值來驗證每一個軟體包，使用加密簽名來驗證頂層的”Release” 檔案，這種方式提供 較好安全性的同時，也有比較好的效能 (參見節 10.3).

如果 源列表條目特別指定了”signed-by” 選項，它下載的頂層”Release” 檔案使用這個指定的公鑰來驗證。這在當源列表包含有非 Debian 檔案庫時有用。

提示
不贊成使用 apt-key(8) 命令來管理 APT 金鑰。

當然，你能夠使用 gpg 手工驗證”Release” 的完整性，使用”Release.gpg” 檔案和在 ftp-master.debian.org 上公佈的 Debian 檔案庫公鑰。

2.5.3 檔案庫層的“Release”檔案

提示
檔案庫層的“Release”檔案將用作 apt_preferences(5) 的規則。

歸檔層次的”Release” 檔案,其全部歸檔位置在 源列表中指定,如以下的”http://deb.debian.org/debian/dists/un
或”http://deb.debian.org/debian/dists/sid/main/binary-amd64/Release”。

```
Archive: unstable
Origin: Debian
Label: Debian
Component: main
Architecture: amd64
```

注意
對於"Archive:" 章節，系列名稱 ("stable"，"testing"，"unstable", ⋯) 用於 Debian archive ，而代號 ("trusty", "xenial", "artful", ⋯) 用於 Ubuntu archive。

對於部分檔案庫, 比如說 experimental 和 bookworm-backports, 它們包含的軟體包不會被自動安裝，這是因為有額外的行,例如在"http://deb.debian.org/debian/dists/experimental/main/binary-amd64/Release" 裡面有如下額外的一行。

```
Archive: experimental
Origin: Debian
Label: Debian
NotAutomatic: yes
Component: main
Architecture: amd64
```

請注意，普通的檔案庫沒有"NotAutomatic: yes", 預設的 Pin-Priority 值是 500, 而對於有"NotAutomatic: yes" 的特殊檔案庫, 預設的 Pin-Priority 值是 1 (參見 apt_preferences(5) 和節 2.7.7).

2.5.4 獲得用於軟體包的元資料

當使用 APT 工具時，如 aptitude, apt-get, synaptic, apt-file, auto-apt，我們需要更新包含 Debian 檔案庫資訊元資料的本地複製。這些本地複製的檔名稱，和在 源列表檔案裡面的 distribution, area, architecture 相應名稱一致。(參見節 2.1.5).

- "/var/lib/apt/lists/deb.debian.org_debian_dists_*distribution*_Release"
- "/var/lib/apt/lists/deb.debian.org_debian_dists_*distribution*_Release.gpg"
- "/var/lib/apt/lists/deb.debian.org_debian_dists_*distribution_area*_binary-*architecture*_Pac
- "/var/lib/apt/lists/deb.debian.org_debian_dists_*distribution_area*_source_Sources"
- "/var/cache/apt/apt-file/deb.debian.org_debian_dists_*distribution*_Contents-*architecture*.g (apt-file)

前 4 種類型的檔案是所有相關的 APT 命令共享的，並且可以透過“apt-get update”或“aptitude update”在命令列中進行更新。如果在源列表中有相應的“deb”行，則“軟體包”元資料會進行更新。如果在 源列表中有相應的“deb-src”行，則“原始碼”元資料會進行更新。

"Packages" 和"Sources" 的元資料檔案包含有“Filename:”欄位，指向二進位制和原始碼包檔案的位置。目前，這些軟體包都統一放在"pool/" 目錄樹下，這樣可以改善跨版本釋出的傳輸。

“軟體包”元資料的本地副本可以使用 aptitude 來進行互動式的搜尋。專門的搜尋指令 grep-dctrl(1) 可以搜尋“軟體包”和“原始碼”元資料的本地副本。

"Contents-*architecture*" 元資料的本地拷貝，能夠被"apt-file update" 更新，它的位置和其它 4 個不同。參見 apt-file(1). (auto-apt 的"Contents-*architecture*.gz" 檔案的本地拷貝預設也使用不同的位置。)

2.5.5 APT 的軟體包狀態

除了遠端獲得元資料,lenny 之後的 APT 工具還會將它在本地產生的安裝狀態資訊儲存在“/var/lib/apt/extended_sta 中，APT 會使用它們來追蹤自動安裝的所有軟體包。

2.5.6 aptitude 的軟體包狀態

除了遠端獲得元資料,aptitude 指令還會將它在本地產生的安裝狀態資訊儲存在“/var/lib/aptitude/pkgstates” 中，這些資訊只能被 aptitude 使用。

2.5.7 獲得的軟體包的本地副本

所有通過 APT 機制遠端獲得的軟體包都被儲存在“/var/cache/apt/archives”中，直到它們被清除。

aptitude 的這個快取檔案清理策略，能夠在”Options” → ”Preferences” 下設定，也可以通過它的選單，”Actions” 下的”Clean package cache” 或”Clean obsolete files” 來執行強制清理。

2.5.8 Debian 軟體包檔名稱

Debian 軟體包檔案有特定的名稱結構。

軟體包型別	名稱結構
二進位制軟體包（亦稱 deb）	*package-name_upstream-version-debian.revision_architec*
用於 debian-installer 的二進位制軟體包（亦稱 udeb）	*package-name_upstream-version-debian.revision_architec*
原始碼軟體包（上游原始碼）	*package-name_upstream-version-debian.revision*.orig.tar
1.0 原始碼軟體包 (Debian 改變)	*package-name_upstream-version-debian.revision*.diff.gz
3.0 (quilt 補丁管理工具) 原始碼軟體包 (Debian 改變)	*package-name_upstream-version-debian.revision*.debian.ta
原始碼軟體包（說明）	*package-name_upstream-version-debian.revision*.dsc

Table 2.15: Debian 軟體包的名稱結構

提示
這裡僅敘述了基本的原始碼包格式。更多內容請參考 dpkg-source(1)。

名稱元件	可用的字元（正則表示式）	存在狀態
package-name	[a-z0-9][-a-z0-9.+]+	必需
epoch:	[0-9]+:	可選
upstream-version	[-a-zA-Z0-9.+:]+	必需
debian.revision	[a-zA-Z0-9.+~]+	可選

Table 2.16: Debian 軟體包名稱中每一個元件可以使用的字元

注
你可以用 dpkg(1) 提供的指令檢查軟體包版本，例如.，”dpkg --compare-versions 7.0 gt 7.~pre1 ; echo $?” .

注
debian-installer (d-i) 使用 udeb 作為它的二進位制軟體包的副檔名，而非普通的 deb。一個 udeb 軟體包是從 deb 軟體包中剝離了一些不必要的內容（例如文件），從而節省空間同時也放寬軟體包政策的要求。deb 和 udeb 軟體包會共享相同的軟體包結構。“u”表示微小。

2.5.9 dpkg 指令

dpkg(1) 是 Debian 軟體包管理中最底層的工具。它非常強大，必須小心使用。

當安裝名為“*package_name*”的軟體包時，dpkg 會按照下列的順序處理它。

1. 解包 deb 檔案（等同於“ar -x”）
2. 使用 debconf(1) 執行“*package_name*.preinst”
3. 將軟體包安裝到系統中（等同於“tar -x”）
4. 使用 debconf(1) 執行“*package_name*.postinst”

debconf 系統提供帶有 I18N 和 L10N （章 8）支援的標準化使用者互動。

檔案	內容說明
/var/lib/dpkg/info/*package_name*.conffiles	列出組態檔案。（使用者可修改的）
/var/lib/dpkg/info/*package_name*.list	列出軟體包安裝的所有檔案和目錄
/var/lib/dpkg/info/*package_name*.md5sums	列出軟體包安裝的檔案的 MD5 雜湊值
/var/lib/dpkg/info/*package_name*.preinst	軟體包安裝之前執行的軟體包指令碼
/var/lib/dpkg/info/*package_name*.postinst	軟體包安裝之後執行的軟體包指令碼
/var/lib/dpkg/info/*package_name*.prerm	軟體包移除之前執行的軟體包指令碼
/var/lib/dpkg/info/*package_name*.postrm	軟體包移除之後執行的軟體包指令碼
/var/lib/dpkg/info/*package_name*.config	用於 debconf 系統的軟體包指令碼
/var/lib/dpkg/alternatives/*package_name*	update-alternatives 指令使用的替代資訊
/var/lib/dpkg/available	所有軟體包的可用性資訊
/var/lib/dpkg/diversions	dpkg(1) 使用的檔案移動資訊，由 dpkg-divert(8) 設定
/var/lib/dpkg/statoverride	dpkg(1) 使用的檔案狀態改變資訊，由 dpkg-statoverride(8) 設定
/var/lib/dpkg/status	所有軟體包的狀態資訊
/var/lib/dpkg/status-old	“var/lib/dpkg/status”檔案的第一代備份
/var/backups/dpkg.status*	第二代備份，以及“var/lib/dpkg/status”檔案更舊的備份

Table 2.17: dpkg 建立的重要檔案

“status”檔案也被例如 dpkg(1)、“dselect update”和“apt-get -u dselect-upgrade”等工具使用。專門的搜尋指令 grep-dctrl(1) 可以搜尋“status”和“available”元資料的本地副本。

提示
在debian 安裝器環境下, udpkg 指令用於開啟 udeb 軟體包，udpkg 指令是 dpkg 指令的一個精簡版本.

2.5.10 update-alternatives 指令

Debian 系統使用 update-alternatives(1) 讓使用者可以不受干擾地安裝多種重疊的程式。例如，如果同時安裝了 vim 和 nvi 軟體包，你可以使 vi 指令選擇執行 vim。

```
$ ls -l $(type -p vi)
lrwxrwxrwx 1 root root 20 2007-03-24 19:05 /usr/bin/vi -> /etc/alternatives/vi
$ sudo update-alternatives --display vi
...
$ sudo update-alternatives --config vi
  Selection    Command
 -----------------------------------------------
      1        /usr/bin/vim
*+    2        /usr/bin/nvi

Enter to keep the default[*], or type selection number: 1
```

Debian 選擇系統在”/etc/alternatives/” 目錄裡通過符號連結來維持它的選擇。選擇程序使用”/var/lib/dpkg/alter
目錄裡面的相應檔案。

2.5.11 dpkg-statoverride 指令

當安裝一個軟體包時，由 dpkg-statoverride(8) 指令提供的 狀態修改，是告訴 dpkg(1) 對 檔案使用不同的屬主或許可權的一個方法。如果使用了"--update" 選項，並且檔案存在，則該檔案會被立即設定為新的屬主和模式。

注意
系統管理員使用 chmod 或 chown 指令直接修改某個軟體包檔案的屬主或許可權，將會在下次軟體包升級時，被重置。

注
本人在此使用了檔案一詞，但事實上也可用於 dpkg 所處理的任何檔案系統物件，包括目錄，裝置等。

2.5.12 dpkg-divert 指令

dpkg-divert(8) 指令提供的檔案轉移，是迫使 dpkg(1) 將檔案不安裝到其預設位置，而是安裝到轉移的位置。dpkg-divert 是軟體包維護指令碼。不建議系統管理員使用這個指令。

2.6 從損壞的系統中恢復

當執行 測試版或 不穩定版系統，系統管理員會遇到從錯誤的軟體包管理進行恢復的情形。

注意
下面的一些方法具有很高的風險。在此先對你進行警告！

2.6.1 缺少依賴導致的安裝失敗

如果你透過"sudo dpkg -i ..." 強制安裝一個軟體包到系統，而不安裝它所依賴的所有軟體包，這個軟體包將作為“部分安裝”而失敗。

你應當安裝所有依賴的軟體包，使用 APT 系統或者"sudo dpkg -i ..."。

然後，使用下列命令來配置所有部分安裝的軟體包。

```
# dpkg --configure -a
```

2.6.2 軟體包資料快取錯誤

軟體包資料快取錯誤，能夠造成奇怪的錯誤，比如 APT 的 "GPG error: ... invalid: BADSIG ..."。

你應該透過"sudo rm -rf /var/lib/apt/* "刪除所有快取的資料,然後重新嘗試。(如果使用了 apt-cacher-ng, 你還應執行"sudo rm -rf /var/cache/apt-cacher-ng/* "。)

2.6.3 不相容舊的使用者調配

如果一個桌面 GUI 程式在重要的上游版本升級後變得不穩定，你應該懷疑這是舊的本地配置檔案（由它建立的）所導致的。如果它在新建的使用者賬號下執行穩定，那麼這個假設就得到了證實。（這是一個打包的 bug 並且打包者通常會避免它。）

為了恢復穩定，你應該移除相應的本地組態檔案並重新啟動 GUI 程式。你可能需要閱讀舊的組態檔案內容以便之後恢復調配資訊。（別將它們刪得太快了。）

2.6.4 具有相同檔案的不同軟體包

文件級的軟體包管理系統，比如說 `aptitude`(8) 或 `apt-get`(1), 使用軟體包依賴，當出現相同檔案時，不會嘗試去安裝軟體包。(參見節 2.1.7).

軟體包維護者的錯誤，或者系統管理員調配了不一致的檔案庫混合源，(參見節 2.7.6)，都會出現不正確的軟體包依賴情況。如果在出現相同檔案的情況下，你通過 `aptitude`(8) 或 `apt-get`(1) 安裝軟體包，`dpkg`(1) 在對軟體包解包時，確定會給呼叫程式回傳錯誤，並不會覆蓋已經存在的檔案。

注意
使用第三方軟體包會導致重大的系統風險，因為其通過使用 root 許可權執行維護者指令碼能夠對你的系統做任何事。`dpkg`(1) 指令只防止解包時的覆蓋行為。

可以先通過刪除舊的令人討厭的軟體包，*`old-package`*，來解決這類錯誤的安裝問題。

```
$ sudo dpkg -P old-package
```

2.6.5 修復損壞的軟體包指令碼

當軟體包指令碼中的一個指令由於某些原因回傳錯誤，指令碼也將由於錯誤而退出，軟體包管理系統忽略它們的行為，並導致部分安裝的軟體包。當一個軟體包在它的刪除指令碼中有錯誤時，該軟體包將會成為不可能刪除的軟體包，處理這些問題，都會變得相當棘手。

對於“*package_name*”的軟體包指令碼問題，你應該檢視下列的軟體包指令碼。

- ”`/var/lib/dpkg/info/`*`package_name`*`.preinst`”
- ”`/var/lib/dpkg/info/`*`package_name`*`.postinst`”
- ”`/var/lib/dpkg/info/`*`package_name`*`.prerm`”
- ”`/var/lib/dpkg/info/`*`package_name`*`.postrm`”

使用下列的方法，以 root 編輯損壞的軟體包指令碼。

- 在行首新增“#”可以禁用出錯的行
- 在出錯行的行尾新增“`|| true`”可以強制回傳成功

然後，按照節 2.6。

2.6.6 使用 dpkg 指令進行救援

因為 dpkg 是非常底層的軟體包工具，它可以在很糟糕的情況下進行工作，例如無法啟動系統且沒有網路連線。讓我們假定 foo 軟體包損壞了，並且需要更換。

你可以在軟體包快取目錄：“/var/cache/apt/archives/”中找到舊的 foo 軟體包的無 bug 版本。(如果找不到，你可以從檔案庫 https://snapshot.debian.org/ 中下載它，或從具有軟體包快取功能的機器中複製它。)

如果你能夠啟動系統，你可以通過下列指令來安裝它。

```
# dpkg -i /path/to/foo_old_version_arch.deb
```

提示
如果你係統損壞較小，你也可以使用更高層的 APT 系統來降級整個系統，就像節 2.7.11 中做的那樣。

如果你的系統無法從硬碟啟動，你應該尋找其它方式來啟動它。

1. 使用 Debian 安裝光碟以救援模式啟動系統。
2. 將硬碟上無法啟動的系統掛載到“/target”。
3. 通過下列指令安裝舊版本的 foo 軟體包。

```
# dpkg --root /target -i /path/to/foo_old_version_arch.deb
```

即使位於硬碟上的 dpkg 指令已損壞，該指令依舊可以執行。

提示
任何由硬碟、live GNU/Linux CD、可啟動的 USB 驅動或網路啟動上的另一系統啟動的 GNU/Linux 系統到可以類似地用來救援損壞的系統。

如果由於依賴問題,無法用這種方式安裝軟體包,並且你真的必須真麼做,你可以使用 dpkg 的“--ignore-depends”、“--fo
和其它選項來無視依賴。如果你這麼做了，之後你必須認真努力地修復依賴關係。更多細節參見 dpkg(8)。

注
如果你的系統嚴重損壞了，你應該將系統完整備份到一個安全的地方 (參見節 10.2) 並進行一次全新的安裝。這是耗時較少且效果較好的辦法。

2.6.7 恢復軟體包選擇資料

如果“/var/lib/dpkg/status”因為某種原因出現錯誤,Debian 系統會丟失軟體包選擇資料並受到嚴重影響。尋找位於“/var/lib/dpkg/status-old”或“/var/backups/dpkg.status.*”中舊的“/var/lib/dpkg/status”檔案。

給“/var/backups/”分配一個單獨的分割槽是一個好習慣，因為這個目錄包含了許多重要的系統資料。

對於嚴重的損壞,我建議備份系統後重新安裝。即使失去“/var/”中的所有資料,你依舊可以從“/usr/share/doc/”目錄恢復一些資訊來引導你進行新的安裝。

重新安裝最小 (桌面) 系統。

```
# mkdir -p /path/to/old/system
```

將舊系統掛載到“/path/to/old/system/”。

```
# cd /path/to/old/system/usr/share/doc
# ls -1 >~/ls1.txt
# cd /usr/share/doc
# ls -1 >>~/ls1.txt
# cd
# sort ls1.txt | uniq | less
```

然後你就可以根據軟體包名稱來進行安裝了。(可能會有一些非軟體包名稱，例如“texmf”。)

2.7 軟體包管理技巧

出於簡化，在 bookworm 釋出後，在這個章節的源列表例子，使用單行式樣在”/etc/apt/sources.list” 裡表示。

2.7.1 上傳軟體包的是誰？

儘管“/var/lib/dpkg/available”和“/usr/share/doc/package_name/changelog”中列出的維護者姓名提供了關於“軟體包運作的幕後者是誰”這一問題的一些資訊，但軟體包的實際上傳者依舊不明。devscripts 軟體包中的 who-uploads(1) 可以識別 Debian 源軟體包的實際上傳者。

2.7.2 限制 APT 的下載頻寬

如果你想限制 APT 的下載頻寬到 800Kib/sec (=100KiB/sec)，你應該像下面那樣設定 APT 的調配參數。

```
APT::Acquire::http::Dl-Limit ”800”;
```

2.7.3 自動下載和升級軟體包

apt 軟體包有自己的 cron 指令碼“/etc/cron.daily/apt”,它支援自動下載軟體包。可以安裝 unattended-upgrades 軟體包來增強這個指令碼,使它能夠自動升級軟體包。可以通過“/etc/apt/apt.conf.d/02backup”和“/etc/apt/apt.
中的參數來進行自定義，相關說明位於“/usr/share/doc/unattended-upgrades/README”中。

unattended-upgrades 軟體包主要用於 stable 系統的安全更新。如果自動升級損壞 stable 系統的風險小於被入侵者利用已被安全更新修復的安全漏洞，你應該考慮使用自動更新，調配引數如下。

```
APT::Periodic::Update-Package-Lists ”1”;
APT::Periodic::Download-Upgradeable-Packages ”1”;
APT::Periodic::Unattended-Upgrade ”1”;
```

如果你執行的是 testing 或 unstable 系統，你應該不會想要使用自動更新，因為它肯定會在某天損壞系統。即使位於這樣的 testing 或 unstable 情況下，你可能依舊想提前下載軟體包以節省互動式升級的時間，其配置引數如下。

```
APT::Periodic::Update-Package-Lists ”1”;
APT::Periodic::Download-Upgradeable-Packages ”1”;
APT::Periodic::Unattended-Upgrade ”0”;
```

2.7.4 更新和向後移植

stable-updates (”bookworm-updates”，在 bookworm-作為-stable 釋出迴圈) 和 backports.debian.org 檔案庫提供了 stable 版軟體包更新。

為了去使用這些檔案庫，你需要在”/etc/apt/sources.list” 檔案裡寫入如下所示的檔案庫列表。

```
deb http://deb.debian.org/debian/ bookworm main non-free-firmware contrib non-free
deb http://security.debian.org/debian-security bookworm-security main non-free-firmware ↩
    contrib non-free
deb http://deb.debian.org/debian/ bookworm-updates main non-free-firmware contrib non-free
deb http://deb.debian.org/debian/ bookworm-backports main non-free-firmware contrib non- ↩
    free
```

並不需要在”/etc/apt/preferences” 檔案中顯式設定 Pin-Priority 值. 當新的包可用時，預設調配提供了更合理的更新 (請見節 2.5.3).

- 所有已安裝的舊軟體包都可以通過 bookworm-updates 檔案庫升級到新軟體包。
- 只有從 bookworm-backports 檔案庫中手動安裝的舊軟體包才會通過 bookworm-backports 檔案庫升級到新軟體包。

當你想要從 bookworm-backports 檔案庫中手動的安裝一個名叫”*package-name*” 的軟體及其依賴包的時候，你應該在目標檔案庫之前加一個 “-t” 參數。

```
$ sudo apt-get install -t bookworm-backports package-name
```

警告
不要從 backports.debian.org 檔案庫安裝太多軟體包。它能夠造成軟體包依賴複雜。替代解決方案參見節 2.1.11。

2.7.5 外部軟體包檔案庫

警告
你應當小心，外部軟體包獲取你係統的 root 許可權。你應當只使用可信賴的外部軟體包檔案庫。替代方案參見節 2.1.11。

你能夠使用安全 APT 來使用 Debian 相容的外部軟體包檔案庫，將它加入到 源列表，並把它的檔案庫金鑰放入”/etc/apt/trusted.gpg.d/” 目錄。參見 sources.list(5)、apt-secure(8) 和 apt-key(8)。

2.7.6 不使用 apt-pinning 的混合源檔案庫軟體包

注意
從混合源檔案庫中安裝軟體包是不被 Debian 官方發行版所支援的，除了官方支援的檔案庫的特殊組合以外，例如 stable 的 security updates 和 stable-updates。

這裡有一個列子，在原有隻跟蹤 testing 的場景，操作包含在 unstable 裡發現的新的上游軟體包版本。

1. 臨時更改”/etc/apt/sources.list” 檔案，使之指向單一的”unstable” 發行版路徑。
2. 執行”aptitude update” 指令。
3. 執行”aptitude install *package-name*” 指令。

4. 恢復到原始"/etc/apt/sources.list" 檔案，使之指向 testing 路徑。
5. 執行"aptitude update" 指令。

使用這個手工方法，你不需要建立"/etc/apt/preferences" 檔案，也不需要擔心 **apt-pinning**。但這個方法仍然是非常麻煩的。

注意
當使用混合檔案源的時候，因為 Debian 不會確保軟體之間的相容性，所以你必須自己去解決相容性問題。如果軟體之間存在不相容性，系統可能會損壞。你必須能夠判斷這些操作所需的技術要求。使用任意混合的檔案源是完全可選的操作，我並不鼓勵你去使用它。

從不同的檔案庫中安裝軟體包的一般規則如下。

- 非二進位制軟體包 ("Architecture: all") 的安裝是更保險的。
  - 文件軟體包：沒有特別的要求
  - 解釋程式的軟體包：相容的直譯器必須是可用的
- 二進位制軟體包 (non "Architecture: all") 通常會面臨很多障礙，它的安裝不保險的。
  - 庫檔案版本的相容性 (包括"libc")
  - 與之相關的有用的程式版本的相容性
  - 核心 ABI 的相容性
  - C++ ABI 的相容性
  - …

注
為了使軟體包的安裝變得更保險，一些商業的非自由的二進位制程式包可能會提供完整的靜態連結庫。你還是應該檢查 ABI 的相容性問題等等。

注
除非為了短期避免破壞軟體包，從非 Debian 檔案庫安裝二進位制軟體包通常是一個壞的主意。你需要查詢所有存在的和你目前 Debian 系統相容的安全技術替代方案。(參見節 2.1.11)。

2.7.7 使用 apt-pinning 調整獲選版本

警告
新手用 **apt-pinning** 命令會造成比較大的問題。你必須避免使用這個命令除非確實需要它。

沒有"/etc/apt/preferences" 檔案，APT 系統使用版本字串來選擇最新的可用版本作為 候選版本。這是通常的狀態，也是 APT 系統最推薦的使用方法。所有官方支援的檔案庫集合，並不要求"/etc/apt/preferences" 檔案，因此，一些不應當被作為自動更新源的軟體包，被標記為 **NotAutomatic**，並被適當處理。

提示
版本字串的比較規則可以被驗證，例子如下，"dpkg --compare-versions ver1.1 gt ver1.1~1; echo $?" (參見 dpkg(1))。

如果經常從混合源檔案庫中安裝軟體包 (參見節 2.7.6), 你可以通過建立"/etc/apt/preferences" 檔案並且在其中寫入關於調整候選版本的軟體包選取規則的合適條目 (如 apt_preferences(5) 中所示) 來自動化這些複雜的操作。這被稱為 **apt-pinning**。

當使用 **apt-pinning** 命令時，因為 Debian 不會確保軟體之間的相容性，所以你必須自己確認其相容性。**apt-pinning** 是完全可選的操作，我並不建議去使用它。

檔案庫層級的 Release 檔案 (參見節 2.5.3) 使用 apt_preferences(5) 的規則. 對於 Debian 通用檔案庫 和 Debian 安全檔案庫，**apt-pinning** 只在"suite" 名下工作。(這點和 Ubuntu 檔案庫不同.) 例如，你在"/etc/apt/preferences" 檔案裡面，可以使用"Pin: release a=unstable"，但不能使用"Pin: release a=sid".

當使用非 Debian 的檔案庫作為 **apt-pinning** 的一部分時，你應該檢查它們的用途和可信度。例如，Ubuntu 和 Debian 是不能混在一起的。

注
即使不建立"/etc/apt/preferences" 檔案，在不用 **apt-pinning** 命令的情況下，你也可以進行相當複雜的系統工作 (參見節 2.6.6和節 2.7.6)。

如下是關於 **apt-pinning** 技術的簡化說明。

可用的軟體包源在"/etc/apt/sources.list" 檔案裡面定義，APT 系統從可用的軟體包源裡面選擇 Pin-Priority 值最大的，作為升級軟體包的候選版本. 如果一個軟體包的 Pin-Priority 大於 1000, 這個版本限制為只能 升級，關閉了軟體包降級功能 (參見節 2.7.11).

每個軟體包的 Pin-Priority 值是在"/etc/apt/preferences" 檔案中的"Pin-Priority" 條目中定義或者是使用它的預設值。

Pin-Priority	**apt-pinning** 對軟體包的影響
1001	安裝該軟體包，即使是一個降級軟體包的指令
990	用作目標發行版檔案庫的預設值
500	用作常規檔案庫的預設值
100	用於 **NotAutomatic** 和 **ButAutomaticUpgrades** 檔案庫的預設值
100	用於已安裝軟體包
1	用於 **NotAutomatic** 檔案庫的預設值
-1	即使被推薦，也絕不安裝這個軟體包

Table 2.18: 用於 **apt-pinning** 技術的值得注意的 Pin-Priority 值列表。

目標版本檔案倉庫，能夠由命令列選項設定，例如: "apt-get install -t testing some-package"

NotAutomatic 和 **ButAutomaticUpgrades** 的檔案是由檔案庫伺服器上檔案層級的 Release 檔案來設定，(參見節 2.5.3)，同時包含"NotAutomatic: yes" 和"ButAutomaticUpgrades: yes". 而 **NotAutomatic** 檔案也是由檔案庫伺服器上的檔案層級的 Release 檔案來設定，但只包含"NotAutomatic: yes".

來自眾多檔案源的軟體包的 **apt-pinning** 情況可以通過"apt-cache policy *package*" 指令顯示。

- "Package pin:" 開頭的行，列出了軟體包版本的 **pin**，如果 *package* 相關的 pin 已經定義, 例如, "Package pin: 0.190".
- 沒有"Package pin:" 的行存在，如果沒有 *package* 相關的定義。
- 與 *package* 相關的 Pin-Priority 值列在所有版本字串的右邊，比如，"0.181 700".
- "0" 是列在所有版本字串的右邊，如果沒有 *package* 相關的定義。例如, "0.181 0".
- 檔案庫 (在"/etc/apt/preferences" 檔案作為"Package: *" 定義) 的 Pin-Priority 值，列在所有檔案庫路徑的左邊，例如，"100 http://deb.debian.org/debian/ bookworm-backports/main Packages".

2.7.8 阻止推薦的軟體包的安裝

警告
新手用 **apt-pinning** 命令會造成比較大的問題。你必須避免使用這個命令除非確實需要它。

如果不想要引入推薦的特定軟體包，你必須建立"`/etc/apt/preferences`" 檔案並且像如下所示的那樣在檔案的頂部明確列出這些軟體包。

```
Package: package-1
Pin: version *
Pin-Priority: -1

Package: package-2
Pin: version *
Pin-Priority: -1
```

2.7.9 使用帶有 **unstable** 軟體包的 **testing** 版本

警告
新手用 **apt-pinning** 命令會造成比較大的問題。你必須避免使用這個命令除非確實需要它。

如下是一個關於 **apt-pinning** 技術的例子，當使用 `testing` 的時候，實現 `unstable` 中的特定的較新的上游版本軟體包的日常升級。你應該按如下所示的在"`/etc/apt/sources.list`" 檔案中列出所有需要的檔案庫。

```
deb http://deb.debian.org/debian/ testing main contrib non-free
deb http://deb.debian.org/debian/ unstable main contrib non-free
deb http://security.debian.org/debian-security testing-security main contrib
```

按如下所示的設定"`/etc/apt/preferences`" 檔案。

```
Package: *
Pin: release a=unstable
Pin-Priority: 100
```

當想要在此調配下從 `unstable` 檔案庫中安裝"*`package-name`*" 軟體及它的依賴包時，你執行帶有"`-t`" 選項 (`unstable` 的 Pin-Priority 值變為 990) 的轉換目標發行版的指令。

```
$ sudo apt-get install -t unstable package-name
```

在此調配下,執行"`apt-get update`"和"`apt-get dist-upgrade`"(或者"`aptitude safe-upgrade`"和"`aptitude full-upgrade`") 指令, 會從 `testing` 檔案庫升級那些從 `testing` 檔案庫安裝的軟體包並且從 `unstable` 檔案庫升級那些從 `unstable` 檔案庫中安裝的軟體包。

注意
小心不要去移除"`/etc/apt/sources.list`" 檔案中的"`testing`" 檔案庫。如果檔案中沒有"`testing`"，APT 系統會使用更加新的 `unstable` 檔案庫升級軟體包。

提示
我通常會在上述操作後，馬上註釋掉”/etc/apt/sources.list” 檔案中的”unstable” 檔案庫記錄。這避免了因為處理”/etc/apt/sources.list” 檔案中的眾多記錄而造成的升級緩慢雖然同時也阻止了那些從 unstable 檔案庫中安裝的軟體包通過 unstable 升級。

提示
如果”/etc/apt/preferences” 檔案中”Pin-Priority: 1” 替代了”Pin-Priority:100”，即使”/etc/apt/sources.list” 檔案中的”testing” 記錄被刪除了，Pin-Priority 值為 100 的已安裝軟體包也不會通過 unstable 檔案庫升級。

如果你希望自動跟蹤 unstable 裡某些特殊的軟體包，而在安裝時不再使用初始化選項”-t unstable”，你必須建立”/etc/apt/preferences” 檔案，並在該檔案頂部按下面的方式清晰的列出所有那些軟體包。

```
Package: package-1
Pin: release a=unstable
Pin-Priority: 700

Package: package-2
Pin: release a=unstable
Pin-Priority: 700
```

如下是為每個特定的軟體包設定 Pin-Priority 值。例如，為了使用最新的 unstable 的英文版”Debian Reference”，你應該在”/etc/apt/preferences” 檔案中寫入以下條目。

```
Package: debian-reference-en
Pin: release a=unstable
Pin-Priority: 700

Package: debian-reference-common
Pin: release a=unstable
Pin-Priority: 700
```

提示
即使你使用的是 stable 檔案庫，**apt-pinning** 技術仍然是有效的。根據我以前的經驗，從 unstable 檔案庫安裝的文件包一直是安全的。

2.7.10 使用帶有 `experimental` 軟體包的 `unstable` 版本

警告
新手用 **apt-pinning** 命令會造成比較大的問題。你必須避免使用這個命令除非確實需要它。

這是使用 **apt-pinning** 的另一個示例，該示例主要使用 unstable 源，但包含了 experimental 源, 該源可用於安裝上游更新的軟體包. 需要包含在”/etc/apt/sources.list” 檔案中的列表如下:

```
deb http://deb.debian.org/debian/ unstable main contrib non-free
deb http://deb.debian.org/debian/ experimental main contrib non-free
deb http://security.debian.org/ testing-security main contrib
```

由於 experimental 源是 非自動 (**NotAutomatic**) 的源 (參見節 2.5.3)，其預設的 Pin-Priority 值被設定為 1 (<<100) . 並不需要在”/etc/apt/preferences” 檔案中設定 Pin-Priority 值，只需要指定 experimental 源，除非你需要在下次更新時自動升級時更新特定軟體包.

2.7.11 緊急降級

警告
新手用 **apt-pinning** 命令會造成比較大的問題。你必須避免使用這個命令除非確實需要它。

注意
降級在 Debian 設計上就不被官方支援。僅僅是在緊急恢復過程中需要做的一部分工作。儘管憎恨這種情形，但降級在很多場景下工作得也不錯。對於重要系統，你應當在恢復操作後備份所有重要資料，並從零開始重新安裝一個新的系統。

你可以通過控制候選版本從新的檔案庫降級到舊的檔案庫（參見節 2.7.7），從而使損壞的系統恢復。下面是一種懶惰的方法，可以避免許多冗長的“`dpkg -i broken-package_old-version.deb`”指令（參見節 2.6.6）。

搜尋“`/etc/apt/sources.list`”檔案中像下面那樣使用 `unstable` 的行。

```
deb http://deb.debian.org/debian/ sid main contrib non-free
```

使用下面的行替換它，從而改為使用 `testing`。

```
deb http://deb.debian.org/debian/ trixie main contrib non-free
```

按如下所示的設定”`/etc/apt/preferences`” 檔案。

```
Package: *
Pin: release a=testing
Pin-Priority: 1010
```

執行“`apt-get update; apt-get dist-upgrade`”使整個系統的軟體包強制降級。

在緊急降級後，移除“`/etc/apt/preferences`”這個特殊的檔案。

提示
這是一個好方法，移除（不是清除！）儘可能多地軟體包，來減少依賴問題。你可能需要手動移除和安裝一些軟體包來使系統降級。需要特別注意 Linux 核心、載入程式、udev、PAM、APT 和網路相關的軟體包以及它們的組態檔案。

2.7.12 equivs 軟體包

如果你從原始碼編譯了一個程式來代替 Debian 軟體包，最好將它做成一個真正的本地 Debian 軟體包（`*.deb`）並使用私人檔案庫。

如果你選擇從原始碼編譯一個程式並將它安裝到“`/usr/local`”，你可能需要使用 `equivs` 作為最後步驟來滿足缺失的軟體包依賴。

```
Package: equivs
Priority: optional
Section: admin
Description: Circumventing Debian package dependencies
 This package provides a tool to create trivial Debian packages.
 Typically these packages contain only dependency information, but they
 can also include normal installed files like other packages do.
 .
 One use for this is to create a metapackage: a package whose sole
```

```
purpose is to declare dependencies and conflicts on other packages so
that these will be automatically installed, upgraded, or removed.
.
Another use is to circumvent dependency checking: by letting dpkg
think a particular package name and version is installed when it
isn't, you can work around bugs in other packages' dependencies.
(Please do still file such bugs, though.)
```

2.7.13 移植一個軟體包到 stable 系統

注意
由於系統差異，不能夠保證這裡描述的過程能夠工作，而不需要額外的手工處理。

對於部分升級的 stable 系統，使用源軟體包在執行環境中重新構建一個軟體包是不錯的選擇。這可以避免因為依賴關係導致大量軟體包升級。

在 stable 系統的“/etc/apt/sources.list”檔案中新增下列條目。

```
deb-src http://deb.debian.org/debian unstable  main contrib non-free
```

如下安裝編譯所需的軟體包並下載源軟體包。

```
# apt-get update
# apt-get dist-upgrade
# apt-get install fakeroot devscripts build-essential
# apt-get build-dep foo
$ apt-get source foo
$ cd foo*
```

如果需要向後移植，可以從 backport 的軟體包中更新一些工具鏈軟體包，例如 dpkg 和 debhelper。

執行下列指令。

```
$ dch -i
```

更新軟體包版本，例如在“debian/changelog”中附加一個“+bp1”

像下面那樣構建軟體包並將它們安裝到系統中。

```
$ debuild
$ cd ..
# debi foo*.changes
```

2.7.14 用於 APT 的代理伺服器

因為映象整個 Debian 檔案庫的子區會浪費硬碟和網路頻寬，當你管理許多 LAN 上的系統時，為 APT 部署一個本地代理伺服器是個好主意。APT 可以通過調配來使用通用 web（http）代理伺服器，例如 squid（參見節 6.5），細節參見 apt.conf(5) 和“/usr/share/doc/apt/examples/configure-index.gz”。環境變數“$http_proxy 會覆蓋“/etc/apt/apt.conf”檔案中設定的代理伺服器。

這裡有一些 Debian 檔案庫的專用代理工具。你應該在使用它們之前檢查 BTS。

注意
當 Debian 重構它的檔案庫結構時，這些專用的代理工具往往需要軟體包維護者重寫程式碼，並可能在一段時間內無法使用。另一方面，通用 web（http）代理伺服器更強健並且更容易應對這種改變。

軟體包	流行度	大小	說明
approx	V:0, I:0	7124	快取 Debian 檔案庫檔案的代理伺服器（已編譯的 OCaml 程式）
apt-cacher	V:0, I:0	266	為 Debian 軟體包和原始碼檔案進行快取代理（Perl 程式）
apt-cacher-ng	V:4, I:4	1816	分發軟體包的快取代理（C++ 編譯的程式）

Table 2.19: Debian 檔案庫的專用代理工具

2.7.15 更多關於軟體包管理的文件

你可以從下面的文件中瞭解軟體包管理的更多資訊。

- 軟體包管理的主要文件：
  - aptitude(8), dpkg(1), tasksel(8), apt(8), apt-get(8), apt-config(8), apt-secure(8), sources.list(5), apt.conf(5), and apt_preferences(5);
  - 來自 apt-doc 軟體包的“/usr/share/doc/apt-doc/guide.html/index.html”和“/usr/share/doc/apt-doc
  - 來自 aptitude-doc-en 軟體包的 “/usr/share/doc/aptitude/html/en/index.html”。
- Debian 檔案庫的官方詳細文件：
  - ”Debian Policy Manual Chapter 2 - The Debian Archive”，
  - ”Debian Developer’s Reference, Chapter 4 - Resources for Debian Developers 4.6 The Debian archive”，
  - ”The Debian GNU/Linux FAQ, Chapter 6 - The Debian FTP archives”。
- 為 Debian 使用者構建一個 Debian 軟體包的教學：
  - “Debian 維護者指南”。

Chapter 3

系統初始化

作爲系統管理員，粗略地瞭解 Debian 系統的啓動和調配方式是明智的。儘管準確的細節在安裝的軟體包及對應的文檔中，但這些知識對我們大多數人來說都是必須掌握的。

下面是 Debian 系統初始化的要點概述。由於 Debian 系統在不斷發展，您應該參考最新的文件。

- Debian Linux 核心手冊 是關於 Debian 核心的主要資訊來源。
- `bootup`(7) 介紹了基於 `systemd` 的系統啟動流程。(近期的 Debian)
- `boot`(7) 介紹了基於 UNIX System V Release 4 的系統啟動流程。(舊版的 Debian)

3.1 啓動過程概述

電腦系統從上電事件到能爲使用者提供完整的作業系統 (OS) 功能爲止，需要經歷幾個階段的啓動過程。

爲簡便起見，筆者將討論範圍限定在具有預設安裝的典型 PC 平臺上。

典型的啓動過程像是一個四級的火箭。每一級火箭將系統控制權交給下一級。

- 節 3.1.1
- 節 3.1.2
- 節 3.1.3
- 節 3.1.4

當然，這些階段可以有不同的調配。比如，你編譯了自己的核心，則可能會跳過迷你 Debian 系統的步驟。因此，在讀者親自確認之前，請勿假定自己系統的情況也是如此。

3.1.1 第一階段：UEFI

Unified Extensible Firmware Interface (UEFI) 統一可擴充套件韌體介面 定義了啟動管理器作為 UEFI 規範的一部分。當一個計算機開啟電源，啟動管理器是啟動流程的第一階段，它檢查啟動配置並基於啟動配置的設定，執行特定的作業系統引導載入程式或作業系統核心 (通常是引導載入程式)。啟動配置透過變數儲存在 NVRAM，變數包括指示作業系統引導載入程式或作業系統核心的檔案系統路徑的變數。

EFI system partition (ESP) EFI 系統分割槽 是一個數據儲存裝置分割槽，在計算機裡用來遵照 UEFI 規範。當計算機開啟電源時，由 UEFI 韌體來訪問，它儲存了 UEFI 應用程式和這些應用程式執行所需要的檔案，包括作業系統的引導載入程式。(在老的 PC 系統，存放在 MBR 裡的 BIOS 可以用來代替。)

3.1.2 第二階段：引載加載程序

引導載入程式是啟動過程的第二階段，由 UEFI 啟動。引導載入程式將系統核心映像和 initrd 映像載入到記憶體並將控制權交給它們。initrd 映像是根檔案系統映像，其支援程度依賴於所使用的引導載入程式。

Debian 系統通常使用 Linux 核心作為預設的系統核心。當前的 5.x Linux 核心的 initrd 映像在技術上是 initramfs（初始 RAM 檔案系統）映像。

有許多引導載入程式和配置選項存在。

軟體包	流行度	大小	**initrd**	引導加載程序	說明
grub-efi-amd64	I:331	184	支援	GRUB UEFI	可智慧識別磁碟分割槽和檔案系統，例如 vfat、ext4⋯（UEFI）
grub-pc	V:21, I:641	557	支援	GRUB 第 2 版	可智慧識別磁碟分割槽和檔案系統，例如 vfat、ext4⋯（BIOS）
grub-rescue-pc	V:0, I:0	6625	支援	GRUB 第 2 版	此爲 GRUB 第 2 版的可引導修復映像（CD 和軟盤）（PC / BIOS 版本）
syslinux	V:3, I:37	344	支援	Isolinux	可識別 ISO9660 檔案系統。引導 CD 使用此項。
syslinux	V:3, I:37	344	支援	Syslinux	可識別 MSDOS 檔案系統（FAT）。引導軟盤使用此項。
loadlin	V:0, I:0	90	支援	Loadlin	新系統從 FreeDOS 或 MSDOS 中啓動。
mbr	V:0, I:4	50	不支援	Neil Turton 的 MBR	此爲取代 MSDOS MBR 的自由軟體。只可識別硬盤分區。

Table 3.1: 引導加載程序列表

警告
假如沒有從 grub-rescue-pc 軟體包中的映像製作出來的可引導修復盤（隨身碟、CD 或軟盤），請勿玩弄引導載入程式。即使硬碟上沒有可正常工作的引導載入程式，可引導修復盤也能引導你的系統。

對於 UEFI 系統，GRUB2 首先讀取 ESP 分割槽，使用”/boot/efi/EFI/debian/grub.cfg” 裡面 search.fs_uuid 指定的 UUID 來確定 GRUB2 選單配置檔案”/boot/grub/grub.cfg” 所在的分割槽。

GRUB2 選單配置檔案的關鍵部分看起來像：

```
menuentry 'Debian GNU/Linux' ... {
        load_video
        insmod gzio
        insmod part_gpt
        insmod ext2
        search --no-floppy --fs-uuid --set=root fe3e1db5-6454-46d6-a14c-071208ebe4b1
        echo    'Loading Linux 5.10.0-6-amd64 ...'
        linux   /boot/vmlinuz-5.10.0-6-amd64 root=UUID=fe3e1db5-6454-46d6-a14c-071208ebe4b1 ↩
             ro quiet
        echo    'Loading initial ramdisk ...'
        initrd  /boot/initrd.img-5.10.0-6-amd64
}
```

對於這部分的 /boot/grub/grub.cfg，這個選單條目的意義如下。

提示
透過刪除”/boot/grub/grub.cfg” 裡面的 quiet ，你能夠檢視核心啟動日誌資訊。為固化這個修改，請編輯”/etc/default/grub” 裡的”GRUB_CMDLINE_LINUX_DEFAULT=”quiet”” 行。

設定	值
GRUB2 模組載入	`gzio`, `part_gpt`, `ext2`
使用的根檔案系統分割槽	由 `UUID=fe3e1db5-6454-46d6-a14c-071208ebe4b1` 指定的分割槽標識
核心映象檔案在根檔案系統中的路徑	`/boot/vmlinuz-5.10.0-6-amd64`
使用的核心啟動引數	”`root=UUID=fe3e1db5-6454-46d6-a14c-071208ebe4b1 ro quiet`”
initrd 映象檔案在根檔案系統中的路徑	`/boot/initrd.img-5.10.0-6-amd64`

Table 3.2: `/boot/grub/grub.cfg` 檔案上面部分選單條目意義

提示
透過設定在“`/etc/default/grub`”的 `GRUB_BACKGROUND` 變數指向到影象檔案，或者把影象檔案本身放入“`/boot/grub/`”，你能夠定製 GRUB 的啟動影象。

參見“`info grub`”及 `grub-install`(8)。

3.1.3 第三階段：迷你 Debian 系統

迷你 Debian 系統是啓動流程的第三階段，由引導加載程序啓動。它會在記憶體中運行系統核心和根檔案系統。這是啓動流程的一個可選準備階段。

注
“迷你 Debian 系統”是筆者自創的術語，用於在本文檔中描述啓動流程的第三個階段。這個系統通常被稱爲 initrd 或 initramfs 系統。記憶體中類似的系統在 Debian 安裝程序中使用。

`/init` 程式是記憶體中的根檔案系統上執行的第一個程式。這個程式在使用者空間把核心初始化，並把控制權交給下一階段。迷你 Debian 系統能夠在主引導流程之前新增核心模組或以加密形式掛載根檔案系統，使引導流程更加靈活。

- 如果 initramfs 是由 `initramfs-tools` 建立，則”`/init`” 程式是一個 shell 指令碼程式。
  - 通過給核心添加“`break=init`” 等啓動參數，你可以中斷這部分啓動流程以獲得 root shell。更多中斷條件請參見”`/init`“腳本。這個 shell 環境已足夠成熟，你可通過它很好地檢查機器的硬體。
  - 迷你 Debian 系統中可用的指令是精簡過的，且主要由一個稱爲 `busybox`(1) 的 GNU 工具提供。
- 如果 initramfs 是由 `dracut` 建立，則”`/init`” 程式是一個二進位制 `systemd` 程式。
  - 迷你 Debian 系統中可用的命令是一個精簡過的 `systemd`(1) 環境。

注意
當在一個只讀的根檔案系統上時，使用 `mount` 指令需要添加 `-n` 選項。

3.1.4 第四階段：常規 Debian 系統

常規 Debian 系統是啓動流程的第四階段，由迷你 Debian 系統啓動。迷你 Debian 系統的核心在此環境下繼續運行。根檔案系統將由記憶體切換到實際的硬盤檔案系統上。

init 程序是系統執行的第一個程序（PID=1），它啓動其它各種程序以完成主引導流程。init 程序的預設路徑是”`/usr/sbin/init`“，但可通過核心啓動參數修改，例如”`init=/path/to/init_program`”。

在 Debian 8 jessie（2015 年釋出）版本後，”`/usr/sbin/init`” 是一個到”`/lib/systemd/systemd`” 的符號連結。

提示
你的系統中實際使用的 init 指令可以使用“ps --pid 1 -f”命令確認。

軟體包	流行度	大小	說明
systemd	V:863, I:965	11166	基於事件且支援並發的 init(8) 後台行程（可替代 sysvinit）
cloud-init	V:3, I:5	2870	雲實例架構的初始化系統
systemd-sysv	V:835, I:963	78	systemd 需用的用以代替 sysvinit 的手冊頁和符號連結
init-system-helpers	V:692, I:972	130	在 sysvinit 和 systemd 之間進行轉換的幫助工具
initscripts	V:34, I:139	198	用於初始化和關閉系統的腳本
sysvinit-core	V:5, I:6	373	類 System V 的 init(8) 工具
sysv-rc	V:70, I:151	88	類 System V 的運行級別修改機制
sysvinit-utils	V:895, I:999	102	類 System V 的實用工具（startpar(8)，bootlogd(8)，……）
lsb-base	V:658, I:702	12	Linux 標準規範 3.2 版的 init 腳本功能
insserv	V:92, I:150	132	利用 LSB init.d 腳本依賴性來組織啓動步驟的工具
kexec-tools	V:1, I:6	316	用於 kexec(8) 重啟（熱啟動）的 kexec 工具
systemd-bootchart	V:0, I:0	131	啟動流程效能分析器
mingetty	V:0, I:2	36	僅包含控制檯的 getty(8)
mgetty	V:0, I:0	315	可智慧調製解調的 getty(8) 替代品

Table 3.3: Debian 系統啓動工具列表

提示
有關啟動流程加速的最新資訊，請參見 Debian 維基：啟動流程加速詞條。

3.2 Systemd

3.2.1 Systemd 初始化

當 Debian 系統啟動，/usr/sbin/init 符號連結到的 /usr/lib/systemd 作為初始系統程序 (PID=1) 啟動，該程序由 root (UID=0) 所有。參見 systemd(1)。

systemd 初始化程序基於單元配置檔案 (參見 systemd.unit(5)) 來並行派生程序，這些單元配置檔案使用宣告樣式來書寫,代替之前的類 SysV 的過程樣式。這些單元配置檔案從下面的一系列路徑來載入 (參見 systemd-system.conf(5))：

派生的程序被放在一個單獨的 Linux control groups，在單元后命名，它們屬於一個私有的 systemd 層級結構 (參見 cgroups 和節 4.7.5)。

系統模式單元從 systemd.unit(5) 描述中的”System Unit Search Path” 載入。主要部分是按照下列優先權順序：

- ”/etc/systemd/system/*”: 管理員建立的系統單元檔案
- ”/run/systemd/system/*”: 執行時單元檔案
- ”/lib/systemd/system/*”: 發行版軟體包管理器安裝的系統單元檔案

他們的相互依賴關係透過”Wants=”, ”Requires=”, ”Before=”, ”After=”, …等指示來配置,(參見 systemd.unit(5) 裡的”MAPPING OF UNIT PROPERTIES TO THEIR INVERSES”)。資源控制也是被定義 (參見 systemd.resource-control(

根據單元配置檔案的字尾來區分它們的型別：

- ***.service** 描述由 systemd 控制和監管的程序. 參見 systemd.service(5).
- ***.device** 描述在 sysfs(5) 裡面作為 udev(7) 裝置樹展示的裝置。參見 systemd.device(5).
- ***.mount** 描述由 systemd 控制和監管的檔案系統掛載點。參見 systemd.mount(5).
- ***.automount** 描述由 systemd 控制和監管的檔案系統自動掛載點。參見 systemd.automount(5).
- ***.swap** 描述由 systemd 控制和監管的 swap 檔案或裝置。參見 systemd.swap(5).
- ***.path** 描述被 systemd 監控的路徑，用於基於路徑的活動。參見 systemd.path(5).
- ***.socket** 描述被 systemd 控制和監管的套接字，用於基於套接字的活動。參見 systemd.socket(5).
- ***.timer** 描述被 systemd 控制和監管的計時器，用於基於時間的活動。參見 systemd.timer(5).
- ***.slice** 管理 cgroups(7) 的資源。參見 systemd.slice(5).
- ***.scope** 使用 systemd 的匯流排介面來程式化的建立，用以管理一系列系統程序。參見 systemd.scope(5).
- ***.target** 把其它單元配置檔案分組，在啟動的時候，來建立同步點。參見 systemd.target(5).

系統啟動時 (即,init),systemd 程序會嘗試啟動"/lib/systemd/system/default.target (通常是到"graphical.ta
的符號連結)。首先,一些特殊的 target 單元 (參見 systemd.special(7)),比如"local-fs.target"、"swap.target"
和"cryptsetup.target" 會被引入以掛載檔案系統。之後，其它 target 單元也會根據單元依賴關係而被引入。詳細
情況，請閱讀 bootup(7)。

systemd 提供向後相容的功能。在"/etc/init.d/rc[0123456S].d/[KS]*name*" 裡面的 SysV 風格的啟動指令碼
仍然會被分析；telinit(8) 會被轉換為 systemd 的單元活動請求。

注意
模擬的執行級別 2 到 4 全部被符號連結到了相同的“multi-user.target”。

3.2.2 Systemd 登入

當一個使用者透過 gdm3(8)、sshd(8) 等登入到 Debian 系統,/lib/systemd/system --user 作為使用者服務管理
器程序啟動，並由相應的使用者所有。參見 systemd(1)。

systemd 初始化程序基於單元配置檔案 (參見 systemd.unit(5)) 來並行派生程序，這些單元配置檔案使用宣告樣式
來書寫,代替之前的類 SysV 的過程樣式。這些單元配置檔案從下面的一系列路徑來載入 (參見 systemd-system.conf(5))
。

使用者模式單元從 systemd.unit(5) 描述中的"User Unit Search Path" 載入。主要部分是按照下列優先權順序:

- "~/.config/systemd/user/*": 使用者配置單元檔案
- "/etc/systemd/user/*": 管理員建立的使用者單元檔案
- "/run/systemd/user/*": 使用者執行時單元檔案
- "/lib/systemd/user/*": 發行版軟體包管理器安裝的使用者單元檔案

這些是和節 3.2.1 用同樣的方式管理。

3.3 核心訊息

在控制檯上顯示的核心錯誤資訊，能夠透過設定他們的閾值水平來配置。

```
# dmesg -n3
```

錯誤級別值	錯誤級別名稱	說明
0	KERN_EMERG	系統不可用
1	KERN_ALERT	行為必須被立即採取
2	KERN_CRIT	危險條件
3	KERN_ERR	錯誤條件
4	KERN_WARNING	警告條件
5	KERN_NOTICE	普通但重要的條件
6	KERN_INFO	資訊提示
7	KERN_DEBUG	debug 級別的資訊

Table 3.4: 核心錯誤級別表

3.4 系統訊息

在 systemd 下, 核心和系統的資訊都透過日誌服務 systemd-journald.service (又名 journald) 來記錄，放在”/var/log/journal” 下的不變的二進位制資料，或放在”/run/log/journal/” 下的變化的二進位制資料. 這些二進位制日誌資料，可以透過 journalctl(1) 命令來訪問。例如，你可以顯示從最後一次啟動以來的日誌，按如下所示：

```
$ journalctl -b
```

操作	命令片段
檢視從最後一次啟動開始的系統服務和核心日誌	”journalctl -b --system”
檢視從最後一次啟動開始的當前使用者的服務日誌	”journalctl -b --user”
檢視從最後一次啟動開始的”$unit” 工作日誌	”journalctl -b -u $unit”
檢視從最後一次啟動開始的”$unit” 的工作日誌 (”tail -f” 式樣)	”journalctl -b -u $unit -f”

Table 3.5: 典型的 journalctl 命令片段列表

在 systemd 下，系統日誌工具 rsyslogd(8) 可以被解除安裝。如果安裝了它，它會改變它的行為來讀取易失性二進位制日誌資料 (代替在 systemd 之前預設的”/dev/log”) 並建立傳統的永久性 ASCII 系統日誌資料。”/etc/default/rsyslog” 和”/etc/rsyslog.conf” 能夠自定義日誌檔案和螢幕顯示。參見 rsyslogd(8) 和 rsyslog.conf(5)，也可以參見節 9.3.2。

3.5 系統管理

systemd 不僅僅提供系統初始化，還用 systemctl(1) 命令提供通用的系統管理操作。

這裡, 上面例子中的”$unit”，可以是一個單元名 (字尾 .service 和 .target 是可選的)，或者，在很多情況下，也可以是匹配的多個單元 (shell 式樣的全域性萬用字元”*”, ”?”, ”[]”，透過使用 fnmatch(3) ，來匹配目前在記憶體中的所有單元的基本名稱).

上面列子的系統狀態改變命令，通常是透過”sudo” 來處理，用以獲得需要的系統管理許可權。

”systemctl status $unit|$PID|$device” 的輸出使用有顏色的點 (”●”) 來概述單元狀態，讓人看一眼就知道。

- 白色的”●” 表示一個” 不活動” 或” 變為不活動中” 的狀態。
- 紅色的”●” 表示“失敗”或者“錯誤”狀態。
- 綠色”●” 表示“活動”、“重新載入中”或“啟用中”狀態。

操作	命令片段
列出所有存在的單元型別	”systemctl list-units --type=help”
列出記憶體中所有 target 單元	”systemctl list-units --type=target”
列出記憶體中所有 service 單元	”systemctl list-units --type=service”
列出記憶體中所有 device 單元	”systemctl list-units --type=device”
列出記憶體中所有 mount 單元	”systemctl list-units --type=mount”
列出記憶體中所有 socket 單元	”systemctl list-sockets”
列出記憶體中所有 timer 單元	”systemctl list-timers”
啟動”$unit”	”systemctl start $unit”
停止”$unit”	”systemctl stop $unit”
重新載入服務相關的配置	”systemctl reload $unit”
停止和啟動所有”$unit”	”systemctl restart $unit”
啟動”$unit” 並停止所有其它的	”systemctl isolate $unit”
轉換到” 圖形” (圖形介面系統)	”systemctl isolate graphical”
轉換到” 多使用者” (命令列系統)	”systemctl isolate multi-user”
轉換到” 應急模式” (單使用者命令列系統)	”systemctl isolate rescue”
向”$unit” 傳送殺死訊號	”systemctl kill $unit”
檢查”$unit” 服務是否是活動的	”systemctl is-active $unit”
檢查”$unit” 服務是否是失敗的	”systemctl is-failed $unit”
檢查”$unit\|$PID|device” 的狀態	”systemctl status $unit\|$PID|$device”
顯示”$unit\|$job” 的屬性	”systemctl show $unit\|$job”
重設失敗的”$unit”	”systemctl reset-failed $unit”
列出所有單元服務的依賴性	”systemctl list-dependencies --all”
列出安裝在系統上的單元檔案	”systemctl list-unit-files”
啟用”$unit” (增加符號連結)	”systemctl enable $unit”
停用”$unit” (刪除符號連結)	”systemctl disable $unit”
取消遮掩”$unit” (刪除到”/dev/null” 的符號連結)	”systemctl unmask $unit”
遮掩”$unit” (增加到”/dev/null” 的符號連結)	”systemctl mask $unit”
獲取預設的 target 設定	”systemctl get-default”
設定預設 target 為”graphical” (圖形系統)	”systemctl set-default graphical”
設定預設的 target 為”multi-user” (命令列系統)	”systemctl set-default multi-user”
顯示工作環境變數	”systemctl show-environment”
設定環境變數”variable” 的值為”value”	”systemctl set-environment variable=value”
取消環境變數”variable” 的設定	”systemctl unset-environment variable”
重新載入所有單元檔案和後臺守護程序 (daemon)	”systemctl daemon-reload”
關閉系統	”systemctl poweroff”
關閉和重啟系統	”systemctl reboot”
掛起系統	”systemctl suspend”
休眠系統	”systemctl hibernate”

Table 3.6: 典型的 systemctl 命令片段列表

3.6 其它系統監控

這裡是 systemd 下其它零星的監控命令列表。請閱讀包括 cgroups(7) 在內的相關的 man 手冊頁。

操作	命令片段
顯示每一個初始化步驟所消耗的時間	"systemd-analyze time"
列出所有單元的初始化時間	"systemd-analyze blame"
載入"$unit" 檔案並檢測錯誤	"systemd-analyze verify $unit"
簡潔的顯示使用者呼叫會話的執行時狀態資訊	"loginctl user-status"
簡潔的顯示呼叫會話的執行時狀態資訊	"loginctl session-status"
跟蹤 cgroups 的啟動過程	"systemd-cgls"
跟蹤 cgroups 的啟動過程	"ps xawf -eo pid,user,cgroup,args"
跟蹤 cgroups 的啟動過程	讀取"/sys/fs/cgroup/" 下的 sysfs

Table 3.7: systemd 下其它零星監控命令列表

3.7 系統配置

3.7.1 主機名

核心維護系統主機名。在啟動的時候，透過 systemd-hostnamed.service 啟動的系統單位設定系統的主機名，此主機名儲存在"/etc/hostname"。這個檔案應該只包含系統主機名，而不是全稱域名。

不帶參數執行 hostname(1) 指令可以打印出當前的主機名。

3.7.2 檔案系統

硬碟和網路檔案系統的掛載選項可以在"/etc/fstab" 中設定，參見 fstab(5) 和節 9.6.7。

加密檔案系統的配置設定在“/etc/crypttab”中。參見 crypttab(5)

軟 RAID 的配置 mdadm(8) 設定在"/etc/mdadm/mdadm.conf". 參見 mdadm.conf(5).

警告
每次啟動的時候，在掛載了所有檔案系統以後，"/tmp", "/var/lock", 和"/var/run" 中的臨時檔案會被清空。

3.7.3 網路介面初始化

對於使用 systemd 的現代 Debian 桌面系統,網路介面通常由兩個服務進行初始化:lo 介面通常在“networking.service”處理，而其它介面則由“NetworkManager.service”處理。

參見章 5 來獲取怎樣來配置它們的資訊。

3.7.4 雲系統初始化

雲系統例項可以由 ”Debian Official Cloud Images” 或類似的映象啟動。對於這樣的系統例項，主機名、檔案系統、網路、語言環境、SSH 金鑰、使用者和組等個性化資訊，可以使用 `cloud-init` 和 `netplan.io` 軟體包提供的功能來配置，利用多個數據源，放在原始系統映象裡面的檔案和在啟動過程中提供的外部資料。這些軟體包使用 YAML 資料來宣告系統配置。

更多資訊參見 ”Cloud Computing with Debian and its descendants”, ”Cloud-init documentation” 和節 5.4。

3.7.5 調整 sshd 服務的個性化例子

使用預設安裝，透過 `systemd` 啟動的過程中，在 `network.target` 啟動後，很多網路服務 (參見章 6) 作為後臺守護程序 (daemon) 啟動。”`sshd`” 也不列外。讓我們修改為按需啟動”`sshd`” 作為一個定製化的例子。

首先，停用系統安裝的服務單元。

```
$ sudo systemctl stop sshd.service
$ sudo systemctl mask sshd.service
```

傳統 Unix 服務的按需套接字啟用 (on-demand socket activation) 系統由 `inetd` (或 `xinetd`) 超級服務來提供。在 `systemd` 下, 相同功能能夠透過增加 ***.socket** 和 ***.service** 單元配置檔案來啟用。

`sshd.socket` 用來定義一個監聽的套接字

```
[Unit]
Description=SSH Socket for Per-Connection Servers

[Socket]
ListenStream=22
Accept=yes

[Install]
WantedBy=sockets.target
```

`sshd@.service` 作為 `sshd.socket` 匹配的服務檔案

```
[Unit]
Description=SSH Per-Connection Server

[Service]
ExecStart=-/usr/sbin/sshd -i
StandardInput=socket
```

然後重新載入。

```
$ sudo systemctl daemon-reload
```

3.8 udev 系統

從 Linux 核心 2.6 版開始，udev 系統 提供了自動硬體發現和初始化機制。(參見 udev(7)). 在核心發現每個裝置的基礎上，udev 系統使用從 sysfs 檔案系統 (參見節 1.2.12) 的資訊啟動一個使用者程序，使用 `modprobe`(8) 程式 (參見節 3.9) 載入支援它所要求的核心模組, 建立相應的裝置節點。

提示
如果由於某些理由，”`/lib/modules/`*`kernel-version`*`/modules.dep`” 沒有被 `depmod`(8) 正常生成，模組可能不會被 udev 系統按期望的方式載入。執行”`depmod -a`” 來修復它。
”`/etc/fstab`” 裡面的掛載規則, 裝置節點不必需是靜態的。你能夠使用 UUID 來掛載裝置，來代替”`/dev/sda`” 之類的裝置名. 參見節 9.6.3.

由於 udev 系統是一個正在變化的事物，我在其它文件進行了詳細描述，在這裡只提供了最少的資訊。

警告
不要嘗試用 udev 規則裡面的 RUN (在 udev(7) 提到) 長期執行程式，比如說備份指令碼。請建立一個適當的 systemd.service(5) 檔案並激活它來替代。參見節 10.2.3.2。

3.9 核心模組初始化

通過 modprobe(8) 程式新增和刪除核心模組，使我們能夠從使用者程序來調配正在執行的 Linux 核心。udev 系統 (參見節 3.8) 自動化它的呼叫來幫助核心模組初始化。

下面的非硬體模組和特殊的硬體驅動模組，需要被預先載入，把它們在"/etc/modules" 檔案裡列出 (參見 modules(5)).

- TUN/TAP 模組提供虛擬的 Point-to-Point 網路裝置 (TUN) 和虛擬的 Ethernet 乙太網路網路裝置 (TAP),
- netfilter 模組提供 netfilter 防火牆能力 (iptables(8), 節 5.7),
- watchdog timer 驅動模組。

modprobe(8) 程式的組態檔案是按 modprobe.conf(5) 的說明放在"/etc/modprobes.d/" 目錄下，(如果你想避免自動載入某些核心模組，考慮把它們作為黑名單放在"/etc/modprobes.d/blacklist" 檔案裡.)

"/lib/modules/*version*/modules.dep" 檔案由 depmod(8) 程式生成，它描述了 modprobe(8) 程式使用的模組依賴性.

注
如果你在啟動時出現模組載入問題，或者 modprobe(8) 時出現模組載入問題, "depmod -a" 可以通過重構"modules.dep" 來解決這些問題。

modinfo(8) 程式顯示 Linux 核心模組資訊。

lsmod(8) 程式以好看的格式展示"/proc/modules" 的內容, 顯示當前核心載入了哪些模組。

提示
你能夠精確識別你係統上的硬體。參見節 9.5.3.
你可以在啟動時調配硬體來啟用期望的硬體特徵。參見節 9.5.4.
你可以重新編譯核心來增加你的特殊裝置的支援。參見節 9.10.

Chapter 4

認證和訪問控制

當使用者（或程式）需要存取系統時，需要進行認證，確認身份是受信任。

警告
PAM 的調配錯誤可能會鎖住你的系統。你必須有一個準備好的救援 CD，或者設立一個替代的 boot 分割槽。為了恢復系統，你需要使用它們啟動系統並糾正錯誤。

4.1 一般的 Unix 認證

一般的 Unix 認證由 PAM（Pluggable Authentication Modules，即可插入的驗證模組）下的 pam_unix(8) 模組提供。它的 3 個重要檔案如下，其內的條目使用“:”分隔。

檔案	許可權	使用者	組	說明
/etc/passwd	-rw-r--r--	root	root	（明文的）使用者帳號資訊
/etc/shadow	-rw-r-----	root	shadow	安全加密的使用者帳號資訊
/etc/group	-rw-r--r--	root	root	組資訊

Table 4.1: pam_unix(8) 使用的 3 個重要組態檔案

“/etc/passwd”包含下列內容。

```
 ...
user1:x:1000:1000:User1 Name,,,:/home/user1:/bin/bash
user2:x:1001:1001:User2 Name,,,:/home/user2:/bin/bash
 ...
```

如 passwd(5) 中所述，這個檔案中被“:”分隔的每項含義如下。

- 登入名
- 密碼形式說明
- 數字形式的使用者 ID
- 數字形式的組 ID
- 使用者名稱或註釋欄位

- 使用者家目錄
- 可選的使用者指令直譯器

“/etc/passwd”的第二項曾經被用來儲存加密後的密碼。在引入了“/etc/shadow”後，該項被用來說明密碼形式。

內容	說明
（空）	無需密碼的帳號
x	加密後的密碼儲存在“/etc/shadow”

Table 4.2: “/etc/passwd”第二項的內容

“/etc/shadow”包含下列內容。

```
 ...
user1:$1$Xop0FYH9$IfxyQwBe9b8tiyIkt2P4F/:13262:0:99999:7:::
user2:$1$vXGZLVbS$ElyErNf/agUDsm1DehJMS/:13261:0:99999:7:::
 ...
```

如 shadow(5) 中所述，這個檔案中被“:”分隔的每項含義如下。

- 登入名
- 加密後的密碼（開頭的“1”表示使用 MD5 加密。“*”表示無法登入。）
- 最後一次修改密碼的時間，其表示從 1970 年 1 月 1 日起的天數
- 允許使用者再次修改密碼的天數間隔
- 使用者必須修改密碼的天數間隔
- 密碼失效前的天數，在此期間使用者會被警告
- 密碼失效後的天數，在次期間密碼依舊會被接受
- 帳號失效的時間，其表示從 1970 年 1 月 1 日起的天數
- …

“/etc/group”包含下列內容。

```
group1:x:20:user1,user2
```

如 group(5) 中所述，這個檔案中被“:”分隔的每項含義如下。

- 組名稱
- 加密後的密碼（不會被真正使用）
- 數字形式的組 ID
- 使用“,”分隔的使用者名稱列表

注
“/etc/gshadow”為“/etc/group”提供了與“/etc/shadow”相似的功能，但沒有被真正地使用。

注
如果"auth optional pam_group.so" 這行新增到了"/etc/pam.d/common-auth", 並且在"/etc/security/group.conf" 裡進行了設定，一個使用者的實際組就可以被動態新增。參見 pam_group(8).

注
base-passwd 軟體包包含了一份使用者和組的官方文件:"/usr/share/doc/base-passwd/users-and-groups.ht

4.2 管理帳號和密碼資訊

下面是一些管理帳號資訊的重要指令。

指令	功能
getent passwd *user_name*	瀏覽"*user_name*"的帳號資訊
getent shadow *user_name*	瀏覽使用者"*user_name*" 隱藏的帳號資訊
getent group *group_name*	瀏覽"*group_name*"的組資訊
passwd	管理帳號密碼
passwd -e	為啟用的帳號設定一次性的密碼
chage	管理密碼有效期資訊

Table 4.3: 管理帳號資訊的指令

其中的一些功能只能被 root 使用。密碼和資料的加密參見 crypt(3)。

注
在設定了 PAM 和 NSS 的系統上 (例如 Debian salsa 機器)，本地的"/etc/passwd"、"/etc/group"和"/etc/shadow"可能不會被系統啟用使用。上述的命令即使處於這種環境下依舊是有效的。

4.3 好密碼

在系統安裝時建立一個帳號或使用 passwd(1) 指令時，你應該選擇一個好密碼，它應該由 6 到 8 個字元組成，其中包含下列根據 passwd(1) 設定的每個組合中的一個或多個字元。

- 小寫字母
- 數字 0 到 9
- 標點符號

警告
密碼中不要使用可以猜到的詞。帳號名、身份證號碼、電話號碼、地址、生日、家庭成員或寵物的名字、字典單詞、簡單的字元序列 (例如"12345"或"qwerty") 等都是糟糕的選擇。

軟體包	流行度	大小	指令	功能
whois	V:25, I:257	387	mkpasswd	具備 crypt(3) 庫所有特性的前端
openssl	V:839, I:995	2294	openssl passwd	計算密碼雜湊 (OpenSSL). passwd(1ssl)

Table 4.4: 生成密碼的工具

4.4 設立加密的密碼

下面是一些用於 生成加鹽的加密密碼 的獨立工具。

4.5 PAM 和 NSS

現代的類 Unix 系統（例如 Debian 系統）提供 PAM（Pluggable Authentication Modules，插入式驗證模組）和 NSS（Name Service Switch，名稱服務切換）機制給本地系統管理員，使他們能夠調配自己的系統。它們的功能可以概括為以下幾點。

- PAM 給應用軟體提供了一個靈活的認證機制，因此涉及到了密碼資料的交換。
- NSS 提供了一個靈活的名稱服務機制，它經常被 C 標準庫使用，使例如 ls(1) 和 id(1) 這樣的程式獲得使用者和組名稱。

PAM 和 NSS 系統必須保持調配一致。

PAM 和 NSS 系統中重要的軟體包如下。

軟體包	流行度	大小	說明
libpam-modules	V:885, I:999	1006	插入式驗證模組（基礎服務）
libpam-ldap	V:0, I:6	249	允許 LDAP 介面的插入式驗證模組
libpam-cracklib	V:0, I:8	117	啟用 cracklib 支援的插入式驗證模組
libpam-systemd	V:558, I:934	625	用於 logind 註冊使用者會話的插入式驗證模組（PAM）
libpam-doc	I:0	1002	插入式驗證模組（html 和文字文件）
libc6	V:923, I:999	12987	GNU C 庫：同樣提供“名稱服務切換”服務的共享庫
glibc-doc	I:9	3502	GNU C 庫：幫助頁面
glibc-doc-reference	I:4	13841	GNU C 庫：參考手冊，有 info、pdf 和 html 格式（non-free）
libnss-mdns	I:507	141	用於解析組播 DNS 名稱的 NSS 模組
libnss-ldap	I:5	265	NSS 模組，用於使用 LDAP 作為一個名稱服務的
libnss-ldapd	I:15	129	NSS 模組，用於使用 LDAP 作為一個名稱服務的（libnss-ldap 的新 fork）

Table 4.5: PAM 和 NSS 系統中重要的軟體包

- libpam-doc 中“The Linux-PAM System Administrators' Guide”是瞭解 PAM 調配的必要文件。
- glibc-doc-reference 中的“System Databases and Name Service Switch”是瞭解 NSS 調配的重要文件。

注
你可以使用“aptitude search 'libpam-|libnss-'”指令檢視更多的相關軟體包。NSS 縮寫也可能意味著“Network Security Service，網路安全服務”，它不同於“Name Service Switch，名稱服務切換”。

注
PAM 是為每個程式初始化環境變數為系統預設值的最基礎方法。

在 systemd 下, libpam-systemd 軟體包被安裝用來管理使用者登入，透過為 logind 在 systemd 控制組層級中註冊使用者會話來實現。參見 systemd-logind(8)、 logind.conf(5) 和 pam_systemd(8)。

4.5.1 PAM 和 NSS 存取的組態檔案

下面是一些 PAM 和 NSS 存取的重要組態檔案。

組態檔案	功能
/etc/pam.d/*program_name*	為“*program_name*”程式設定 PAM 調配；參加 pam(7) 和 pam.d(5)
/etc/nsswitch.conf	為每個服務條目設定 NSS 調配。參見 nsswitch.conf(5)
/etc/nologin	通過 pam_nologin(8) 模組限制使用者登入
/etc/securetty	通過 pam_securetty(8) 模組限制 root 存取 tty
/etc/security/access.conf	通過 pam_access(8) 模組設定存取限制
/etc/security/group.conf	通過 pam_group(8) 模組設定基於組的限制
/etc/security/pam_env.conf	通過 pam_env(8) 模組設定環境變數
/etc/environment	通過帶有“readenv=1”參數的 pam_env(8) 模組設定額外的環境變數
/etc/default/locale	通過帶有“readenv=1 envfile=/etc/default/locale”參數的 pam_env(8) 模組設定語言環境值（在 Debian 系統中）
/etc/security/limits.conf	通過 pam_linits(8) 模組設定資源限制（ulimit、core 等等）
/etc/security/time.conf	通過 pam_time(8) 模組設定時間限制
/etc/systemd/logind.conf	設定 systemd 的登入管理器配置 (參見 logind.conf(5) 和 systemd-logind.service(8))

Table 4.6: PAM 和 NSS 存取的組態檔案

密碼選擇的限制是通過 PAM 模組 pam_unix(8) 和 pam_cracklib(8) 來實現的。它們可以通過各自的參數進行調配。

提示
PAM 模組在檔名中使用字尾“.so”。

4.5.2 現代的集中式系統管理

現代的集中式系統管理可以使用集中式的輕量目錄存取協議（LDAP）伺服器進行部署，從而通過網路管理許多類 Unix 和非類 Unix 系統。輕量目錄存取協議的開源實現是 OpenLDAP 軟體。

LDAP 伺服器使用帶有 PAM 和 NSS 的 libpam-ldap 和 libnss-ldap 軟體包為 Debian 系統提供帳號資訊。需要一些動作來啟用 LDAP（我沒有使用過這個設定，並且下面的資訊純粹是第二手的資訊。請在這種前提下閱讀下列內容。）。

- 你通過執行一個程式，例如獨立的 LDAP 背景程式 slapd(8)，來建立集中式的 LDAP 伺服器。
- 你在“/etc/pam.d/”目錄中的 PAM 組態檔案裡，使用“pam_ldap.so”替代預設值“pam_unix.so”。
  - Debian 使用“/etc/pam_ldap.conf”作為 libpam-ldap 的組態檔案，“/etc/pam_ldap.secret”作為儲存 root 密碼的檔案。
- 你在“/etc/nsswitch.conf”檔案中改變 NSS 調配，使用“ldap”替代預設值（“compat”或“file”）。

  - Debian 使用“/etc/libnss-ldap.conf”作為 libnss-ldap 的組態檔案。
- 為了密碼的安全，你必須讓 libpam-ldap 使用 SLL (或 TLS) 連線。
- 為了確保 LDAP 網路開銷資料的完整性，你必須讓 libpam-ldap 使用 SLL (或 TLS) 連線。
- 為了減少 LDAP 網路流量，你應該在本地執行 nscd(8) 來快取任何 LDAP 搜尋結果。

參見由 libpam-doc 軟體包提供的 pam_ldap.conf(5) 中的文件和“/usr/share/doc/libpam-doc/html/”，以及 glibc-doc 軟體包提供的“info libc 'Name Service Switch'”。

類似地，你可以使用其它方法來設定另一種集中式的系統。

- 同 Windows 系統整合使用者和組。
  - 通過 winbind 和 libpam_winbind 軟體包訪問 Windows domain 服務.
  - 參見 winbindd(8) 和 Integrating MS Windows Networks with Samba。
- 同古老的類 Unix 系統整合使用者和組。
  - 通過 nis 軟體包存取 NIS (之前叫 YP) 或 NIS+。
  - 參見 The Linux NIS(YP)/NYS/NIS+ HOWTO。

4.5.3 “為什麼 GNU su 不支援 wheel 組”

這是在舊的“info su”底部 Richard M. Stallman 所說的一句名言。別擔心：Debian 系統中當前的 su 指令使用了 PAM，這樣當在“/etc/pam.d/su”中啟用了帶有“pam_wheel.so”的行後，就能夠限制非 wheel 組的使用者 su 到 root 組的能力。

4.5.4 嚴格的密碼規則

安裝 libpam-cracklib 軟體包你能夠強制使用嚴格的密碼規則。

在一個典型的 GNOME 系統，將會安裝 libpam-gnome-keyring，”/etc/pam.d/common-password” 看起來像：

```
# here are the per-package modules (the "Primary" block)
password  requisite     pam_cracklib.so retry=3 minlen=8 difok=3
password  [success=1 default=ignore]  pam_unix.so obscure use_authtok try_first_pass  ↩
    yescrypt
# here's the fallback if no module succeeds
password  requisite     pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
password  required      pam_permit.so
# and here are more per-package modules (the "Additional" block)
password  optional  pam_gnome_keyring.so
# end of pam-auth-update config
```

4.6 安全認證

注
這裡的資訊也許不夠完全滿足你的安全需求，但這也是一個好的開始.

4.6.1 網際網路密碼安全

許多流行的傳輸層服務，互動資訊使用純文字的密碼認證。這是非常差的方式，通過公共的網際網路傳輸純文字密碼，密碼能夠被截獲到。你能夠執行這些服務，使用"Transport Layer Security" (TLS) 或它的前身, "Secure Sockets Layer" (SSL)，通過加密來使包括密碼在內的整個通訊更加安全。

不安全的服務名	埠	安全的服務名	埠
www (http)	80	https	443
smtp (郵件)	25	ssmtp (smtps)	465
ftp-data	20	ftps-data	989
ftp	21	ftps	990
telnet	23	telnets	992
imap2	143	imaps	993
pop3	110	pop3s	995
ldap	389	ldaps	636

Table 4.7: 安全和不安全的服務埠列表

加密消耗 CPU 時間。作為對 CPU 有益的替代方案，你可以保持使用純文字通訊，僅僅使用安全認證協議加密密碼，比如說：POP 使用"Authenticated Post Office Protocol" (APOP)，SMTP 和 IMAP 使用"Challenge-Response Authentication Mechanism MD5" (CRAM-MD5)。(你的郵件客戶端通過網際網路上你的郵件伺服器傳送郵件時，最近流行使用新的遞交埠 587 來代替傳統的 SMTP 埠 25，這樣可以避免在使用 CRAM-MD5 認證自己時，網路提供商阻塞 25 埠。)

4.6.2 安全 Shell

安全 Shell (SSH) 程式使用安全認證來提供不安全網路上兩個不可信任主機之間的安全加密通訊。它由 OpenSSH 客戶端, `ssh`(1), 和 OpenSSH 後臺背景程式 (daemon) , `sshd`(8) 組成.SSH 使用埠轉發特性，可以給 POP 和 X 之類的不安全的協議通訊建立隧道，使其可以在網際網路上安全傳輸。

客戶端可以使用如下方式來認證自己：基於主機的認證、公鑰認證、質疑應答認證、密碼認證。使用公鑰認證，可以實現遠端免密碼登入。參見節 6.3.

4.6.3 網際網路額外的安全方式

即使你執行 Secure Shell (SSH) 和 Point-to-point tunneling protocol (PPTP) 這樣的安全服務，在網際網路上，仍然有機會使用野蠻暴力猜測密碼攻擊進入。使用防火牆策略 (參見節 5.7)，並和下面的安全工具一起，可以提升安全形勢。

軟體包	流行度	大小	說明
`knockd`	V:0, I:2	110	小的 port-knock 後臺守護程序 (daemon) `knockd`(1) 和客戶端 `knock`(1)
`fail2ban`	V:99, I:112	2126	禁用造成多個認證錯誤的 IP
`libpam-shield`	V:0, I:0	115	把嘗試猜測密碼的遠端攻擊者關在外面

Table 4.8: 提供額外安全方式的工具列表

4.6.4 root 密碼安全

為阻止人們使用 root 許可權存取你的機器，你需要做下面的操作。

- 阻止對硬碟的物理存取
- 鎖住 UEFI/ BIOS 來阻止從可移動介質啟動

- 為 GRUB 互動式會話設定密碼
- 鎖住 GRUB 選單，禁止編輯

如果可以物理存取硬碟，則可以使用下面的步驟，相對簡單的重置密碼。

1. 將硬碟拿到一個可以設定 UEFI/BIOS 從 CD 啟動的電腦。
2. 使用緊急介質啟動系統（Debian 啟動磁碟, Knoppix CD, GRUB CD, ⋯)。
3. 用讀寫存取掛載根分割槽。
4. 編輯根分割槽的”/etc/passwd” 檔案，使 root 帳號條目的第二段為空。

對於 grub-rescue-pc ，即使用緊急介質啟動的電腦，如果有編輯 GRUB 選單條目 (參見節 3.1.2) 的許可權，在啟動時，使用下面的步驟更加簡單。

1. 使用核心參數啟動系統來修改一些事情，比如說，”root=/dev/hda6 rw init=/bin/sh”.
2. 編輯”/etc/passwd” 檔案，使 root 帳號條目的第二段為空。
3. 重啟系統。

系統的 root shell 現在可以無密碼存取了。

注
一旦某人擁有 root shell 存取許可權，他能夠存取任何內容，並可以重設系統上的任何密碼。此外，他可以使用 john 和 crack 等軟體包的暴力破解工具來比較所有使用者的密碼 (參見節 9.5.11)。被破解的密碼，可以用來和其它系統進行比較。

為避免這些相關問題，僅有的理論上的軟體解決方案是使用 dm-crypt 和 initramfs (參見節 9.9) 加密 root 分割槽 (或”/etc” 分割槽) 。這樣的話，你總是需要密碼來啟動系統。

4.7 其它的存取控制

在密碼基於認證和檔案許可權之外，系統也有其它的訪問控制。

注
參見節 9.4.16 來限制核心的安全警告金鑰（SAK）功能。

4.7.1 訪問控制列表 (ACLs)

訪問控制列表 ACL 是在節 1.2.3 裡面解釋的普通許可權的一個超集。

在現代桌面環境中,你會遇到 ACL 行為。比如,當一個已經格式化的 USB 儲存裝置自動掛載到”/media/penguin/USBSTICK” 一個普通使用者 penguin 能夠執行:

```
 $ cd /media/penguin
 $ ls -la
total 16
drwxr-x---+ 1 root    root    16 Jan 17 22:55 .
drwxr-xr-x  1 root    root    28 Sep 17 19:03 ..
drwxr-xr-x  1 penguin penguin 18 Jan  6 07:05 USBSTICK
```

在第 11 列的"+" 表示 ACL 在使用。如果沒有 ACL，一個普通使用者 penguin 應該不能夠像這樣列出目錄內容，因為 penguin 不在 root 組。你能夠按如下方式檢視 ACL：

```
 $ getfacl .
# file: .
# owner: root
# group: root
user::rwx
user:penguin:r-x
group::---
mask::r-x
other::---
```

這裡：

- "user::rwx"、"group::---" 和"other::---" 相應的為普通所有者、組和其它人的許可權。
- ACL "user:penguin:r-x" 執行普通使用者 penguin 有"r-x" 許可權。這個能夠讓"ls -la" 列出目錄內容。
- ACL "mask::r-x" 設定上層繫結許可權。

更多資訊參見"POSIX Access Control Lists on Linux"、acl(5)、getfacl(1) 和 setfacl。

4.7.2 sudo

sudo(8) 程式是為了使一個系統管理員可以給使用者受限的 root 許可權並記錄 root 活動而設計的。sudo 只需要一個普通使用者的密碼。安裝 sudo 軟體包並通過設定"/etc/sudoers"中的選項來使用它。參見"/usr/share/doc/sudo/examp 和節 1.1.12 中的調配示例。

我將 sudo 用於單使用者系統（參見節 1.1.12）是為了防止自己可能做出的愚蠢行為。就我個人而言，我認為使用 sudo 會比使用 root 帳號作業系統來得好。例如，下列指令將"*some_file*"的擁有者改變為"*my_name*"。

```
$ sudo chown my_name some_file
```

當然如果你知道 root 密碼（比如自行安裝 Debian 的使用者所做的），任何使用者賬號都可以使用"su -c"讓任何指令以 root 執行。

4.7.3 PolicyKit

PolicyKit 是在類 Unix 作業系統中控制整個系統許可權的一個作業系統元件。

新的 GUI 圖形介面程式，在設計的時候，不是作為特權程序來執行。它們通過 PolicyKit 來和特權程序通訊，執行管理操作。

在 Debian 系統中，PolicyKit 限制了屬於 sudo 組的使用者帳號的這種操作。

參見 polkit(8)。

4.7.4 限制存取某些服務端的服務

對系統安全而言，儘可能的禁用服務程式，是一個好的主意。網路服務是危險的。有不使用的服務，不管是直接由後臺背景程式（daemon）啟用，還是通過super-server 程式啟用，都被認為是安全風險。

許多程式, 比如說 sshd(8), 使用基於 PAM 的存取控制。也還有許多方式來限制存取一些服務端的程式。

- 組態檔案: "/etc/default/*program_name*"
- 後臺守護程序（daemon）的 Systemd 服務單元配置

- PAM (Pluggable Authentication Modules)
- super-server 使用"/etc/inetd.conf"
- TCP wrapper 使用"/etc/hosts.deny" 和"/etc/hosts.allow" , tcpd(8)
- Sun RPC" 使用 /etc/rpc.conf"
- atd(8) 使用"/etc/at.allow" 和"/etc/at.deny"
- crontab(1) 使用"/etc/cron.allow" 和"/etc/cron.deny"
- Network firewall 或netfilter 框架

參見節 3.5、節 4.5.1 和節 5.7。

提示
NFS 和其它基於 RPC 的程式，需要啟用 Sun RPC 服務。

提示
如果你遠端存取最新的 Debian 系統有問題，看下在"/etc/hosts.deny" 裡是否存在"ALL: PARANOID" 這樣討厭的調配，請把它註釋掉。(但是你必須注意這種行為所帶來的安全風險。)

4.7.5 Linux 安全特性

Linux 核心已經發展和支援在傳統的 UNIX 實現裡面沒有的安全特徵。

Linux 支援 擴充套件屬性，擴充套件了傳統的 UNIX 屬性 (參見 xattr(7))。

Linux 把傳統的超級使用者相關的特權分開到不同的單元，被稱為 capabilities(7)，它能夠獨立的啟用和停用。從 2.2 版本核心開始，Capabilities 是一個執行緒獨立的屬性。

Linux Security Module (LSM) 安全模組框架 提供了一個多方面的安全檢查機制，和新的核心擴充套件關聯。例如：

- AppArmor
- Security-Enhanced Linux (SELinux)
- Smack (Simplified Mandatory Access Control Kernel)
- Tomoyo Linux

這些擴充套件緊縮的權力模型比普通的類 Unix 安全模型策略更加嚴格，甚至 root 的權力也被限制。建議你閱讀 kernel.org 上的 Linux 安全模組 (LSM) 框架文件。

Linux 的 namespaces 封裝了一個全域性系統資源到一個抽象的概念，全域性系統資源在 namespace 內對程序可見，並且 namespace 有它們自己的全域性資源隔離例項。對其它程序全域性資源的可見性的改變是，同一個 namespace 的成員可見，但是對非同一個 namespace 的其它程序不可見。從核心 5.6 版本起，有 8 種 namespaces (參見 namespaces(7), unshare(1), nsenter(1))。

在 Debian 11 Bullseye (2021) 中, Debian 使用 unified cgroup hierarchy (統一 cgroup 層級架構) (亦稱為 cgroups-v2)。

namespaces 同 cgroups 一起來隔離它們的程序，允許資源控制的使用示例是：

- Systemd。參見節 3.2.1。
- 沙盒環境。參見節 7.6。
- Linux 容器，比如 Docker、LXC。參見節 9.11。

這些功能不能夠透過節 4.1 實現。這些高階話題大部分超出了本介紹文件的範圍。

Chapter 5

網絡設置

提示
關於 Debian 專屬的網絡手冊，請查看Debian 管理員手冊—網絡調配。

提示
systemd環境下，可以用networkd來調配網絡。請參考 `systemd-networkd`(8)。

5.1 基本網絡架構

讓我們來回顧一下現代 Debian 作業系統中的基本網絡架構。

5.1.1 主機名解析

主機名解析，目前也是由 NSS (名字服務轉換 Name Service Switch) 機制來支援。這個解析的流程如下。

1. "`/etc/nsswitch.conf`" 檔案裡的"`hosts: files dns`"這段規定主機名解析順序。(代替"`/etc/host.conf`" 檔案裡的"`order`" 這段原有的功能。)
2. `files` 方式首先被呼叫。如果主機名在"`/etc/hosts`" 檔案裡面發現，則回傳所有有效地址並退出。("`/etc/host.conf`" 檔案包含"`multi on`".)
3. `dns` 方式被呼叫。如果主機名通過查詢"`/etc/resolv.conf`" 檔案裡面寫的 網際網路域名系統 Domain Name System (DNS) 來找到，則回傳所有有效地址並退出。

一個典型的工作站在安裝時就會設定主機名，例如"*host_name*" 和設定為空字串的可選域名。這樣，"`/etc/hosts`" 看起來像如下：

```
127.0.0.1 localhost
127.0.1.1 host_name

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

軟體包	流行度	大小	類型	說明
network-manager	V:388, I:455	15414	調配:: NM	NetworkManager（守衛行程）：自動管理網絡
network-manager-gnome	V:118, I:367	5583	調配:: NM	NetworkManager（GNOME 前端）
netplan.io	V:1, I:5	250	配置:: NM+networkd	Netplan (生成器): 統一的，宣告網路介面到 NetworkManager 和 systemd-networkd 後端
ifupdown	V:595, I:980	199	調配:: ifupdown	用來啓動/關閉網絡的標準工具（Debian 特有）
isc-dhcp-client	V:218, I:981	2866	調配:: 底層	DHCP 客戶端
pppoeconf	V:0, I:5	186	調配:: 輔助	調配助手，以便於使用 PPPoE 連接
wpasupplicant	V:348, I:509	3862	調配:: 輔助	WPA 和 WPA2 客戶端支援（IEEE 802.11i)
wpagui	V:0, I:1	774	調配:: 輔助	wpa_supplicant Qt 圖形界面客戶端
wireless-tools	V:176, I:243	293	調配:: 輔助	操控 Linux 無線擴展的工具
iw	V:35, I:471	302	調配:: 輔助	配置 Linux 無線裝置的工具
iproute2	V:725, I:971	3606	調配:: iproute2	iproute2, IPv6 和其他高級網絡調配：ip(8),tc(8) 等等
iptables	V:312, I:730	2414	調配:: Netfilter	封包過濾和網絡地址轉換管理工具（Netfilter）
nftables	V:106, I:686	182	調配:: Netfilter	封包過濾和網路地址轉換管理工具（Netfilter）({ip,ip6,arp,eb}tables 的後續替代者)
iputils-ping	V:195, I:997	120	測試	測試能否連接遠程主機，通過主機名或IP 地址（iproute2）
iputils-arping	V:3, I:38	49	測試	測試能否連接遠程主機，通過ARP地址
iputils-tracepath	V:2, I:31	45	測試	跟蹤存取遠程主機的路徑
ethtool	V:95, I:268	739	測試	顯示或更改以太網設備的設定
mtr-tiny	V:5, I:47	156	測試:: 底層	追蹤連接遠程主機的路徑（文本界面）
mtr	V:4, I:42	209	測試:: 底層	追蹤連接遠程主機的路徑（文本界面和 GTK 界面）
gnome-nettool	V:1, I:18	2492	測試:: 底層	獲得常見網絡資訊的工具（GNOME)
nmap	V:25, I:201	4498	測試:: 底層	網絡映射/端口掃描（Nmap，控制檯）
tcpdump	V:16, I:177	1340	測試:: 底層	網絡流量分析（Tcpdump，控制檯）
wireshark	I:45	10417	測試:: 底層	網絡流量分析（Wireshark,GTK）
tshark	V:2, I:25	400	測試:: 底層	網絡流量分析 (控制檯）
tcptrace	V:0, I:2	401	測試:: 底層	根據 tcpdump 的輸出生成的連接數據統計
snort	V:0, I:0	2203	測試:: 底層	靈活的網絡入侵偵測系統（Snort）
ntopng	V:0, I:1	15904	測試:: 底層	在網頁瀏覽器中展示網絡流量
dnsutils	V:17, I:289	275	測試:: 底層	BIND軟體包提供的網絡客戶端程序：nslookup(8),nsupdate(8),dig(8)
dlint	V:0, I:3	53	測試:: 底層	利用域名服務器查詢來查看DNS域資訊
dnstracer	V:0, I:1	59	測試:: 底層	跟蹤DNS查詢直至源頭

Table 5.1: 網絡調配工具一覽表

每一行由 IP 地址 開始，接下來是相關聯的主機名.

在這個例子的第二行 `127.0.1.1` IP 地址也許不會在其它類 Unix 系統發現。Debian Installer 為沒有永久 IP 地址的系統建立這個條目，作為某些軟體（如 GNOME）的一個變通方法，見文件 bug #719621.

host_name 匹配在"`/etc/hostname`" 裡定義的主機名。(參見節 3.7.1)。

對於有永久 IP 地址的系統，這個永久 IP 地址應當代替這裡的 `127.0.1.1`。

對於有永久 IP 地址和有 域名系統 Domain Name System (DNS)提供完全資格域名 fully qualified domain name (FQDN) 的系統，規範名 *host_name.domain_name* 應當被用來代替 *host_name*.

如果 `resolvconf` 軟體包沒有安裝，"`/etc/resolv.conf`" 是一個靜態檔案。如果安裝了，它是一個符號連結。此外，它包含有解析策略的初始化資訊。如 DNS 是 IP="`192.168.11.1`", 則包含如下。

```
nameserver 192.168.11.1
```

`resolvconf` 軟體包使這個"`/etc/resolv.conf`" 檔案成為一個符號連結，並通過鉤子指令碼自動管理其內容。

對於典型 adhoc 區域網環境下的 PC 工作站，除了基本的 `files` 和 `dns` 方式之外，主機名還能夠透過 Multicast DNS (mDNS) 進行解析。

- Avahi 提供 Debian 下的組播 DNS 發現框架。
- 它和 Apple Bonjour / Apple Rendezvous 相當.
- `libnss-mdns` 外掛包提供 mDNS 的主機名解析，GNU C 庫 (glibc) 的 GNU 名字服務轉換 Name Service Switch (NSS) 功能支援 mDNS。
- "`/etc/nsswitch.conf`" 檔案應當有像"`hosts: files mdns4_minimal [NOTFOUND=return] dns`" 這樣的一段 (其它配置參見 `/usr/share/doc/libnss-mdns/README.Debian`)。
- 一個使用 ".local" pseudo-top-level domain 字尾的主機名解析，是用 IPv4 地址"`224.0.0.251`" 或 IPv6 地址"`FF02::FB`" 傳送一個組播 UDP 包的 mDNS 查詢資訊。

注
域名系統 Domain Name System 中的擴充通用頂級域名 expansion of generic Top-Level Domains (gTLD) 還在進行中。在區域網內，選擇一個域名時，請提防名字衝突 name collision。

注
使用軟體包，比如 `libnss-resolve` 聯合 `systemd-resolved`，或者 `libnss-myhostname`，或者 `libnss-mymachine`, 並在"`/etc/nsswitch.conf`" 檔案裡面的"`hosts`" 行裡相應列出 mymachines 或 myhostname 關鍵字，這將忽略我們上面討論的傳統網路配置。更多資訊參見 `nss-resolve`(8)、`systemd-resolved`(8)、`nss-myhostname`(8) 和 `nss-mymachines`(8)。

5.1.2 網路介面名稱

systemd 使用"`enp0s25`" 之類的"可預測網路介面名稱"。

5.1.3 區域網網路地址範圍

讓我們重新提醒下在 rfc1918 裡規定的區域網 local area networks (LANs)IPv4 32 位地址在各類地址的保留範圍. 這些地址保證不會與因特網上專有的地址衝突。

注
IP 地址書寫中有冒號的是 IPv6 地址，例如，"`::1`" 是 `localhost` 本地主機。

類別	網路地址	子網掩碼	子網掩碼/位數	子網數
A	10.x.x.x	255.0.0.0	/8	1
B	172.16.x.x —172.31.x.x	255.255.0.0	/16	16
C	192.168.0.x —192.168.255.x	255.255.255.0	/24	256

Table 5.2: 網路地址範圍列表

注
如果這些地址分配到一個主機，那麼這個主機一定不能夠直接存取網際網路，必須通過一個作為閘道器的代理服務或通過 網路地址轉換 Network Address Translation (NAT). 消費區域網環境，寬頻路由器通常使用 NAT。

5.1.4 網路裝置支援

儘管 Debian 系統支援大多數硬體裝置，但依舊有一些網路裝置需要 DFSG non-free 韌體來支援它們。參見節 9.10.5。

5.2 現代的桌面網路調配

對於使用 `systemd` 的現代 Debian 桌面系統,網路介面通常由兩個服務進行初始化:`lo` 介面通常在“`networking.service`”處理，而其它介面則由“`NetworkManager.service`”處理。

Debian 可以透過後臺守護程序（daemon）管理軟體來管理網路連線，例如 NetworkManager (NM)（network-manager 和相關軟體包）。

- 它們有自己的 GUI 和指令列程式來作為使用者介面。
- 它們有自己的後臺背景程式（daemon）作為它們的系統後端。
- 它們使你可以簡單地將系統連線到網路。
- 它們使你可以簡單地管理有線和無線網路的調配。
- 它們允許你調配網路而不依賴傳統的 `ifupdown` 軟體包。

注
不要在伺服器上使用這些自動網路調配工具。它們主要針對於膝上型電腦上的移動桌面使用者。

這些現代的網路調配工具需要進行適當的調配，以避免與傳統 `ifupdown` 軟體包發生衝突，它的組態檔案位於“`/etc/network/interfaces`”。

5.2.1 圖形介面的網路調配工具

Debian 系統 NM 的官方文件位於“`/usr/share/doc/network-manager/README.Debian`”。

本質上，如下操作即可完成桌面的網路調配。

1. 通過下列指令使桌面使用者 `foo` 歸屬“`netdev`”組（另外，例如 GNOME 和 KDE 這樣的現代桌面環境會通過 D-bus 自動完成該操作）。

```
$ sudo usermod -a -G foo netdev
```

2. 使“/etc/network/interfaces”的調配保持下面那樣簡潔。

```
auto lo
iface lo inet loopback
```

3. 透過下列命令重新啟動 NM。

```
$ sudo systemctl restart network-manager
```

4. 通過圖形介面調配網路。

注
只有不列在“/etc/network/interfaces”中的介面會被 NM 管理，以避免與 ifupdown 的衝突。

提示
如果你想擴充 NM 的網路組態功能，請尋找適當的外掛模組和補充軟體包，例如 network-manager-openconnect、network-manager-openvpn-gnome、network-manager-pptp-gnome、mobile-broadband-provider-info、gnome-bluetooth 等等。這同樣適用於 Wicd。

5.3 沒有影象介面的現代網路配置

使用 systemd 的系統中，可以在 /etc/systemd/network/ 裡調配網路。參見 systemd-resolved(8)、resolved.conf(5) 和 systemd-networkd(8)。

這個允許在沒有影象介面的情況下配置現代網路。

DHCP 客戶端的配置可以透過建立”/etc/systemd/network/dhcp.network” 檔案來進行設定。例如：

```
[Match]
Name=en*

[Network]
DHCP=yes
```

一個靜態網路配置能夠透過建立”/etc/systemd/network/static.network” 來設定. 比如：

```
[Match]
Name=en*

[Network]
Address=192.168.0.15/24
Gateway=192.168.0.1
```

5.4 現代雲網絡配置

雲的現代網路配置可以使用 cloud-init 和 netplan.io 軟體包 (參見節 3.7.4)。

netplan.io 軟體包支援把 systemd-networkd 和 NetworkManager 作為它的網路配置後端，能夠使用 YAML 資料宣告網路配置。當你改變 YAML:

- 執行”netplan generate” 命令，從 YAML 生成所有必須的後端配置。

- 執行”netplan apply” 命令應用生成的配置到後端。

參見 ”Netplan documentation”, netplan(5), netplan-generate(8) 和 netplan-apply(8)。

也可以參見 ”Cloud-init documentation” (特別是圍繞 ”Configuration sources” 和 ”Netplan Passthrough”) 瞭解 cloud-init 是怎樣能夠整合替代的資料來源到 netplan.io 配置。

5.4.1 使用 DHCP 的現代雲網絡配置

DHCP 客戶端的配置可以透過建立一個數據原始檔”/etc/netplan/50-dhcp.yaml” 來進行設定:

```
network:
  version: 2
  ethernets:
    all-en:
      match:
        name: "en*"
      dhcp4: true
      dhcp6: true
```

5.4.2 使用靜態 IP 的現代雲網絡配置

一個靜態網路配置能夠透過建立一個數據原始檔”/etc/netplan/50-static.yaml” 來設定:

```
network:
  version: 2
  ethernets:
    eth0:
      addresses:
        - 192.168.0.15/24
      routes:
        - to: default
          via: 192.168.0.1
```

5.4.3 使用 Network Manger 的現代雲網絡配置

使用 Network Manger 架構的網路客戶端配置,可以透過建立一個數據原始檔”/etc/netplan/00-network-manager.yam
來進行設定:

```
network:
  version: 2
  renderer: NetworkManager
```

5.5 底層網路調配

在 Linux 上的底層網路配置，使用 iproute2 程式 (ip(8), ⋯) .

5.5.1 Iproute2 指令

Iproute2 指令集提供完整的底層網路調配能力。有個從舊的 net-tools 指令集到新的 iproute2 指令集的轉換表。

參見 ip(8) 和 Linux 高階路由和流量控制 (Linux Advanced Routing & Traffic Control) .

舊的 **net-tools**	新的 **iproute2**	操作
ifconfig(8)	ip addr	一個裝置上的協議（IP 或 IPv6）地址
route(8)	ip route	路由表條目
arp(8)	ip neigh	ARP 或 NDISC 快取條目
ipmaddr	ip maddr	多播地址
iptunnel	ip tunnel	IP 隧道
nameif(8)	ifrename(8)	基於 MAC 地址的網路介面名
mii-tool(8)	ethtool(8)	乙太網路裝置設定

Table 5.3: 從舊的 net-tools 指令集到新的 iproute2 指令集轉換表

5.5.2 安全的底層網路操作

你可以按下面的方式安全的使用底層網路指令，這些指令不會改變網路調配。

指令	說明
ip addr show	顯示活動的網路介面連線和地址狀態
route -n	用數字地址顯示全部路由表
ip route show	用數字地址顯示全部路由表
arp	顯示當前 ARP 快取表的內容
ip neigh	顯示當前 ARP 快取表的內容
plog	顯示 ppp 後臺背景程式（daemon）日誌
ping yahoo.com	檢查到"yahoo.com" 的因特網連線
whois yahoo.com	在域名資料庫裡面檢查誰註冊了"yahoo.com"
traceroute yahoo.com	跟蹤到"yahoo.com" 的因特網連線
tracepath yahoo.com	跟蹤到"yahoo.com" 的因特網連線
mtr yahoo.com	跟蹤到"yahoo.com" 的因特網連線（重複的）
dig [@dns-server.com] example.com [{a|mx|any}]	查詢由"dns-server.com" 提供服務的"example.com" 域名的 DNS 記錄："a", "mx" 或"any" 記錄
iptables -L -n	檢視包過濾
netstat -a	找出所有開啟的埠
netstat -l --inet	找出監聽埠
netstat -ln --tcp	找出 TCP 監聽埠（數字的）
dlint example.com	查詢"example.com" 的 DNS zone 資訊

Table 5.4: 底層網路指令列表

提示
部分底層網路調配工具放在"/usr/sbin/" 目錄。你可以像"/usr/sbin/ifconfig" 這樣使用完整指令路徑，或把"/usr/sbin" 加到"~/.bashrc" 檔案列出的"$PATH" 環境變數裡。

5.6 網路最佳化

通用的網路優化超出了本文的範圍。我提及消費等級連線相關的主題。

5.6.1 找出最佳 MTU

網路管理器通常會自動設定最佳 最大傳輸單元 (MTU)。

軟體包	流行度	大小	說明
iftop	V:7, I:102	93	顯示一個網路介面上的頻寬使用資訊
iperf	V:2, I:43	360	網際網路協議頻寬測量工具
ifstat	V:0, I:7	59	介面統計監控
bmon	V:1, I:17	144	行動式頻寬監視器和網速估計工具
ethstatus	V:0, I:3	40	快速測量網路裝置吞吐的指令碼
bing	V:0, I:0	80	實驗性的隨機頻寬測試器
bwm-ng	V:1, I:14	95	小巧簡單的控制檯頻寬監測器
ethstats	V:0, I:0	23	基於控制檯的乙太網路統計監視器
ipfm	V:0, I:0	82	頻寬分析工具

Table 5.5: 網路最佳化工具清單

在一些場景中，在用 ping(8) 加上"-M do" 選項傳送各種大小的 ICMP 報文資料包進行實驗後，你希望可以手動設定 MTU。MTU 是最大可完成沒有 IP 分片的資料包大小加上 28 位元組（IPv4）或 48 位元組（IPv6）。下面的列子，發現 IPv4 連線的 MTU 是 1460，IPv6 連線的 MTU 是 1500。

```
$ ping -4 -c 1 -s $((1500-28)) -M do www.debian.org
PING  (149.20.4.15) 1472(1500) bytes of data.
ping: local error: message too long, mtu=1460

---  ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

$ ping -4 -c 1 -s $((1460-28)) -M do www.debian.org
PING  (130.89.148.77) 1432(1460) bytes of data.
1440 bytes from klecker-misc.debian.org (130.89.148.77): icmp_seq=1 ttl=50 time=325 ms

---  ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 325.318/325.318/325.318/0.000 ms
$ ping -6 -c 1 -s $((1500-48)) -M do www.debian.org
PING www.debian.org(mirror-csail.debian.org (2603:400a:ffff:bb8::801f:3e)) 1452 data bytes
1460 bytes from mirror-csail.debian.org (2603:400a:ffff:bb8::801f:3e): icmp_seq=1 ttl=47  ↩
    time=191 ms

--- www.debian.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 191.332/191.332/191.332/0.000 ms
```

這個過程是 路徑 MTU (PMTU) 發現 (RFC1191)，tracepath(8) 指令能夠自動完成這個。

網路環境	**MTU**	基本原理
撥號連線 (IP: PPP)	576	標準的
乙太網路連線 (IP: DHCP 或固定)	1500	預設標準值

Table 5.6: 最佳 MTU 值的基本指引方法

除了這些基本的指引方法外，你還應當知道下面的資訊。

- 使用任何隧道方式 (VPN 等.) 的最佳 MTU 需要進一步減去它們上面的頭部。
- MTU 值不應當超過通過實驗驗證的 PMTU 值。
- 當遇到其它限制的時候，較大的 MTU 值通常比較好。

最大分片大小 (MSS) 是另外一種衡量包大小的方法。MSS 和 MTU 的關係如下.

- 對於 IPv4，MSS = MTU - 40
- 對於 IPv6，MSS = MTU - 60

注
基於 iptables(8) (參見節 5.7) 的優化，能夠通過 MSS 來壓縮包大小，路由器會用到 MMS 。參見 iptables(8) 中的"TCPMSS" .

5.6.2 WAN TCP 最佳化

現代大頻寬和高延時的 WAN，TCP 吞吐量能夠透過調整 TCP 緩衝大小的引數，在"TCP 調整" 裡，來最大化。到目前為止，當前 Debian 預設設定能夠很好的服務好我的 1G bps 光纖到戶 LAN 連線。

5.7 Netfilter 網路過濾框架

Netfilter 使用 Linux 核心 模組 (參見節 3.9) 提供 狀態防火牆 和 網路地址轉換 (NAT) 框架。

軟體包	流行度	大小	說明
iptables	V:312, I:730	2414	netfilter 管理工具 (iptables(8) 用於 IPv4, ip6tables(8) 用於 IPv6)
arptables	V:0, I:1	100	netfilter 管理工具 (arptables(8) 用於 ARP)
ebtables	V:14, I:29	264	netfilter 管理工具 (ebtables(8) 用於乙太網路橋)
iptstate	V:0, I:2	119	持續性監控 netfilter 狀態 (和 top(1) 相似)
ufw	V:53, I:76	857	Uncomplicated Firewall (UFW) 是一個管理 netfilter 防火牆的程式
gufw	V:5, I:10	3660	Uncomplicated Firewall (UFW) 的影象使用者介面
firewalld	V:10, I:15	2612	firewalld 是一個動態管理防火牆的程式，支援網路區域
firewall-config	V:0, I:2	1162	firewalld 影象使用者介面
shorewall-init	V:0, I:0	88	Shoreline 防火牆 初始化
shorewall	V:3, I:8	3090	Shoreline 防火牆, netfilter 組態檔案生成器
shorewall-lite	V:0, I:0	71	Shoreline 防火牆, netfilter 組態檔案生成器 (精簡版)
shorewall6	V:0, I:1	1334	Shoreline 防火牆, netfilter 組態檔案生成器 (IPv6 版本)
shorewall6-lite	V:0, I:0	71	Shoreline 防火牆, netfilter 組態檔案生成器 (IPv6, 精簡版)

Table 5.7: 防火牆工具列表

netfilter 主要的使用者層程式是 iptables(8). 你能從 shell 手工交付式的調配 netfilter，使用 iptables-save(8) 儲存當前狀態, 當系統重啟時，通過 init 指令碼呼叫 iptables-restore(8) 來恢復。

像 shorewall 這樣的調配幫助指令碼能夠使這個過程變得更簡單。

參見 Netfilter 文件 上的文件 (或在"/usr/share/doc/iptables/html/" 裡面的文件)。

- Linux Networking-concepts HOWTO
- Linux 2.4 Packet Filtering HOWTO
- Linux 2.4 NAT HOWTO

提示
雖然這些是為 Linux **2.4** 寫的,iptables(8) 指令和 netfilter 核心功能都能夠在 Linux**2.6** 和 **3.x** 核心系列實現.

Chapter 6

網路應用

建立網路連線後（參加章 5），你可以執行各種網路應用。

提示
對於現代的 Debian 網路基礎設施的具體說明，閱讀 Debian 管理員手冊——網路基礎設施。

提示
在某些 ISP 下，如果你啟用“兩步驗證”，你可能需要獲取一個應用密碼以從你的程式訪問 POP 和 SMTP 服務。你也可能需要事先允許你的主機 IP 進行訪問。

6.1 網頁瀏覽器

有許多網頁瀏覽器軟體包，使用超文字傳輸協議（HTTP）存取遠端內容。

軟體包	流行度	大小	類型	網路瀏覽器說明
chromium	V:33, I:109	231205	X	Chromium，（來自 Google 的開源瀏覽器）
firefox	V:8, I:12	234690	同上	Firefox，（來自 Mozilla 的開源瀏覽器，僅在 Debian Unstable 中可用）
firefox-esr	V:211, I:433	228939	同上	Firefox ESR（Firefox 延長支援版本）
epiphany-browser	V:3, I:15	2192	同上	GNOME，相容 HIG，Epiphany
konqueror	V:24, I:105	25892	同上	KDE，Konqueror
dillo	V:0, I:5	1565	同上	Dillo，（基於 FLTK 的輕量級瀏覽器）
w3m	V:14, I:188	2837	文字	w3m
lynx	V:24, I:330	1948	同上	Lynx
elinks	V:3, I:21	1654	同上	ELinks
links	V:3, I:29	2314	同上	Links（純文字）
links2	V:1, I:12	5492	影象	Links（沒有 X 的控制檯影象）

Table 6.1: 網頁瀏覽器列表

6.1.1 偽裝使用者代理字串

為了訪問一些過度限制的網站，你可能需要偽裝網頁瀏覽器程式返回的 User-Agent 字串。參見：

- MDN Web Docs: userAgent
- Chrome Developers: Override the user agent string
- How to change your user agent
- How to Change User-Agent in Chrome, Firefox, Safari, and more
- How to Change Your Browser’ s User Agent Without Installing Any Extensions
- How to change the User Agent in Gnome Web (epiphany)

注意
偽裝的使用者代理字串可以會導致 來自 Java 的不良副作用。

6.1.2 瀏覽器擴充套件

所有現代的 GUI（圖形使用者介面）瀏覽器支援基於 browser extension 的原始碼，它在按 web extensions 變成標準化。

6.2 郵件系統

本章節關注於消費者級網際網路連線的典型的移動工作站。

注意
如果你想設定郵件伺服器來直接通過網際網路交換郵件，你應該最好閱讀一下這個基本文件。

6.2.1 電子郵件基礎

電子郵件 由三個部分組成，訊息的信封，郵件標頭及郵件正文。

- SMTP 用電子郵件信封上的”To” 和”From” 資訊來投遞郵件。(信封上的”From” 資訊也被叫做退回地址, 例如 From_ 等等)。
- 電子郵件頭的”To” 和”From” 資訊，顯示在 電子郵件客戶端上. (在大部分情況下，這些資訊是跟電子郵件信封一致，但並不全是這樣。)
- 覆蓋郵件頭和正文資料的電子郵件訊息格式被 多用途網際網路郵件擴充套件 (MIME) 擴充套件，從純文字的 ASCII 到其它字元編碼，包括作為附件的音訊、影片、影象和應用程式。

功能全面的基於 電子郵件客戶端的 GUI 程式使用基於 GUI 的直觀的配置，提供下列所有功能。

- 為了處理正文資料型別及其編碼，它建立和使用多用途網際網路郵件擴充套件 (MIME)來解釋郵件標頭和郵件正文。
- 它使用舊的 基礎訪問認證 或現代的 OAuth 2.0向 ISP（網際網路服務提供商）的 SMTP 和 IMAP 伺服器認證它自己。(對於 OAuth 2.0, 透過桌面環境設定來設定它，例如，”Settings” -> ”Online Accounts”.)
- 它傳送訊息到 ISP 的智慧主機的 SMTP 服務監聽的訊息遞交埠（587)。
- 從 TLS/IMAP4 埠（993）接收儲存在 ISP 的伺服器上的訊息。
- 它能夠透過他們的屬性過濾郵件。
- 它能夠提供額外的功能：聯絡人、日曆、任務、備忘錄。

軟體包	流行度	大小	類型
evolution	V:29, I:236	486	X GUI 程式 (GNOME3, groupware 套件)
thunderbird	V:55, I:119	224712	X GUI 程式 (GTK，Mozilla Thunderbird)
kmail	V:37, I:96	23871	X GUI 程式 (KDE)
mutt	V:16, I:154	7104	很有可能與 vim 一起使用的字元終端程式
mew	V:0, I:0	2319	(x)emacs 下的字元終端程式

Table 6.2: 郵件使用者代理列表 (MUA)

6.2.2 現代郵件服務限制

現代郵件伺服器有一些限制來最小化暴露濫用（不希望和未被要求的電子郵件）問題。

- 在消費者級的網路上執行 SMTP 伺服器來直接可靠的傳送郵件到遠端主機是不現實的。
- 一個郵件能夠被任何主機靜悄悄的拒絕，即使路由到了目的地，除非它儘可能看起來是經過認證的。
- 期望單個智慧主機可靠的傳送不相關的源郵件地址到遠端主機，這是不現實的。

這是因為：

- 從消費者級網路提供的主機連線到網際網路的 SMTP 埠（25）已經被封鎖了。
- 從網際網路的 SMTP 埠（25）連線到消費者級網路提供的主機已經被封鎖了。
- 從消費者級網路提供的主機發出到網際網路的訊息，只能夠透過訊息遞交埠（587）傳送。
- 像域名金鑰識別郵件 (DKIM)、發信者策略框架 (SPF) 和 基於域名的訊息認證、報告和反應（DMARC）這樣的反垃圾郵件技術廣泛用於電子郵件過濾。
- 域名金鑰識別郵件服務可能會用於你的通過 smarthost 的電子郵件傳送。
- 智慧主機可以在郵件頭重寫源電子郵件地址為你的郵件賬戶，來阻止電子郵件欺詐。

6.2.3 歷史郵件服務端期望

一些在 Debian 上的程式，它們預設期望訪問 /usr/sbin/sendmail 命令來發送郵件，或者從一個個性化設定的 UNIX 系統郵件伺服器來發送郵件，實現歷史的功能：

- 郵件是由純文字檔案建立。
- 郵件是由 /usr/sbin/sendmail 命令處理。
- 對於目的地址為同一主機，/usr/sbin/sendmail 命令進行郵件的本地分發，將郵件附在 /var/mail/$username 檔案後。
  - 期望這個特徵的命令：apt-listchanges, cron, at, ...
- 對於目的地址在遠端主機，/usr/sbin/sendmail 命令遠端傳輸郵件到目的主機，使用 SMTP 發現 DNS MX 記錄。
  - 期望這個特徵的命令：popcon, reportbug, bts, ...

6.2.4 郵件傳輸代理 (MTA)

在 Debian 12 Bookworm 後，在沒有 mail transfer agent (MTA) 程式的情況下，Debian 移動工作站可以基於 電子郵件客戶端，配置為全功能的 GUI（影象使用者介面）。

以往的 Debian 會安裝某個 MTA 程式來支援期望 `/usr/sbin/sendmail` 命令的程式。移動工作站上這樣的 MTA 必須和節 6.2.2 節 6.2.3 協同工作。

對於移動工作站，典型的 MTA 選擇是 `exim4-daemon-light` 或 `postfix`，並選擇類似這樣的安裝選項：“Mail sent by smarthost; received via SMTP or fetchmail”。這些是輕量 MTA 和”`/etc/aliases`” 匹配。

提示
配置 `exim4` 來發送網際網路郵件，多個源電子郵件地址使用多個相應的智慧主機，這是不尋常的。如果一些程式需要這樣的能力，使用 `msmtp` 來設定他們，它比較容易來設定多個源電子郵件地址。然後給主 MTA 僅僅保留單個電子郵件地址。

軟體包	流行度	大小	說明
`exim4-daemon-light`	V:219, I:231	1576	Exim4 郵件傳輸代理 (MTA : Debian 預設的)
`exim4-daemon-heavy`	V:6, I:6	1740	Exim4 郵件傳輸代理 (MTA : 靈活的替代品)
`exim4-base`	V:226, I:239	1699	Exim4 文件 (文字) 和通用檔案
`exim4-doc-html`	I:1	3746	Exim4 文件 (html)
`exim4-doc-info`	I:0	637	Exim4 文件 (info)
`postfix`	V:126, I:135	4039	Postfix 郵件傳輸代理 (MTA : 安全的替代品)
`postfix-doc`	I:7	4646	Postfix 文件 (html+text)
`sasl2-bin`	V:5, I:14	371	Cyrus SASL API 實現 (實現 postfix SMTP 認證)
`cyrus-sasl2-doc`	I:1	2154	Cyrus SASL - 文件
`msmtp`	V:6, I:12	667	輕量 MTA
`msmtp-mta`	V:5, I:6	124	輕量 MTA (sendmail 對 `msmtp` 的相容擴充)
`esmtp`	V:0, I:0	129	輕量 MTA
`esmtp-run`	V:0, I:0	32	輕量 MTA (sendmail 對 `esmtp` 的相容擴充)
`nullmailer`	V:8, I:9	474	部分功能 MTA，沒有本地郵件
`ssmtp`	V:5, I:8	2	部分功能 MTA，沒有本地郵件
`sendmail-bin`	V:13, I:14	1877	全功能 MTA(如果你已經對它熟悉)
`courier-mta`	V:0, I:0	2407	全功能 MTA(web 介面等.)
`git-email`	V:0, I:10	1087	`git-send-email`(1) 傳送系列補丁郵件程式

Table 6.3: 基礎的郵件傳輸代理相關的軟體包列表

6.2.4.1 exim4 的調配

對於那些通過 smarthost 的網路郵件，你應該按如下所示的 (重新) 調配 `exim4-*` 軟體包。

```
$ sudo systemctl stop exim4
$ sudo dpkg-reconfigure exim4-config
```

調配”General type of mail configuration” 時，選擇”mail sent by smarthost; received via SMTP or fetchmail”。

設定”System mail name:” 為預設的 FQDN (參見節 5.1.1)。

設定”IP-addresses to listen on for incoming SMTP connections:” 為預設的”127.0.0.1; ::1”。

”Other destinations for which mail is accepted:” 選項留空。

”Machines to relay mail for:” 選項留空。

設定”IP address or host name of the outgoing smarthost:” 為”smtp.hostname.dom:587”。

設定”Hide local mail name in outgoing mail?” 選項為”NO”。(或者像節 6.2.4.3描述的那樣使用 `/etc/email-addresses` 代替)

選擇如下所示的其中一個來回答”Keep number of DNS-queries minimal (Dial-on-Demand)?”。

- ”No” 如果啟動的時候，系統就連上了網際網路。
- ”Yes” 如果啟動的時候，系統沒有連上網際網路。

設定”Delivery method for local mail:” 選項為”mbox format in /var/mail/”。

”Split configuration into small files?:” 選項設為”Yes”。

通過修改”`/etc/exim4/passwd.client`” 檔案，來建立用於 smarthost 的密碼條目。

```
$ sudo vim /etc/exim4/passwd.client
 ...
$ cat /etc/exim4/passwd.client
^smtp.*\.hostname\.dom:username@hostname.dom:password
```

配置 `exim4`(8),在”`/etc/default/exim4`” 檔案中寫入”`QUEUERUNNER='queueonly'`”,”`QUEUERUNNER='nodaemon'`” 等等，來最小化系統資源使用。(可選的)

通過如下所示的啟動 `exim4`。

```
$ sudo systemctl start exim4
```

”`/etc/exim4/passwd.client`” 檔案中的主機名不應該是別名，你應該按如下所示的檢查真正的主機名。

```
$ host smtp.hostname.dom
smtp.hostname.dom is an alias for smtp99.hostname.dom.
smtp99.hostname.dom has address 123.234.123.89
```

我在”`/etc/exim4/passwd.client`” 檔案中使用正規表達式來繞過別名問題。即使 ISP 更改了別名所指向的主機名，SMTP AUTH 還是可能工作的。

你能夠通過如下所示的手動更新 `exim4` 調配:

- 更新”`/etc/exim4/`” 目錄下的 `exim4` 組態檔案。
  - 建立”`/etc/exim4/exim4.conf.localmacros`” 來設定巨集指令和修改”`/etc/exim4/exim4.conf.template`” 檔案。(沒有分割的調配)
  - 在”`/etc/exim4/exim4.conf.d`” 子目錄中建立新檔案或編輯已存在的檔案。(分割的調配)
- 執行”`systemctl reload exim4`”.

注意
如果 debconf 詢問”Keep number of DNS-queries minimal (Dial-on-Demand)?” 這個問題時，選擇了”No” (預設值)，那麼啟動 `exim4` 會花很長時間並且系統在啟動的時候不會連線到網際網路。

請閱讀”`/usr/share/doc/exim4-base/README.Debian.gz`” 官方指導和 `update-exim4.conf`(8)。

警告
從所有的實踐考慮，使用帶 STARTTLS 的 SMTP 埠 587，或者 SMTPS (SSL 之上的 SMTP) 埠 465, 代替純 SMTP 埠 25。

6.2.4.2 帶有 SASL 的 postfix 調配

對於通過 smarthost 的網路郵件，你應該首先閱讀 postfix 文件和關鍵的手冊頁。

指令	功能
`postfix`(1)	Postfix 控制程式
`postconf`(1)	Postfix 調配工具
`postconf`(5)	Postfix 調配參數
`postmap`(1)	Postfix 查詢表維護
`postalias`(1)	Postfix 別名資料庫維護

Table 6.4: 重要的 postfix 手冊頁列表

你應該按如下所示的 (重新) 調配 `postfix` 和 `sasl2-bin` 軟體包。

```
$ sudo systemctl stop postfix
$ sudo dpkg-reconfigure postfix
```

選擇”Internet with smarthost”。

設定”SMTP relay host (blank for none):” 為”`[smtp.hostname.dom]:587`” 並按如下所示調配。

```
$ sudo postconf -e 'smtp_sender_dependent_authentication = yes'
$ sudo postconf -e 'smtp_sasl_auth_enable = yes'
$ sudo postconf -e 'smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd'
$ sudo postconf -e 'smtp_sasl_type = cyrus'
$ sudo vim /etc/postfix/sasl_passwd
```

為 smarthost 建立密碼條目。

```
$ cat /etc/postfix/sasl_passwd
[smtp.hostname.dom]:587     username:password
$ sudo postmap hush:/etc/postfix/sasl_passwd
```

通過如下所示的啟動 `postfix`。

```
$ sudo systemctl start postfix
```

`dpkg-reconfigure` 會話中使用的”[” 和”]” 和”`/etc/postfix/sasl_passwd`” 確保不去檢查 MX 記錄而是直接使用指定的明確主機名。參見”`/usr/share/doc/postfix/html/SASL_README.html`” 裡面的”Enabling SASL authentication in the Postfix SMTP client” 條目。

6.2.4.3 郵件地址調配

這裡有一些用於郵件傳輸、投遞和使用者代理的郵件地址組態檔案。

檔案	功能	應用
`/etc/mailname`	用於 (外發) 郵件的預設主機名	Debian 專用的，`mailname`(5)
`/etc/email-addresses`	用於外發郵件的主機名偽裝	`exim`(8) 專用的，`exim4-config_files`(5)
`/etc/postfix/generic`	用於外發郵件的主機名偽裝	`postfix`(1) 專用的，`postmap`(1) 指令執行後啟用。
`/etc/aliases`	用於接收郵件的帳號別名	通用的，`newaliases`(1) 指令執行後啟用。

Table 6.5: 與郵件地址相關的組態檔案列表

”`/etc/mailname`” 檔案中的 **mailname** 通常是全稱域名 (FQDN)，這個全程域名將會被解析成主機的 IP 地址。對於沒有可解析成 IP 地址的主機名的移動工作站，設定 **mailname** 為”`hostname -f`” 的值。(這對於 `exim4-*` 和 `postfix` 都是安全有效的選擇。)

提示
"/etc/mailname" 中的內容被許多非 MTA 程式用作它們的預設行為。對於 mutt, 在 ~/muttrc 檔案中設定"hostname" 和"from" 變數來覆蓋 **mailname** 值。對於 devscripts 軟體包的程式，例如 bts(1) 和 dch(1)，匯出環境變數"$DEBFULLNAME" 和"$DEBEMAIL" 的值來覆蓋它。

提示
popularity-contest 軟體包一般以 FQDN 形式的 root 帳號傳送郵件。你需要像 /usr/share/popularity-contest/default.conf 檔案中描述的那樣去設定 /etc/popularity-contest.conf 檔案中的 MAILFROM 值。否則，你的郵件會被 smarthost SMTP 伺服器拒絕。儘管這些過程很乏味，這種方法比為所有通過 MTA 並且是以 root 使用者傳送的郵件重寫源地址更安全。這也可以被其他背景程式或者是 cron 指令碼使用。

當設定 **mailname** 為"hostname -f" 的值時，通過 MTA 的源郵件地址的偽裝可以通過如下所示的來實現。

- 用於 exim4(8) 的"/etc/email-addresses" 檔案，exim4-config_files(5) 手冊頁中有關於它的解釋
- 用於 postfix(1) 的"/etc/postfix/generic" 檔案，generic(5) 手冊頁中有關於它的解釋

對於 postfix，接下來的額外步驟需要執行。

```
# postmap hash:/etc/postfix/generic
# postconf -e 'smtp_generic_maps = hash:/etc/postfix/generic'
# postfix reload
```

你能夠通過如下所示的來測試郵件地址調配。

- exim(8) 用 -brw, -bf, -bF, -bV, ... 選項
- postmap(1) 用 -q 選項。

提示
Exim 帶有一些有用的程式，例如 exiqgrep(8) 和 exipick(8)。參見"dpkg -L exim4-base|grep man8/" 來獲得可用的指令。

6.2.4.4 基礎 MTA 操作

這裡有一些基礎的 MTA 操作。有一些可能會通過 sendmail(1) 的相容性介面來實現。

提示
往"/etc/ppp/ip-up.d/*" 裡寫一個重新整理所有郵件的指令碼會是個不錯的主意。

6.3 伺服器遠端存取和工具 (SSH)

Secure SHell (SSH) 是因特網上的 安全連線方式。在 Debian 裡面，有一個叫 OpenSSH 的免費 SSH 版本，在 openssh-client 和 openssh-server 包裡。

對於使用者來講, ssh(1) 功能比 telnet(1) 更加智慧和安全. 不像 telnet 指令, ssh 指令不會在遇到 telnet 的退出字元 (初始預設是 CTRL-]) 時停止.

雖然 shellinabox 不是一個 SSH 程式，它列在這裡作為遠端終端訪問的一個有趣的替代。

連線到遠端 X 客戶端程式，參見：節 7.8。

exim 指令	**postfix** 指令	說明
`sendmail`	`sendmail`	從標準輸入讀取郵件並且安排投遞 (`-bm`)
`mailq`	`mailq`	列出帶有狀態和佇列 ID 的郵件佇列 (`-bq`)
`newaliases`	`newaliases`	初始化別名資料庫 (`-I`)
`exim4 -q`	`postqueue -f`	重新整理等待郵件 (`-q`)
`exim4 -qf`	`postsuper -r ALL deferred; postqueue -f`	重新整理所有郵件
`exim4 -qff`	`postsuper -r ALL; postqueue -f`	重新整理甚至已經凍結的郵件
`exim4 -Mg queue_id`	`postsuper -h queue_id`	通過郵件的佇列 ID 來凍結它
`exim4 -Mrm queue_id`	`postsuper -d queue_id`	通過郵件的佇列 ID 來移除它
N/A	`postsuper -d ALL`	移除所有郵件

Table 6.6: 基礎 MTA 操作列表

軟體包	流行度	大小	工具	說明
`openssh-client`	V:857, I:995	4959	`ssh`(1)	SSH 客戶端
`openssh-server`	V:726, I:817	1804	`sshd`(8)	SSH 服務端
`ssh-askpass`	I:23	102	`ssh-askpass`	請求使用者輸入密碼的 ssh-add (plain X)
`ssh-askpass-gnome`	V:0, I:3	200	`ssh-askpass`	請求使用者輸入 ssh-add 密碼 (GNOME)
`ssh-askpass-fullscreen`	V:0, I:0	48	`ssh-askpass-fullscreen`(1)	請求使用者輸入 ssh-add 密碼 (GNOME)，有更好看的介面
`shellinabox`	V:0, I:1	507	`shellinaboxd`(1)	利用瀏覽器訪問 VT100 終端模擬器 網頁伺服器

Table 6.7: 伺服器遠端存取和工具列表

注意
如果你的 SSH 是從因特網來存取，參見節 4.6.3。

提示
請使用 `screen`(1) 程式來讓遠端 shell 在中斷的連線上存活 (參見節 9.1.2).

6.3.1 SSH 基礎

OpenSSH SSH 後臺守護程序 (daemon) 只支援 SSH 2 協議。

請閱讀”`/usr/share/doc/openssh-client/README.Debian.gz`”、`ssh`(1)、`sshd`(8)、`ssh-agent`(1)、`ssh-keygen`(1)、`ssh-add`(1) 和 `ssh-agent`(1).

警告
如果想要執行 OpenSSH 服務，”`/etc/ssh/sshd_not_to_be_run`” 必須不存在。
不要開啟基於 rhost 的認證 (`/etc/ssh/sshd_config` 中的 `HostbasedAuthentication`)。

從客戶端啟動一個 `ssh`(1) 連線.

組態檔案	組態檔案描述
/etc/ssh/ssh_config	SSH 客戶端預設, 參見 ssh_config(5)
/etc/ssh/sshd_config	SSH 服務端預設, 參見 sshd_config(5)
~/.ssh/authorized_keys	該帳號連線到這個伺服器上的客戶端使用的預設 SSH 公鑰
~/.ssh/id_rsa	使用者的 SSH-2 RSA 私鑰
~/.ssh/id_*key-type-name*	使用者的 SSH-2 金鑰，*key-type-name* 為 ecdsa、ed25519 等

Table 6.8: SSH 組態檔案列表

指令	說明
ssh username@hostname.domain.ext	使用預設模式連線
ssh -v username@hostname.domain.ext	有詳細資訊的預設連線模式
ssh -o PreferredAuthentications=password username@hostname.domain.ext	SSH 2 版本，強制使用密碼
ssh -t username@hostname.domain.ext passwd	在遠端主機上執行 passwd 命令來更新密碼

Table 6.9: SSH 客戶端啟動例子列表

6.3.2 遠端主機上的使用者名稱

如果你在本地和遠端主機上使用相同的使用者名稱，你能夠省略輸入”username@”。

如果本地和遠端主機，使用同樣的使用者名稱，你可以省略輸入”username@”. 即使在本地和遠端主機使用不同的使用者名稱，你可以使用”~/.ssh/config” 來省略輸入使用者名稱. 對於 Debian Salsa 伺服器，使用帳號名”foo-guest”，你可以設定”~/.ssh/config” 包含下面的內容。

```
Host salsa.debian.org people.debian.org
User foo-guest
```

6.3.3 免密碼遠端連線

使用”PubkeyAuthentication” (SSH-2 協議)，人們可以避免記住遠端系統的密碼.

在遠端系統的”/etc/ssh/sshd_config” 裡，設定相應的條目, ”PubkeyAuthentication yes”。

在本地生成授權祕鑰對，並安裝公鑰到遠端系統。

```
$ ssh-keygen -t rsa
$ cat .ssh/id_rsa.pub | ssh user1@remote "cat - >>.ssh/authorized_keys"
```

你可以在”~/.ssh/authorized_keys” 裡給條目增加選項來限制主機和執行特定的命令。參見 sshd(8)”AUTHORIZED_KEYS FILE FORMAT” 。

6.3.4 處理其它 SSH 客戶端

其它平臺上有一些免費的 SSH 客戶端。

環境	免費 **SSH** 程式
Windows	puTTY (PuTTY：一個自由的 SSH 、 Telnet 客戶端) (GPL)
Windows (cygwin)	cygwin 裡的 SSH(Cygwin: Get that Linux feeling - on Windows) (GPL)
Mac OS X	OpenSSH; 在終端應用中使用 ssh (GPL)

Table 6.10: 其它平臺上免費 SSH 客戶端列表

6.3.5 建立 ssh 代理

用密碼來保護你的 SSH 認證私鑰是安全的。如果密碼沒有設定，使用”ssh-keygen -p” 來設定。

把你的公鑰 (比如：”~/.ssh/id_rsa.pub”) 放到遠端主機的”~/.ssh/authorized_keys”，這個遠端主機使用上面描述的基於密碼的連線方式。

```
$ ssh-agent bash
$ ssh-add ~/.ssh/id_rsa
Enter passphrase for /home/username/.ssh/id_rsa:
Identity added: /home/username/.ssh/id_rsa (/home/username/.ssh/id_rsa)
```

從這裡執行接下來的指令，就不再需要密碼。

```
$ scp foo username@remote.host:foo
```

按 ^D 來終結 ssh 代理會話。

對於 X 服務端，通常的 Debian 啟動指令碼會作為父程序執行 ssh-agent。所以你只需要執行一次 ssh-add。進一步的資訊，請閱讀 ssh-agent(1) 和 ssh-add(1).

6.3.6 從遠端主機發送郵件

如果你在一個正確設定了 DNS 的伺服器上有一個 SSH shell 賬號，你能夠將在你本地工作站上生成的郵件，作為遠端伺服器上的郵件，真正的從遠端伺服器上傳送。

```
$ ssh username@example.org /usr/sbin/sendmail -bm -ti -f ”username@example.org” < mail_data ↩
    .txt
```

6.3.7 SMTP/POP3 隧道的埠轉發

通過 ssh 建立一個這樣的管道連線，從 localhost 的 4025 埠到 remote-server 的 25 埠，並從 localhost 的 4110 埠到 remote-server 的 110 埠，請在本機執行如下指令.

```
# ssh -q -L 4025:remote-server:25 4110:remote-server:110 username@remote-server
```

這是跨越因特網建立 SMTP/POP3 服務連線的安全方法。在遠端主機”/etc/ssh/sshd_config” 裡設定”AllowTcpForwarc
條目為”yes”.

6.3.8 怎樣通過 SSH 關閉遠端系統

你可以使用 at(1) 指令 (參見節 9.4.13) 來從 SSH 終端裡保護”shutdown -h now” (參見節 1.1.8) 操作過程。

```
# echo ”shutdown -h now” | at now
```

在 screen(1) (參見節 9.1.2) 會話裡執行”shutdown -h now”，是另外一個方法來做這同樣的事情。

6.3.9 SSH 故障排查

如果你遇到問題，檢查組態檔案的許可權並用”-v” 選項執行 ssh。

如果你是 root 帳號，並有使用防火牆，使用”-p” 選項; 這可以避免使用 1 —1023 之間的服務埠.

如果 ssh 連線到遠端站點突然停止工作，這也許是系統管理員胡亂操作的結果，可能是在系統維護時改變了”host_key”. 在確認這個情況後,並且沒有人試圖用聰明的黑客技術來篡改遠端主機,你可以在本機”~/.ssh/known_hos裡刪除”host_key” 條目來重新獲得連線。

6.4 列印服務和工具

在老的類 Unix 系統中，BSD Line printer daemon(lpd) 行印表機後臺守護 曾經是標準。傳統的自由軟體的標準列印輸出格式是 PostScript (PS)。為了能夠列印到非 PostScript 印表機，需要將一些過濾器系統和 Ghostscript 一道使用。參見節 11.4.1。

在現代的 Debian 系統中，Common UNIX Printing System 通用 UNIX 列印系統是事實上的標準。現代自由軟體的標準列印輸出格式是 Portable Document Format (PDF) 可移植檔案格式。

CUPS 使用 Internet Printing Protocol 網際網路列印協議 (IPP). IPP 現在已經被其它作業系統，如 Windows XP 和 Mac OS X 支援。它已經變成新的具備雙向通訊能力的跨平臺遠端列印的事實標準。

幸虧有 CUPS 系統的檔案格式依賴自動轉化特徵，簡單的傳送任何資料到 lpr 指令，都將產生期望的列印輸出。(在 CUPS 裡, lpr 能夠通過安裝 cups-bsd 軟體包來獲取.)

Debian 系統有一些不錯的軟體包用於列印服務和作為列印工具。

軟體包	流行度	大小	埠	說明
lpr	V:2, I:3	367	printer (515)	BSD lpr/lpd (線性印表機後臺背景程式 daemon)
lprng	V:0, I:0	3051	同上	, , (增強)
cups	V:95, I:436	1061	IPP (631)	網際網路列印 CUPS 伺服器
cups-client	V:116, I:457	426	同上	用於 CUPS 的 System V 印表機指令: lp(1), lpstat(1), lpoptions(1), cancel(1), lpmove(8), lpinfo(8), lpadmin(8), …
cups-bsd	V:31, I:223	131	同上	用於 CUPS 的 BSD 印表機指令: lpr(1), lpq(1), lprm(1), lpc(8)
printer-driver-gutenprint	V:25, I:120	1219	沒有使用	CUPS 印表機驅動

Table 6.11: 列印服務和工具列表

提示
你可以讓你的 web 瀏覽器存取”http://localhost:631/” 來調配 CUPS 系統。

6.5 其它網路應用服務

這裡是其它網路應用服務。

通用網際網路檔案系統協議 (CIFS) 和服務訊息塊 (SMB) 協議一樣，被微軟 Windows 廣泛應用。

提示
參見節 4.5.2 服務系統整合。

軟體包	流行度	大小	協議	說明
telnetd	V:0, I:2	54	TELNET	TELNET 服務
telnetd-ssl	V:0, I:0	159	同上	TELNET 服務 (支援 SSL)
nfs-kernel-server	V:49, I:64	769	NFS	Unix 檔案共享
samba	V:108, I:134	3997	SMB	Windows 檔案和列印共享
netatalk	V:1, I:1	2003	ATP	Apple/Mac 檔案和列印共享 (AppleTalk)
proftpd-basic	V:9, I:17	452	FTP	通用檔案下載
apache2	V:215, I:264	561	HTTP	通用 web 伺服器
squid	V:11, I:12	9265	同上	通用 web 代理伺服器
bind9	V:43, I:49	1123	DNS	其它主機的 IP 地址
isc-dhcp-server	V:19, I:37	6082	DHCP	客戶端自身的 IP 地址

Table 6.12: 其它網路應用服務列表

提示
主機名解析通常由 DNS 服務提供. 對於由 DHCP 動態分配的主機 IP 地址, 動態 DNS 能夠使用 bind9 和 isc-dhcp-server 建立主機名解析，Debian wiki 的 DDNS 頁 有說明.

提示
使用 squid 之類的代理伺服器，和使用 Debian 文件庫的完全本地映象伺服器相比，能夠大量節省頻寬。

6.6 其它網路應用客戶端

這裡是其它網路應用客戶端。

6.7 系統後臺背景程式（daemon）診斷

telnet 程式能夠手工連線到系統後臺背景程式（daemon），並進行診斷。

測試純 POP3 服務，嘗試用下面的操作

```
$ telnet mail.ispname.net pop3
```

部分 ISP 提供 TLS/SSL 加密的POP3 服務，為了測試它，你需要用到 telnet-ssl 包裡支援 TLS/SSL 的 telnet 客戶端，或 openssl 軟體包。

```
$ telnet -z ssl pop.gmail.com 995
```

```
$ openssl s_client -connect pop.gmail.com:995
```

下面的 RFCs 提供每一個系統後臺背景程式（daemon）所需要的知識。

在”/etc/services” 裡，描述了埠用途.

軟體包	流行度	大小	協議	說明
netcat	I:28	16	TCP/IP	TCP/IP 瑞士軍刀
openssl	V:839, I:995	2294	SSL	安全套接字層 (SSL) 二進位制和相關的加密工具
stunnel4	V:7, I:12	538	同上	通用 SSL 封裝
telnet	V:30, I:537	54	TELNET	TELNET 客戶端
telnet-ssl	V:0, I:2	196	同上	TELNET 服務 (支援 SSL)
nfs-common	V:153, I:237	1124	NFS	Unix 檔案共享
smbclient	V:23, I:204	2071	SMB	微軟 Windows 檔案和列印共享客戶端
cifs-utils	V:30, I:122	317	同上	遠端微軟 Windows 檔案系統掛載和解除安裝指令
ftp	V:7, I:119	53	FTP	FTP 客戶端
lftp	V:4, I:30	2361	同上	同上
ncftp	V:2, I:15	1389	同上	全螢幕 FTP 客戶端
wget	V:209, I:980	3681	HTTP 和 FTP	web 下載工具
curl	V:178, I:620	518	同上	同上
axel	V:0, I:3	201	同上	下載加速器
aria2	V:2, I:19	1981	同上	BitTorrent 和 Metalink 支援的下載加速器
bind9-host	V:128, I:940	392	DNS	來自 bind9 的 host(1), "Priority: standard"
dnsutils	V:17, I:289	275	同上	來自 bind 的 dig(1), "Priority: standard"
isc-dhcp-client	V:218, I:981	2866	DHCP	獲得 IP 地址
ldap-utils	V:13, I:65	767	LDAP	從 LDAP 服務獲得資料

Table 6.13: 網路應用客戶端列表

RFC	說明
rfc1939 和 rfc2449	POP3 服務
rfc3501	IMAP4 服務
rfc2821 (rfc821)	SMTP 服務
rfc2822 (rfc822)	郵件檔案格式
rfc2045	多用途網際網路郵件擴充套件 (MIME)
rfc819	DNS 服務
rfc2616	HTTP 服務
rfc2396	URI 定義

Table 6.14: 常用 RFC 列表

Chapter 7

GUI（圖形使用者介面）系統

7.1 GUI（圖形使用者介面）桌面環境

在 Debian 系統上，有幾個功能全面的 GUI 桌面環境選擇。

任務軟體包	流行度	大小	說明
task-gnome-desktop	I:195	9	GNOME 桌面環境
task-xfce-desktop	I:94	9	Xfce 桌面環境
task-kde-desktop	I:80	6	KDE Plasma 桌面環境
task-mate-desktop	I:41	9	MATE 桌面環境
task-cinnamon-desktop	I:39	9	Cinnamon 桌面環境
task-lxde-desktop	I:27	9	LXDE 桌面環境
task-lxqt-desktop	I:16	9	LXQt 桌面環境
task-gnome-flashback-desktop	I:11	6	GNOME Flashback 桌面環境

Table 7.1: 桌面環境列表

提示
選擇的任務元軟體包的依賴軟體包，在 Debian 非穩定版/測試版環境下，由於最新的軟體包變遷狀態，可能沒有及時同步。對於 `task-gnome-desktop`，你可以按下面的方法調整軟體包選擇：

- 用 `sudo aptitude -u` 啟動 `aptitude`(8)。
- 移動游標到”Tasks” 並按回車鍵。
- 移動游標到”End-user” 並按回車鍵。
- 移動游標到”GNOME” 並按回車鍵。
- 移動游標到 `task-gnome-desktop` 並按回車鍵。
- 移動游標到”Depends” 並按”m”（手工選擇）。
- 移動游標到”Recommends” 並按”m”（手工選擇）。
- 移動游標到”`task-gnome-desktop` 並按”-”。（刪除）
- 調整選擇的軟體包，並刪除造成軟體包衝突的問題軟體包。
- 按”g” 來開始安裝。

本章將大部分關注 Debian 預設的桌面環境：`task-gnome-desktop`，在 wayland 上提供 GNOME。

7.2 GUI（圖形使用者介面）通訊協議

在 GNOME 桌面使用的 GUI 通訊協議可以為：

- Wayland (服務端顯示協議) (原生)
- X 視窗系統核心協議 (透過 `xwayland`)

請檢視 freedesktop.org 站點來了解 Wayland 架構和 X 視窗架構是如何不同。

從使用者的觀點，不同能夠被通俗的概況為：

- Wayland 是在同一個主機上的 GUI 通訊協議：新、簡單、快速，不需要 setuid root 二進位制
- X Window 是一個具備網路功能的 GUI 通訊協議：傳統、複雜、慢，需要 setuid root 二進位制

對於使用 Wayland 協議的應用，由 VNC 或 RDP 來支援從一個遠端主機上訪問它們顯示的內容。參見節 7.7

現代 X 伺服器具有MIT 共享記憶體擴充 ，他們和本地 X 客戶端通過本地共享記憶體進行通訊。這就繞過了網路透明的 Xlib 程序間通訊通道，提升了大型影象的處理效能。這也是 Wayland 僅能作為本地端 GUI 通訊協定的建立背景。

使用從 GNOME 終端啟動的 `xeyes` 程式，你能夠檢查每個 GUI（圖形使用者介面）應用程式使用的 GUI 通訊協議。

```
$ xeyes
```

- 如果滑鼠是在使用 Wayland 服務端顯示協議的應用程式上，比如“GNOME 終端”，眼睛不會跟隨滑鼠移動。
- 如果滑鼠是在使用 X 視窗系統核心協議的應用程式上，比如“xterm”，眼睛會跟隨滑鼠移動，暴露出不是那麼孤立的 X 視窗架構的特性。

到 2021 年 4 月，許多流行的 GUI 應用程式，比如 GNOME 和 LibreOffice (LO) 已經被移植到了 Wayland 服務端顯示協議。我發現 `xterm`, `gitk`, `chromium`, `firefox`, `gimp`, `dia` 和 KDE 應用程式任然使用 X 視窗系統核心協議。

注
對於 Wayland 之上的 xwayland 或原生的 X 視窗系統，這兩個上面的舊的 X 服務端配置檔案"/etc/X11/xorg.conf" 不應當在系統上存在。顯示卡和輸入裝置目前是由核心的 DRM、KMS 和 udev 配置。原生的 X 服務端已經重寫來使用它們。參見 Linux 核心文件的"modedb default video mode support"。

7.3 GUI（圖形使用者介面）架構

這裡是 Wayland 環境上用於 GNOME 的著名的 GUI 架構軟體包。

軟體包	流行度	軟體包大小	說明
mutter	V:1, I:64	187	GNOME 的 mutter 視窗管理器 **[auto]**
xwayland	V:232, I:312	2388	執行在 wayland 之上的一個 X 服務端 **[auto]**
gnome-remote-desktop	V:35, I:214	1068	使用 PipeWire 的 GNOME 遠端桌面後臺守護程序（daemon）**[auto]**
gnome-tweaks	V:19, I:225	1200	GNOME 的高階配置設定
gnome-shell-extension-prefs	V:13, I:207	60	啟用/停用 GNOME 外殼擴充套件的工具

Table 7.2: 著名的 GUI 架構軟體包列表

這裡，"**[auto]**" 表示這些軟體包在 task-gnome-desktop 安裝時會自動安裝。

提示
gnome-tweaks 是一個不可缺少的配置工具。例如：

- 你能強制調整聲音音量，從“General（普通）”到"Over-Amplification（過分放大）"。
- 你能夠強迫"Caps" 鍵變成"Esc" 鍵，從"Keyboard & Mouse" -> "Keyboard" -> "Additional Layout Option".

提示
GNOME 桌面環境的詳細特徵能夠使用工具來配置，在按下 Super-鍵後，透過選擇"settings", "tweaks" 或"extensions" 來啟動配置。

7.4 GUI（圖形使用者介面）應用

現在在 Debian 上，有許多有用的 GUI 應用存在。如果在 GNOME 桌面環境中沒有相應功能的軟體，那麼安裝例如 scribus（KDE）這樣的軟體包是完全可以接受的。但安裝過多功能重複的軟體包，會使你的系統凌亂。

這裡是一份捕獲我眼球的 GUI（圖形使用者介面）程式列表。

7.5 字型

對於 Debian 的使用者，有許多有用的向量字型存在。使用者關注是怎樣避免冗餘，怎樣配置停用部分已經安裝的字型。此外，無用的字型選擇可以搞亂你的 GUI（圖形使用者介面）應用程式選單。

Debian 系統使用 FreeType 2.0 庫來柵格化許多向量字型格式，用於螢幕和列印：

軟體包	流行度	軟體包大小	類型	說明
evolution	V:29, I:236	486	GNOME	個人資訊管理（群組軟體和電子郵件）
thunderbird	V:55, I:119	224712	GTK	電子郵件客戶端 (Mozilla Thunderbird（雷鳥）)
kontact	V:1, I:12	2208	KDE	個人資訊管理（群組軟體和電子郵件）
libreoffice-writer	V:112, I:431	31473	LO	文書處理軟體
abiword	V:1, I:8	3542	GNOME	文書處理軟體
calligrawords	V:0, I:7	6097	KDE	文書處理軟體
scribus	V:1, I:17	30242	KDE	編輯 PDF 檔案的 desktop publishing 編輯器
glabels	V:0, I:3	1338	GNOME	標籤編輯器
libreoffice-calc	V:106, I:428	26008	LO	電子表格
gnumeric	V:3, I:15	9910	GNOME	電子表格
calligrasheets	V:0, I:5	11396	KDE	電子表格
libreoffice-impress	V:66, I:425	2645	LO	簡報
calligrastage	V:0, I:5	5339	KDE	簡報
libreoffice-base	V:27, I:125	5002	LO	資料庫管理
kexi	V:0, I:1	7118	KDE	資料庫管理
libreoffice-draw	V:69, I:426	10311	LO	向量圖形編輯器（繪圖）
inkscape	V:14, I:114	99800	GNOME	向量圖形編輯器（繪圖）
karbon	V:0, I:6	3610	KDE	向量圖形編輯器（繪圖）
dia	V:2, I:23	3908	GTK	流程圖和示意圖編輯器
gimp	V:51, I:252	19303	GTK	點陣圖圖形編輯器 (繪圖)
shotwell	V:17, I:252	6263	GTK	數碼照片管理器
digikam	V:2, I:10	293	KDE	數碼照片管理器
darktable	V:4, I:13	30563	GTK	攝影師的虛擬燈臺和暗房
planner	V:0, I:4	1394	GNOME	專案管理
calligraplan	V:0, I:2	19013	KDE	專案管理
gnucash	V:2, I:8	28929	GNOME	個人會計
homebank	V:0, I:2	1218	GTK	個人會計
lilypond	V:1, I:7	16092	-	音樂排版
kmymoney	V:0, I:2	13937	KDE	個人會計
librecad	V:1, I:15	8963	Qt 應用	計算機輔助設計 (CAD) 系統 (2D)
freecad	I:18	36	Qt 應用	計算機輔助設計 (CAD) 系統 (3D)
kicad	V:2, I:14	236002	GTK	電路圖和 PCB 設計軟體
xsane	V:12, I:143	2339	GTK	掃描器前段
libreoffice-math	V:50, I:429	1897	LO	數學方程/公式編輯器
calibre	V:7, I:29	63180	KDE	電子書轉換器和庫管理
fbreader	V:1, I:9	3783	GTK	電子書閱讀器
evince	V:88, I:312	942	GNOME	文件（pdf）閱讀器
okular	V:38, I:122	17728	KDE	文件（pdf）閱讀器
x11-apps	V:31, I:459	2460	單純的 X 應用	xeyes(1) 等。
x11-utils	V:197, I:564	651	單純的 X 應用	xev(1)、xwininfo(1) 等。

Table 7.3: 著名的的 GUI（圖形使用者介面）應用列表

- Type 1 (PostScript) 字型 使用三次 貝塞爾曲線 (差不多廢棄的格式)
- TrueType 字型 使用二次 貝塞爾曲線 (好的選擇格式)
- OpenType 字型 使用三次 貝塞爾曲線 (最佳選擇格式)

7.5.1 基礎字型

下面的編撰的表格希望幫助使用者選擇適當的向量字型，並清楚的理解排版指標相容（metric compatibility）和字形覆蓋。大部分字型覆蓋了所有拉丁、希臘和 Cyril 字元。最終選擇的啟用字型也受你的審美觀影響。這些字型能夠被用於螢幕顯示和紙張列印。

軟體包	流行度	大小	**sans**	**serif**	**mono**	字型註釋
fonts-cantarell	V:211, I:304	572	59	-	-	Cantarell (GNOME 3, 顯示)
fonts-noto	I:149	31	61	63	40	Noto fonts (Google, 有 CJK 的多語言)
fonts-dejavu	I:421	35	58	68	40	DejaVu (GNOME 2, MCM:Verdana, 擴充套件 Bitstream Vera)
fonts-liberation2	V:127, I:421	15	56	60	40	Liberation 字型 用於 LibreOffice (Red Hat, MCMATC)
fonts-croscore	V:20, I:41	5274	56	60	40	Chrome OS: Arimo, Tinos 和 Cousine (Google, MCMATC)
fonts-crosextra-carlito	V:21, I:135	2696	57	-	-	Chrome 作業系統: Carlito (Google, MCM:Calibri)
fonts-crosextra-caladea	I:132	347	-	55	-	Chrome 作業系統: Caladea (Google, MCM:Cambria) (只有拉丁字元)
fonts-freefont-ttf	V:73, I:218	14460	57	59	40	GNU FreeFont (擴充套件 URW Nimbus)
fonts-quicksand	V:117, I:433	392	56	-	-	Debian 任務桌面, Quicksand (顯示，只有拉丁字元)
fonts-hack	V:24, I:116	2508	-	-	40 P	給原始碼設計的一個字型 Hack (Facebook)
fonts-sil-gentiumplus	I:32	14345	-	54	-	Gentium SIL
fonts-sil-charis	I:27	6704	-	59	-	Charis SIL
fonts-urw-base35	V:162, I:462	15558	56	60	40	URW Nimbus (Nimbus Sans, Roman No. 9 L, Mono L, MCAHTC)
fonts-ubuntu	V:2, I:5	4339	58	-	33 P	Ubuntu 字型 (顯示)
fonts-terminus	V:0, I:3	452	-	-	33	Cool retro 終端字型
ttf-mscorefonts-installer	V:1, I:50	85	56?	60	40	下載微軟非開源字型 (見下)

Table 7.4: 著名的 TrueType 和 OpenType 字型列表

這裡：

- ”MCM” 表示” 與微軟提供的字型是排版指標相容的”

- ”MCMATC” 表示和” 微軟提供的字型： Arial, Times New Roman, Courier New 排版指標相容”
- ”MCAHTC” 表示” 和 Adobe 提供的字型: Helvetica, Times, Courier 排版指標相容”
- 在字型型別列的數字表示對相同磅數的字型與 M 字重的相對粗細程度（譯註： M 表示 Medium 適中，字型粗細程度的適中值）。
- 在 mono 字型型別列中的”P” 表示用於程式設計中，能夠清晰的區分”0”/”O” 和”1”/”I”/”l”。
- `ttf-mscorefonts-installer` 軟體包下載微軟的”Core fonts for the Web” 並安裝 Arial, Times New Roman, Courier New, Verdana, ... 。這些安裝的字型資料，是非開源的資料。

許多開源的拉丁字型，有 URW Nimbus 家族或 Bitstream Vera 的血統痕跡。

提示
如果你的語言環境所需要的字型沒有在上面的字型中涵蓋，請使用 aptitude 在”Tasks” -> ”Localization” 下面檢查任務軟體包列表。字型軟體包作為”Depends:” 或”Recommends:” 列出，在本地化任務軟體包裡面是首要候選軟體包。

7.5.2 字型柵格化

Debian 使用 FreeType 來柵格化字型。它的字型選擇架構由 Fontconfig 字型配置庫提供。

軟體包	流行度	大小	說明
`libfreetype6`	V:563, I:997	938	FreeType 字型柵格化庫
`libfontconfig1`	V:559, I:848	581	Fontconfig 字型配置庫
`fontconfig`	V:441, I:719	680	`fc-*`: Fontconfig 命令列命令
`font-manager`	V:2, I:8	1023	Font 管理器: Fontconfig GUI（圖形使用者介面）命令
`nautilus-font-manager`	V:0, I:0	37	Font 管理器 的 Nautilus 擴充套件

Table 7.5: 著名的字型環境和相關軟體包列表

提示
一些字型軟體包，比如說 `fonts-noto*`，會安裝太多的字型。你可以保持某些字型軟體包的安裝，但在通常使用的情況下停用。由於 Han unification 中日韓統一表意文字，一些 Unicode 碼點被期望有多個 字形，不希望的字形變體會被沒有配置的 Fontconfig 庫選擇。一個最令人討厭的情形是在 CJK 中日韓國家中的”U+3001 IDEOGRAPHIC COMMA” 和”U+3002 IDEOGRAPHIC FULL STOP”。你能夠使用 GUI（圖形使用者介面）字型管理器 (font-manager) 簡單的配置存在的字型來避免這個問題情形。

你也可以從命令列列出字型配置狀態。

- 使用“`fc-match(1)`”檢視 fontconfig 的預設字型
- 使用“`fc-list(1)`”檢視所有可用的 fontconfig 字型

你能夠從文字編輯器配置字型配置狀態，但這是瑣碎的。參見 `fonts.conf`(5)。

7.6 沙盒

Linux 上大部分 GUI（圖形使用者介面）應用在非 Debian 的源上，是以二進位制格式存在。

- AppImage -- 任何地方執行的 Linux 應用
- FLATHUB -- Linux 應用，就是這裡
- snapcraft -- Linux 應用商店

警告
從這些站點來的二進位制軟體包，有可能包括私有的非開源軟體。

對使用 Debian 的自由軟體的狂熱愛好者，這些二進位制格式的分發，有一些存在的理由。因為這能夠得到一個乾淨的庫集合，由 Debian 提供的庫和由每個應用程式相應的上游開發者使用的庫，獨立開來。

執行外部二進位制的固有風險，能夠使用 沙盒環境 減少，它有現代 Linux 安全特性的手段。（參見節 4.7.5）。

- 對於 AppImage 和一些上游站點來的二進位制，在 手工配置 後的 firejail 裡執行。
- 對於從 FLATHUB 來的二進位制，在Flatpak 裡執行它們。（不需要手工配置。）
- 對於從 snapcraft 來的二進位制，在 Snap 裡面執行它們。（不需要手工配置。和後臺守護程序（daemon）相容。）

`xdg-desktop-portal` 軟體包為通用的桌面特性提供一個標準的 API。參見 xdg-desktop-portal (flatpak) 和 xdg-desktop-portal (snap)。

軟體包	流行度	大小	說明
`flatpak`	V:64, I:69	7499	Flatpak 桌面應用程式配置框架
`gnome-software-plugin-flatpak`	V:20, I:29	246	GNOME 軟體管理器的 Flatpak 支援
`snapd`	V:64, I:69	62774	啟用 snap 軟體包的後臺守護程序（daemon）和工具
`gnome-software-plugin-snap`	V:1, I:2	117	GNOME 軟體管理器的 Snap 支援
`xdg-desktop-portal`	V:296, I:385	1936	Flatpak 和 Snap 的桌面整合門戶
`xdg-desktop-portal-gtk`	V:265, I:384	715	gtk (GNOME) 的 xdg-desktop-portal 後端
`xdg-desktop-portal-kde`	V:49, I:69	1438	Qt (KDE) 的 xdg-desktop-portal 後端
`xdg-desktop-portal-wlr`	V:0, I:3	131	wlroots (Wayland) 的 xdg-desktop-portal 後端
`firejail`	V:1, I:4	1771	和 AppImage 一起使用的 SUID 安全沙盒程式 firejail

Table 7.6: 著名的沙盒環境和相關軟體包列表

這個沙盒環境技術和在智慧手機作業系統上的應用程式非常相像，這裡的應用程式也是在資源訪問受到控制下執行的。

一些大的 GUI（圖形使用者介面）應用程式，比如說 Debian 上的網頁瀏覽器，也在內部使用了沙盒環境技術，這樣讓它們安全性更好。

軟體包	流行度	大小	協議	說明
gnome-remote-desktop	V:35, I:214	1068	RDP	GNOME 遠端桌面 服務端
xrdp	V:22, I:25	3194	RDP	xrdp, 遠端桌面協議 (RDP) 伺服器
x11vnc	V:6, I:24	2107	RFB (VNC)	x11vnc, 遠端幀快取協議 (VNC) 伺服器
tigervnc-standalone-server	V:4, I:15	2768	RFB (VNC)	TigerVNC, 遠端幀快取協議 (VNC) 伺服器
gnome-connections	V:0, I:1	1267	RDP, RFB (VNC)	GNOME 遠端桌面客戶端
vinagre	V:2, I:72	4249	RDP, RFB (VNC), SPICE, SSH	Vinagre: GNOME 遠端桌面客戶端
remmina	V:14, I:71	884	RDP, RFB (VNC), SPICE, SSH, ...	Remmina: GTK 遠端桌面客戶端
krdc	V:1, I:17	3873	RDP, RFB (VNC)	KRDC: KDE 遠端桌面客戶端
guacd	V:0, I:0	80	RDP, RFB (VNC), SSH / HTML5	Apache Guacamole: 無客戶端的遠端桌面閘道器 (HTML5)
virt-viewer	V:5, I:52	1284	RFB (VNC), SPICE	虛擬機器管理器 下的客戶機作業系統的 GUI 顯示客戶端

Table 7.7: 著名的遠端訪問服務端列表

7.7 遠端桌面

7.8 X 服務端連線

有幾種方法從遠端主機上的應用連線到 X 服務端 (包括本地主機的 `xwayland`)。

軟體包	流行度	大小	指令	說明
openssh-server	V:726, I:817	1804	sshd 使用選項 `X11-forwarding`	SSH 服務端 (安全)
openssh-client	V:857, I:995	4959	`ssh -X`	SSH 客戶端 (安全)
xauth	V:164, I:960	81	`xauth`	X 授權檔案工具
x11-xserver-utils	V:298, I:524	568	`xhost`	X 服務端訪問控制

Table 7.8: 連線到 X 伺服器的方式

7.8.1 X 服務端本地連線

使用 X 核心協議的本地應用，能夠透過本地 UNIX 域名套接字進行本地連線，來訪問本地的 X 服務端。這可以透過擁有 access cookie 的授權檔案來授權。授權檔案的位置透過”$XAUTHORITY” 環境變數確定，X 顯示透過”$DISPLAY” 環境變數確定。由於這些環境變數通常會被自動設定，不需要另行指定。例如，下面的”gitk”。

```
username $ gitk
```

注
對於 xwayland, XAUTHORITY 有類似"/run/user/1000/.mutter-Xwaylandauth.YVSU30" 的值。

7.8.2 X 服務端遠端連線

使用 X 核心協議的遠端應用訪問本地的 X 伺服器顯示，由 X11 轉發特性支援。

- 在本地主機中開啟一個 gnome 終端。
- 透過下列命令，執行帶 -X 選項的 ssh(1)，建立與遠端站點的連線。

```
localname @ localhost $ ssh -q -X loginname@remotehost.domain
Password:
```

- 透過下列命令，在遠端站點執行一個 X 應用程式，例如“gitk”。

```
loginname @ remotehost $ gitk
```

這個方法可以顯示來自遠端 X 客戶端的輸出，相當於它是通過一個本地 UNIX 域名 socket 進行本地的連線。

參見介紹 SSH/SSHD 的節 6.3 。

警告
由於安全的原因，在 Debian 系統上，遠端 TCP/IP 連線到 X 服務端，是預設被停用的。不要透過簡單的設定"xhost +" 來啟用它們。如果能夠避免的話，也不要啟用 XDMCP 連線。

7.8.3 X 服務端 chroot 連線

在同一個環境下 (比如 chroot)，使用 X 核心協議的應用訪問同一主機的 X 服務端，授權檔案無法訪問，能夠使用 xhost 進行安全的授權，透過使用 User-based access ，例如下面的"gitk"。

```
username $ xhost + si:localuser:root ; sudo chroot /path/to
# cd /src
# gitk
# exit
username $ xhost -
```

7.9 剪貼簿

剪貼文字到剪貼簿，參見節 1.4.4。

剪貼影象到剪貼簿，參見節 11.6。

一些命令列的命令也能操作字元剪貼簿 (主要鍵和剪貼簿)。

軟體包	流行度	軟體包大小	當前目標	說明
xsel	V:9, I:42	55	X	X 選擇的命令列介面 (剪貼簿)
xclip	V:12, I:62	64	X	X 選擇的命令列介面 (剪貼簿)
wl-clipboard	V:2, I:14	162	Wayland	wl-copy wl-paste: Wayland 剪貼簿 的命令列介面
gpm	V:10, I:12	521	Linux 控制檯	在 Linux 控制檯上捕獲滑鼠事件的後臺守護程序 (daemon)

Table 7.9: 操作字元剪貼簿相關程式列表

Chapter 8

I18N 和 L10N

一個應用軟體的 多語言化 (M17N) 或本地語言支援，通過 2 個步驟完成。

- 國際化 (I18N): 使一個軟體能夠處理多個語言環境。
- 本地化 (L10N)：使一個軟體處理一個特定的語言環境。

提示
在 multilingualization (多語言化)、internationalization (國際化) 和 localization (本地化) 中，有 17, 18, 或 10 個字母在”m” 和”n”, ”i” 和”n”, 或”l” 和”n” 中間，它們相應表示為 M17N, I18N 和 L10N。細節參見 國際化和本地化 。

8.1 語言環境

程式支援國際化的行為，是透過配置環境變數”$LANG” 來支援本地化。語言環境的實際支援，依賴 `libc` 庫提供的特性，並要求安裝 `locales` 或 `locales-all` 軟體包。`locales` 軟體包需要被適當的初始化。

如果 `locales` 或 `locales-all` 軟體包均沒有安裝，支援語言環境的特性丟失，系統使用 US 英語訊息，並按 **ASCII** 處理資料。這個行為和”$LANG” 被設定為”LANG=”、”LANG=C” 或”LANG=POSIX” 相同。

GNOME 和 KDE 等現代軟體是多語言的。他們透過處理 UTF-8 資料來實現國際化，並透過 `gettext`(1) 架構提供翻譯資訊來本地化。翻譯資訊可以由獨立的本地化軟體包來提供。

目前的 Debian 桌面 GUI (圖形使用者介面) 系統通常在 GUI 環境中設定語言環境為”LANG=xx_YY.UTF-8”。這裡，”xx” 是 ISO 639 語言程式碼 ，”YY” 是 ISO 3166 國家地區程式碼。這些值由配置桌面的 GUI 對話方塊來設定，並改變程式的行為。參見節 1.5.2

8.1.1 UTF-8 語言環境的基本原理

最簡單的文字資料表達是 **ASCII**，它對英語是足夠的，少於 127 個字元 (使用 7 位描述)。

即使純英文文字也可能包含非 ASCII 字元，例如微微卷曲的左右引號在 ASCII 中是不可用的。

```
b’’“b’’double quoted textb’’”b’’ is not ”double quoted ASCII”
b’’‘b’’single quoted textb’’’b’’ is not ’single quoted ASCII’
```

為了支援更多字元，許多字符集和編碼系統被用來支援多語言。(參見表格 11.2)。

Unicode 字符集可以用 21 位碼點範圍來顯示幾乎所有人類已知的字元 (例如，十六進位制的 0 到 10FFFF)。

文字編碼系統 UTF-8 將 Unicode 碼點適配到一個合理的 8 位資料流，並大部分相容 ASCII 資料處理系統。這個使 **UTF-8** 作為現代推薦的選擇。**UTF** 表示 Unicode 轉換格式 (Unicode Transformation Format)。當 ASCII 純文字資料轉換為 UTF-8 資料，它有和原始 ASCII 完全一樣的內容和大小。所以配置 UTF-8 語言環境不會有任何丟失。

在 UTF-8 語言環境下相容的應用程式，你可以顯示和編輯外語文字資料，在所要求的字型和輸入法安裝和啟用後。例如在"LANG=fr_FR.UTF-8" 語言環境下，gedit(1) (GNOME 桌面的文字編輯器) 能夠顯示和編輯中文字元文字資料，而顯示的選單是法語。

提示
新標準的"en_US.UTF-8" 和老標準的"C"/"POSIX" 語言環境都使用標準的 US 英文訊息，它們在排序等方面有細微的不同。在維護老的"C" 本地行為時，如果你不僅想處理 ASCII 字元，同時還想優雅的處理 UTF-8 編碼的字元，在 Debian 上使用非標準的"C.UTF-8" 語言環境。

注
一些程式在支援 I8N 後會消耗更多的記憶體。這是因為它們為了速度優化，而在內部使用 UTF-32(UCS4) 來支援 Unicode，並且每個獨立於語言環境所選的 ASCII 字元資料都會消耗 4 個位元組。再一次地，使用 UTF-8 語言環境並不會使你損失什麼。

8.1.2 語言環境的重新調配

為了讓系統訪問某一語言環境，語言環境資料必須從語言環境資料庫中編譯。

locales 軟體包 沒有包含預先編譯的語言環境資料。你需要按下面的方法配置：

```
# dpkg-reconfigure locales
```

該過程包含 2 個步驟。

1. 選擇所有需要的語言環境資料編譯為二進位制形式。(請確認至少包含一個 UTF-8 語言環境)
2. 透過建立“/etc/default/locale”來設定系統預設的語言環境值給 PAM 使用 (參見節 4.5)。

設定在"/etc/default/locale" 裡的系統範圍的預設語言環境，可以被 GUI (圖形使用者介面) 應用程式的 GUI 配置覆蓋。

注
所使用的確切傳統編碼系統可以透過“/usr/share/i18n/SUPPORTED”來確認。因此，"LANG=en_US" 是"LANG=en_US.ISO-8859-1"。

---

locales-all 軟體包有所有預編譯的語言環境資料，但是不建立"/etc/default/locale"，你可能還需要安裝 locales 軟體包。

提示
一些 Debian 系發行版的 locales 軟體包，包含有所有語言環境的預先編譯好的語言環境資料。為了模擬這樣的系統環境，你需要同時在 Debian 安裝 locales 和 locales-all 軟體包。

8.1.3 檔名編碼

對於跨平臺的資料交換 (參見節 10.1.7), 你需要使用特殊的編碼掛載檔案系統. 舉個例子，不使用選項時，mount(8) 假設 vfat 檔案系統 使用 CP437. 你需要給檔名提供明確的掛載選項來使用UTF-8 或 CP932.

注
在 GNOME 這類的現代桌面環境下，當自動掛載一個熱拔插 U 盤時，你可以提供這樣的掛載選項。右擊桌面上的圖示，點選”Drive” , ”Setting”, 輸入”utf8” 到”Mount options:”. 當這個 U 盤下次掛載時，UTF-8 就可以了。

注
如果你在升級一個系統，或者從老的非 UTF-8 系統遷移磁碟，非 ASCII 字元的檔名也許是使用老舊的 ISO-8859-1 或 eucJP 來編碼. 請尋求文字轉換工具把他們轉換到 UTF-8. 參見節 11.1.

在預設情況下，Samba 對新的客戶端 (Windows NT, 200x, XP) 使用 Unicode，但對老的客戶端 (DOS 和 Windows 9x/Me) 使用 CP850. 可以在”`/etc/samba/smb.conf`” 檔案裡面，使用”`dos charset`” 來改變老客戶端的這個預設編碼。比如說，CP932 表示為日語。

8.1.4 本地化資訊和翻譯文件

在 Debian 系統中顯示的許多文件和文字資訊有翻譯存在，比如錯誤資訊、標準程式輸出、選單和手冊頁。GNU gettext(1) 指令工具鏈是大部分翻譯活動的後端工具。

`aptitude`(8) 裡，”Tasks” → ”Localization” 提供一個有用的二進位制包擴充列表，給應用程式增加本地化資訊和提供翻譯文件。

舉個例子，你可以安裝 `manpages-`*`LANG`* 包來獲得本地化 man 手冊頁資訊。從”`/usr/share/man/it/`” 來讀取 *programname* 義大利語的 man 手冊頁, 執行下面的操作。

```
LANG=it_IT.UTF-8 man programname
```

透過 `$LANGUAGE` 環境變數，GNU gettext 能夠適應翻譯語言的優先順序列表。例如：

```
 $ export LANGUAGE="pt:pt_BR:es:it:fr"
```

獲取更多資訊，參見 `info gettext` ，閱讀”The LANGUAGE variable” 章節。

8.1.5 語言環境的影響

`sort`(1) 和 `ls`(1) 的字元排序受語言環境 locale 影響。匯出 `LANG=en_US.UTF-8`,排序用字典 `A->a->B->b...->Z->z` 順序，而匯出 `LANG=C.UTF-8`，排序使用 ASCII 二進位制 `A->B->...->Z->a->b...` 順序。

`ls`(1) 的日期格式受語言環境影響 (參見節 9.3.4)。

`date`(1) 程式的日期格式受語言環境的影響。例如：

```
 $ unset LC_ALL
 $ LANG=en_US.UTF-8 date
Thu Dec 24 08:30:00 PM JST 2023
 $ LANG=en_GB.UTF-8 date
Thu 24 Dec 20:30:10 JST 2023
 $ LANG=es_ES.UTF-8 date
jue 24 dic 2023 20:30:20 JST
 $ LC_TIME=en_DK.UTF-8 date
2023-12-24T20:30:30 JST
```

不同語言環境的數字標點不一樣。比如說，英語語言環境中, 一千點一顯示為”`1,000.1`”，而在德語語言環境中, 它顯示為”`1.000,1`”。你可以在電子表格程式裡面看到這個不同。

”`$LANG`” 環境變數的每一個細節特徵能夠透過設定”`$LC_*`” 變數來覆蓋。這些環境變數又能夠透過設定“`$LC_ALL`” 變數而被再次覆蓋。細節參見 `locale`(7) man 手冊頁。除非你有強烈的理由建立複雜的配置，請遠離他們並只使用”`$LANG`” 變數來設定一個 UTF-8 語言環境。

8.2 鍵盤輸入

8.2.1 Linux 控制檯和 X 視窗的鍵盤輸入

Debian 系統可以使用 keyboard-configuration 和 console-setup 軟體包調配多個國際化鍵盤佈局。

```
# dpkg-reconfigure keyboard-configuration
# dpkg-reconfigure console-setup
```

對於 Linux 控制檯和 X 視窗系統，這將更新在"/etc/default/keyboard" 和"/etc/default/console-setup" 裡的配置引數。這個也會配置 Linux 控制檯字型。許多非 ASCII 字元，包括許多歐洲語言使用的重音字元，可以使用死鍵、AltGr 鍵 和 組合鍵 來輸入它們。

8.2.2 Wayland 鍵盤輸入

Wayland 桌面系統上的 GNOME, 節 8.2.1 不支援非英語的歐洲語言。IBus 不僅支援亞洲語言，也支援歐洲語言。GNOME 桌面環境的軟體包依賴關係透過"gnome-shell" 推薦"ibus" 。"ibus" 的程式碼已經更新整合 setxkbmap 和 XKB 選項功能。對多語言鍵盤輸入，你需要從"GNOME Settings" 或"GNOME Tweaks" 配置 ibus。

注
如果 ibus 啟用，即使在基於傳統的 X 的桌面環境下，透過 setxkbmap 配置的傳統的 X 鍵盤，也會被 ibus 覆蓋。你能夠停用安裝的 ibus，使用 im-config 設定輸入法為"None"。更多資訊，參見 Debian 維基：鍵盤。

8.2.3 IBus 支援的輸入法

因 GNOME 桌面環境透過"gnome-shell" 推薦"ibus"，"ibus" 對於輸入法來說，是一個好的選擇。

輸入多種語言到應用程式的處理流程如下：

```
Keyboard                                          Application
   |                                                  ^
   |                                                  |
   +-> Linux kernel ->  Input method (ibus)  -> Gtk, Qt, X, Wayland
                          +-- Engine--+
```

下面是 IBus 和它的引擎軟體包列表。

注
對於中文，"fcitx5" 可以是一個替代的輸入法框架。對於 Emacs 的狂熱愛好者，"uim" 可以是一個替代。無論哪種方式，你可能需要用 im-config 做一些額外的手工配置。像"kinput2" 這類老的 輸入法 任然在 Debian 倉庫中存在，但是不推薦用到現代環境中。

8.2.4 一個日語的例子

我發現在英語環境 ("en_US.UTF-8") 下啟動日文輸入法非常有用. 下面是在 Wayland 上的 GNOME 下使用 IBus 的做法:

1. 安裝日文輸入法軟體包 ibus-mozc (或 ibus-anthy)，以及 im-config 等推薦的軟體包。
2. 選擇"Settings" → "Keyboard" → "Input Sources" → 在"Input Sources" 中單擊"+" → "Japanese" → "Japanese mozc (or anthy)"，然後單擊"Add"。如果它沒有被啟用。

軟體包	流行度	大小	支援的語言環境
ibus	V:192, I:240	1723	使用 dbus 的輸入方式框架
ibus-mozc	V:1, I:3	935	日文
ibus-anthy	V:0, I:1	8856	同上
ibus-skk	V:0, I:0	242	同上
ibus-kkc	V:0, I:0	210	同上
ibus-libpinyin	V:1, I:3	2760	中文 (zh_CN)
ibus-chewing	V:0, I:0	424	中文 (zh_TW)
ibus-libzhuyin	V:0, I:0	40987	中文 (zh_TW)
ibus-rime	V:0, I:0	73	中文 (zh_CN/zh_TW)
ibus-cangjie	V:0, I:0	119	中文 (zh_HK)
ibus-hangul	V:0, I:2	264	韓文
ibus-libthai	I:0	90	泰文
ibus-table-thai	I:0	58	泰文
ibus-unikey	V:0, I:0	318	越南語
ibus-keyman	V:0, I:0	138	多語言：Keyman 引擎，超過 2000 種語言
ibus-table	V:0, I:1	2176	IBus 表引擎
ibus-m17n	V:0, I:1	389	多語言：印度語、阿拉伯語和其它
plasma-widgets-addons	V:48, I:98	1992	Plasma 5 的額外元件，包含有鍵盤指示器

Table 8.1: IBus 和它的引擎軟體包列表

3. 你可以選擇許多輸入源。
4. 重新登入使用者帳號。
5. 右鍵單擊 GUI 工具條圖示，設定每一個輸入源。
6. 使用 SUPER-SPACE 在安裝的輸入法之間進行切換. (SUPER 鍵通常是 Windows 鍵.)

提示
如果你希望在日本物理鍵盤（shift-2 按鍵刻有有一個″雙引號標記）上訪問只有字母表的鍵盤環境，在上面的過程中選擇”Japanese”。你能夠在物理的“US”鍵盤（shift-2 按鍵刻有有一個 @ 標記）上使用”Japanese mozc (或 anthy)” 輸入日文。

- `im-config`(8) 的使用者介面選單入口是”Input method”。
- 此外，從使用者的 shell 來執行”`im-config`”。
- `im-config`(8) 如果指令是從 root 帳號執行的表現會有所不同。
- `im-config`(8) 讓最佳的輸入法作為系統預設而不需要使用者干預。

8.3 顯示輸出

Linux 控制檯只能顯示有限的字元。(你需要使用特殊的終端程式，例如 `jfbterm`(1)，從而在非 GUI 控制檯中顯示非歐洲語言。)

只要需要的字型安裝並被啟用，GUI (圖形使用者介面) 環境 (章 7) 能夠顯示任意 UTF-8 字元。(原始字型資料的編碼會被處理，並對使用者來說是透明的。)

8.4 東亞環境下寬度有歧義的字元

在東亞語言環境下，方框繪製、希臘字元和西里爾字元可能會顯示得比你預期的樣子更寬，這樣會導致終端輸出排列不再整齊（參見 Unicode 標準附錄 #11）。

您可以繞過這個問題：

- `gnome-terminal`：首選項 → 配置檔案 → 配置名 → 相容性 → 寬度有歧義的字元 → 窄
- `ncurses`：設定環境變數 `export NCURSES_NO_UTF8_ACS=0`。

Chapter 9

系統技巧

這裡，描述調配和管理系統的基本技巧，大部分在控制檯操作。

9.1 控制檯技巧

有一些工具程式來幫助你的控制檯活動。

軟體包	流行度	大小	說明
mc	V:49, I:212	1542	參見節 1.3
bsdutils	V:519, I:999	356	script (1) 命令來記錄終端會話
screen	V:72, I:234	1003	VT100/ANSI 終端模擬器混合複用的終端
tmux	V:42, I:146	1114	終端複用的備選方案（使用“Control-B”代替）
fzf	V:4, I:15	3648	模糊的文字查詢器
fzy	V:0, I:0	54	模糊的文字查詢器
rlwrap	V:1, I:15	330	具備 readline 特徵的命令列封裝
ledit	V:0, I:11	331	具備 readline 特徵的命令列封裝
rlfe	V:0, I:0	45	具備 readline 特徵的命令列封裝
ripgrep	V:4, I:18	5152	在原始碼樹中快速遞迴搜尋字串，並自動過濾

Table 9.1: 支援控制檯活動的程式列表

9.1.1 清晰的記錄 shell 活動

簡單地使用 script(1)（參見節 1.4.9）記錄 shell 活動會產生一個有控制字元的檔案。這些控制字元可以按下面的方式，使用 col(1) 去掉。

```
$ script
Script started, file is typescript
```

做些操作……按 Ctrl-D 退出 script.

```
$ col -bx < typescript > cleanedfile
$ vim cleanedfile
```

有替代的方式來記錄 shell 活動：

- 使用 tee (在 initramfs 的啟動過程中可用)：

```
$ sh -i 2>&1 | tee typescript
```

- 使用 gnome-terminal 增加行緩衝，用捲軸檢視。
- 使用 screen 和”^A H” (參見節 9.1.2) 來進行控制檯記錄。
- 使用 vim 輸入”:terminal” 進入終端模式。使用”Ctrl-W N” 從終端模式退出到普通模式。使用”:w typescript” 將快取寫到一個檔案。
- 使用 emacs 和”M-x shell”, ”M-x eshell”, 或”M-x term” 來進入記錄控制檯。使用”C-x C-w” 將快取寫到檔案。

9.1.2 screen 程式

screen(1) 不但允許一個終端視窗執行多個程序，還允許遠端 **shell** 程序支援中斷的連線. 這裡是一個典型的 screen(1) 使用場景.

1. 登入到一個遠端機器。
2. 在單個控制檯上啟動 screen。
3. 使用 ^A c (”Control-A” 接著”c”) 在 screen 中建立的視窗執行多個程式.
4. 按 ^A n (”Control-A” 接著”n”) 來在多個 screen 視窗間轉換.
5. 突然，你需要離開你的終端，但你不想丟掉正在做的工作，而這些工作需要連線來保持。
6. 你可以通過任何方式分離 screen 會話。
   - 殘忍地拔掉你的網路連線
   - 輸入 ^A d (”Control-A” 接著”d”) 並手工從遠端連線退出登入
   - 輸入 ^A DD (”Control-A” 接著”DD”) 分離 screen 並退出登入
7. 你重新登入到同一個遠處主機 (即使從不同的終端)。
8. 使用”screen -r” 啟動 screen.
9. screen 魔術般的重新附上先前所有的 screen 視窗和所有在活動執行的程式.

提示
對於撥號或者按包計費的網路連線，你可以通過 screen 節省連線費用，應為你可以在斷開連線時讓一個程序繼續執行，當你稍後再次連線時重新附上它。

在 screen 會話裡, 除了指令按鍵外的所有鍵盤輸入都會被髮送到當前視窗。screen 所有指令按鍵是通過 ^A (”Control-A”) 加單個鍵 [加任何參數] 來輸入. 這裡有一些重要的指令按鍵需要記住。

細節參見 screen(1).

參見 tmux(1) ，瞭解替代命令的功能。

鍵綁定功能	說明
^A ?	顯示幫助螢幕（顯示鍵繫結）
^A c	建立一個新的視窗並切換到該視窗
^A n	到下一個視窗
^A p	到前一個視窗
^A 0	到 0 號視窗
^A 1	到 1 號視窗
^A w	顯示視窗列表
^A a	作為鍵盤輸入傳送 Ctrl-A 到當前視窗
^A h	把當前視窗的硬拷貝寫到一個檔案
^A H	開始/結束當前視窗到檔案的記錄
^A ^X	鎖定終端 (密碼保護)
^A d	從終端分離 screen 會話
^A DD	分離 screen 會話並退出登入

Table 9.2: screen 鍵繫結列表

9.1.3 在目錄間遊走

在節 1.4.2，2 個技巧允許快速在目錄間遊走，在 $CDPATH 和 mc 描述。

如果你使用模糊文字過濾程式，你能夠不輸入精準路徑。對於 fzf 軟體包，在 ~/.bashrc 裡面包括下列內容。

```
FZF_KEYBINDINGS_PATH=/usr/share/doc/fzf/examples/key-bindings.bash
if [ -f $FZF_KEYBINDINGS_PATH ]; then
  . $FZF_KEYBINDINGS_PATH
fi
FZF_COMPLETION_PATH=/usr/share/doc/fzf/examples/completion.bash
if [ -f $FZF_COMPLETION_PATH ]; then
  . $FZF_COMPLETION_PATH
fi
```

例如：

- 你能夠最小化的操作跳入非常深的子目錄。你首先輸入”cd **” 後按 Tab。然後你將被提示候選路徑。輸入部分路徑字串，比如 s/d/b foo，將會縮窄候選路徑。透過有游標和回車鍵的 cd，你選擇將要使用的路徑。
- 你可以用最小化的操作，從命令歷史裡面選擇一個命令。在命令列提示符下按 Ctrl-R。然後你將被提示候選的命令。輸入部分命令字串，比如 vim d，將會縮窄候選項。使用游標和回車鍵選擇將要使用的命令。

9.1.4 Readline 封裝

一些命令，比如 /usr/bin/dash，它缺少命令列歷史編輯能力，但在 rlwrap 或它的等價物下執行就能夠透明的增加這樣的功能。

```
$ rlwrap dash -i
```

這提供一個便利平臺來測試 dash 的細微之處，使用類似 bash 的友好環境。

9.1.5 掃描原始碼樹

在 ripgrep 軟體包中的 rg(1) 命令，在掃描原始碼樹的典型場景中，提供了一個比 grep(1) 命令更快速的替代。它充分利用了現代多核 CPU，並自動使用適當的過濾器來忽略一些檔案。

9.2 定製 vim

在你透過節 1.4.8 學習基本的 vim(1) 後, 請閱讀 Bram Moolenaar 的”Seven habits of effective text editing (2000)” 來理解 vim 應當怎樣被使用。

注意
沒有非常好的理由，請不要嘗試改變預設的鍵繫結。

9.2.1 用內部特性定製 vim

vim 的行為能夠被顯著的改變，透過 Ex-模式的命令，啟用它的內部特性，比如”set ...” 來設定 vim 選項。

這些 Ex-模式的命令，能夠在使用者的 vimrc 檔案裡面包括，傳統的”~/.vimrc” 或 git 友好的”~/.vim/vimrc”。這裡有一個非常簡單的例子[1]:

```
colorscheme murphy              " from /usr/share/vim/vim??/colors/*.vim
filetype plugin indent on       " filetype aware behavior
syntax enable                   " Syntax highlight
set spelllang=en_us             " Spell check language as en_us
set spell                       " Enable spell check
set autoindent                  " Copy indent from current line
set smartindent                 " More than autoindent (Drop/Pop after {/})
set nosmarttab                  " <Tab>-key always inserts blanks
set backspace=indent,eol,start  " Back space through everything
set laststatus=2                " Always show status line
set statusline=%<%f%m%r%h%w%=%y[U+%04B]%2l/%2L=%P,%2c%V
highlight RedundantSpaces ctermbg=red guibg=red " highlight tailer spaces red color
```

9.2.2 用外部軟體包定製 vim

透過簡單定製，即安裝 vim-scripts 軟體包，並附加下面的內容到使用者的 vimrc 檔案，能夠啟用 secure-modelines 和傳統的 IDE。

```
packadd! secure-modelines
packadd! winmanager
let mapleader = ' '
" Toggle paste mode with <SPACE>p for Vim (no need for Nvim)
set pastetoggle=<leader>p
" IDE-like UI for files and buffers with <space>w
nnoremap <leader>w          :WMToggle<CR>
" Use safer keys <C-?> for moving to another window
nnoremap <C-H>          <C-W>h
nnoremap <C-J>          <C-W>j
nnoremap <C-K>          <C-W>k
nnoremap <C-L>          <C-W>l
" Record key MACRO with <ESC>qq ... <ESC>q and replay it with Q
nnoremap Q          @q
" Y works like D without delete (default for Nvim)
nnoremap Y y$
```

為了使上面的按鍵繫結正確地執行，終端程式需要配置：Backspace-鍵產生”ASCII DEL”、Delete-鍵產生”Escape sequence”。

1更多精心製作的定製例子：”Vim Galore”, ”sensible.vim”, ”#vim Recommendations” ...

新的原生 Vim 軟體包系統同"git" 和"git submodule" 順利的工作。一個這樣的配置例子能夠在 我的 git 倉庫: dot-vim 找到。本質上是這樣做的:

- 透過使用"git" 和"git submodule",最新的擴充套件軟體包,比如說"*name*",會被放到 ~/.vim/pack/*/opt/*name* 和類似的地方。
- 透過增加 :packadd! *name* 行到使用者的 vimrc 檔案，這些軟體包被放到 runtimepath。
- Vim 在它的初始化時載入這些軟體包到 runtimepath。
- 在它初始化的最後，安裝文件的標籤被更新，使用"helptags ALL"。

更多資訊，請使用"vim --startuptime vimstart.log" 啟動 vim 來檢查實際的執行順序和每一個步驟消耗的時間。

下面能夠發現有趣的外部外掛軟體包:

- Vim - 無所不在的文字編輯器 -- Vim 和 vim 指令碼的官方上游站點
- VimAwsome -- Vim 外掛列表
- vim-scripts -- Debian 軟體包: 一個 vim 指令碼的收集

是相當迷惑的看到這麼多的方式[2] 來管理和載入這些外部的軟體包到 vim。檢查原始的資訊是最好的方法。

按鍵	資訊
:help package	解釋 vim 軟體包機制
:help runtimepath	解釋 runtimepath 機制
:version	內部狀態，包括 vimrc 檔案的候選
:echo $VIM	環境變數"$VIM" 用來定位 vimrc 檔案的路徑
:set runtimepath?	列出用來搜尋所有執行時支援檔案的目錄
:echo $VIMRUNTIME	環境變數"$VIMRUNTIME" 用來定位大量系統提供的執行時支援檔案

Table 9.3: vim 的初始化資訊

9.3 資料記錄和展示

9.3.1 日誌後臺背景程式（daemon）

許多傳統的程式在"/var/log/" 目錄下用文字檔案格式記錄它們的活動.

在一個產生很多日誌檔案的系統上，用 logrotate(8) 來簡化日誌檔案的管理。

許多新的程式使用 systemd-journald(8) 日誌服務的二進位制檔案格式來記錄它們的活動,在"/var/log/journal" 目錄下。

你能夠從 shell 指令碼記錄資料到 systemd-journald(8) 日誌，使用 systemd-cat(1) 命令。

參見節 3.4 和節 3.3.

9.3.2 日誌分析

這裡是主要的日誌分析軟體 ("~Gsecurity::log-analyzer" 在 aptitude(8) 中).

注
CRM114 提供語言架構來寫模糊過濾器，使用了 TRE 正規表達式庫 。它主要在垃圾郵件過濾器中使用，但也能夠用於日誌分析。

[2] vim-pathogen 也很流行。

軟體包	流行度	大小	說明
logwatch	V:12, I:13	2328	用 Perl 寫的日誌分析軟體，有好的輸出
fail2ban	V:99, I:112	2126	禁用造成多個認證錯誤的 IP
analog	V:3, I:96	3739	web 伺服器日誌分析
awstats	V:7, I:11	6928	強大和特性全面的 web 伺服器日誌分析
sarg	V:1, I:1	845	生成 squid 分析報告
pflogsumm	V:1, I:4	109	Postfix 日誌條目概要
fwlogwatch	V:0, I:0	481	防火牆日誌分析軟體
squidview	V:0, I:0	189	監控和分析 squid access.log 檔案
swatch	V:0, I:0	99	有正規表達式、高亮和曲線的日誌檔案檢視器
crm114	V:0, I:0	1119	Controllable Regex Mutilator 和垃圾郵件過濾 (CRM114)
icmpinfo	V:0, I:0	44	解釋 ICMP 資訊

Table 9.4: 系統日誌分析軟體列表

9.3.3 定製文字資料的顯示

儘管例如 more(1) 和 less(1) 這樣的分頁程式（參見節 1.4.5）和用於高亮和格式的自定義工具（參見節 11.1.8）可以漂亮地顯示文字資料，但通用的編輯器（參見節 1.4.6）是用途最廣的，且可定製性最高。

提示
對於 vim(1) 和它的分頁模式別名 view(1)，“:set hls”可以啟用高亮搜尋。

9.3.4 定製時間和日期的顯示

“ls -l”命令預設的時間和日期顯示格式取決於語言環境（相關的值參見節 1.2.6）。“$LANG”變數將被首先考慮，但它會被匯出的“$LC_TIME”或”$LC_ALL” 環境變數覆蓋。

每個語言環境實際的預設顯示格式取決於所使用的 C 標準庫的版本（libc6 軟體包），也就是說，不同的 Debian 發行版有不同的預設情況。對於 iso-formates，參見 ISO 8601。

如果你真的想自定義超出語言環境的時間和日期顯示格式，你應該通過“--time-style”參數或“$TIME_STYLE”的值來設定時間樣式值（參見 ls(1)、date(1)、“info coreutils 'ls invocation'”）。

時間樣式值	語言環境	時間和日期顯示
iso	任何值	01-19 00:15
long-iso	任何值	2009-01-19 00:15
full-iso	任何值	2009-01-19 00:15:16.000000000 +0900
locale	C	Jan 19 00:15
locale	en_US.UTF-8	Jan 19 00:15
locale	es_ES.UTF-8	ene 19 00:15
+%d.%m.%y %H:%M	任何值	19.01.09 00:15
+%d.%b.%y %H:%M	C or en_US.UTF-8	19.Jan.09 00:15
+%d.%b.%y %H:%M	es_ES.UTF-8	19.ene.09 00:15

Table 9.5: 使用 時間樣式值的”ls -l” 命令的時間和日期的顯示例子

提示
你可以使用命令別名以避免在命令列中輸入長的選項，（參見節 1.5.9）:

```
alias ls='ls --time-style=+%d.%m.%y %H:%M'
```

9.3.5 shell 中 echo 的顏色

大部分現代終端的 shell 中 echo 能夠使用 ANSI 轉義字元來顯示顏色 (參見“/usr/share/doc/xterm/ctlseqs.txt.gz” 嘗試下列例子

```
$ RED=$(printf "\x1b[31m")
$ NORMAL=$(printf "\x1b[0m")
$ REVERSE=$(printf "\x1b[7m")
$ echo "${RED}RED-TEXT${NORMAL} ${REVERSE}REVERSE-TEXT${NORMAL}"
```

9.3.6 有顏色輸出的指令

在互動式的環境下，指令的輸出帶顏色，能夠給檢查指令的輸出帶來便利。我在我的”~/.bashrc” 里加入了下面內容.

```
if [ "$TERM" != "dumb" ]; then
    eval "`dircolors -b`"
    alias ls='ls --color=always'
    alias ll='ls --color=always -l'
    alias la='ls --color=always -A'
    alias less='less -R'
    alias ls='ls --color=always'
    alias grep='grep --color=always'
    alias egrep='egrep --color=always'
    alias fgrep='fgrep --color=always'
    alias zgrep='zgrep --color=always'
else
    alias ll='ls -l'
    alias la='ls -A'
fi
```

在互動式指令中，使用別名來限制顏色的影響範圍。匯出環境變數”export GREP_OPTIONS='--color=auto'” 也有好處，這樣能夠讓 less(1) 之類的頁面程式看到顏色。當使用管道到其它指令時，你想去掉顏色，上面列子”~/.bashrc” 中的內容，可以使用”--color=auto” 代替.

提示
在互動式的環境中，通過”TERM=dumb bash” 呼叫 shell ，你能夠關閉這些顏色別名。

9.3.7 記錄編輯器複雜的重複操作動作

你能夠記錄編輯器複雜的重複操作動作。

對於 Vim, 請按下面操作。

- ”qa”: 開始記錄輸入字元到有名字的暫存器”a”.
- …編輯器操作
- ”q”: 結束記錄輸入的字元。
- ”@a”: 執行暫存器”a 的內容”.

對於 Emacs, 請按下面操作。

- ”C-x (”: 開始定義一個鍵盤巨集.
- …編輯器操作
- ”C-x)”: 結束定義一個鍵盤巨集.
- ”C-x e”: 執行一個鍵盤巨集.

9.3.8 記錄 X 應用程式的圖形

有少量方法可以記錄 X 應用程式的影象，包括 `xterm` 顯示。

軟體包	流行度	大小	螢幕
gnome-screenshot	V:18, I:178	1134	Wayland
flameshot	V:7, I:15	3364	Wayland
gimp	V:51, I:252	19303	Wayland + X
x11-apps	V:31, I:459	2460	X
imagemagick	I:317	74	X
scrot	V:5, I:62	131	X

Table 9.6: 圖形影象處理工具列表

9.3.9 記錄組態檔案的變更

有特定的工具可以透過 DVCS 的幫助來記錄配置檔案的變更和在 Btrfs 上製作系統快照。

軟體包	流行度	大小	說明
etckeeper	V:26, I:30	168	使用 Git（預設）、Mercurial 或 GNU Bazaar 來儲存配置檔案和它們的元資料
timeshift	V:5, I:10	3506	使用 rsync 或 BTRFS 快照的系統恢復工具
snapper	V:4, I:5	2392	Linux 檔案系統快照管理工具

Table 9.7: 記錄配置歷史的軟體包列表

你也可以考慮本地指令碼節 10.2.3 方案。

9.4 監控、控制和啟動程式活動

程式活動能夠使用特殊的工具監控和控制。

> 提示
> `procps` 包提供了非常基礎的監控、控制程式活動功能和啟動程式功能。你應當把他們全部學會。

9.4.1 程序耗時

顯示指令呼叫程序的時間消耗。

```
# time some_command >/dev/null
real    0m0.035s        # time on wall clock (elapsed real time)
user    0m0.000s        # time in user mode
sys     0m0.020s        # time in kernel mode
```

軟體包	流行度	大小	說明
coreutils	V:879, I:999	18307	nice(1): 用指定的排程優先權執行一個程式
bsdutils	V:519, I:999	356	renice(1): 調整一個目前在執行的程序的排程優先權值
procps	V:760, I:999	2391	"/proc" 檔案系統工具: ps(1), top(1), kill(1), watch(1), …
psmisc	V:416, I:776	908	"/proc" 檔案系統工具: killall(1), fuser(1), peekfd(1), pstree(1)
time	V:8, I:137	129	time(1): 執行一個程式，並從時間消耗方面來報告系統資源的使用
sysstat	V:150, I:172	1904	sar(1), iostat(1), mpstat(1), …: linux 系統性能工具
isag	V:0, I:3	109	sysstat 的互動式的系統活動圖
lsof	V:419, I:944	482	lsof(8): 使用"-p" 選項列出被一個系統程序開啟的檔案
strace	V:12, I:121	2897	strace(1): 跟蹤系統呼叫和訊號
ltrace	V:0, I:16	330	ltrace(1): 跟蹤庫呼叫
xtrace	V:0, I:0	353	xtrace(1): 跟蹤 X11 客戶端和伺服器端之間的通訊
powertop	V:18, I:216	677	powertop(1): 系統能耗使用資訊
cron	V:867, I:995	241	根據 cron(8) 後臺背景程式（daemon）的排程執行一個程序
anacron	V:392, I:475	93	用於非整天 24 小時執行系統的指令計劃，類 cron
at	V:103, I:159	158	at(1) 或 batch(1): 在一個特定的時間執行任務或在某一系統負載下執行

Table 9.8: 監控和控制程式活動工具列表

程序優先順序值	排程優先順序
19	最低優先順序程序
0	非常高的普通使用者優先順序程序
-20	root 使用者非常高的優先順序程序

Table 9.9: 排程優先順序值列表

9.4.2 排程優先順序

程序的排程優先順序是被一個程序優先順序值控制。

```
# nice  -19 top                                    # very nice
# nice --20 wodim -v -eject speed=2 dev=0,0 disk.img # very fast
```

在某些情況下，極端的程序優先順序值會對系統造成傷害。小心使用這個指令。

9.4.3 ps 指令

在 Debian 系統上的 ps(1) 指令同時支援 BSD 和 SystemV 特徵，有助於識別靜態的程序活動。

樣式	典型的指令	特徵
BSD	ps aux	顯示%CPU %MEM
System V	ps -efH	顯示 PPID

Table 9.10: ps 指令樣式列表

對於殭屍（死了的）子程序，你能夠通過”PPID” 欄位的父程序 ID 來殺死它們。

pstree(1) 指令顯示程式樹。

9.4.4 top 指令

Debian 系統上的 top(1) 擁有豐富的特徵，有助於識別程序有趣的動態行為。

它是一個互動式的全屏程式。你可以透過按”h” 鍵來得到它的使用幫助，按”q” 鍵來終止該程式。

9.4.5 列出被一個程序開啟的檔案

你能夠通過一個程序 ID(PID) 來列出該程序所有開啟的檔案，例如，PID 為 1 的程序，使用下面的方式。

```
$ sudo lsof -p 1
```

PID=1 通常用於 init 程式.

9.4.6 跟蹤程式活動

你能夠跟蹤程式活動，使用 strace(1), ltrace(1), xtrace(1) 來跟蹤系統呼叫和訊號、庫呼叫、X11 客戶端和伺服器端之間的通訊。

跟蹤 ls 指令的系統呼叫。

```
$ sudo strace ls
```

提示
使用在 **/usr/share/doc/strace/examples/** 中發現的 **strace-graph** 指令碼來生成一個好看的樹形檢視

9.4.7 識別使用檔案和套接字的程序

你可以通過 fuser(1) 來識別出使用檔案的程序，例如，用下面的方式識別出"/var/log/mail.log" 由哪個程序開啟。

```
$ sudo fuser -v /var/log/mail.log
                     USER        PID ACCESS COMMAND
/var/log/mail.log:   root       2946 F.... rsyslogd
```

你可以看到"/var/log/mail.log" 是由 rsyslogd(8) 指令開啟並寫入。

你可以通過 fuser(1) 來識別出使用套接字的程序，例如，用下面的方式識別出"smtp/tcp" 由哪個程序開啟。

```
$ sudo fuser -v smtp/tcp
                     USER        PID ACCESS COMMAND
smtp/tcp:            Debian-exim   3379 F.... exim4
```

現在你知道你的系統執行 exim4(8) 來處理連線到 SMTP 埠 (25) 的 TCP 連線.

9.4.8 使用固定間隔重複一個指令

watch(1) 使用固定間隔重新執行一個指令，並全螢幕顯示輸出。

```
$ watch w
```

顯示哪些人登入到系統，每 2 秒鐘更新一次。

9.4.9 使用檔案迴圈來重複一個指令

通過匹配某些條件的檔案來迴圈重複一個指令，有幾種方法，例如，匹配全域性模式"*.ext".

- Shell 迴圈方式 (參見節 12.1.4):

```
for x in *.ext; do if [ -f "$x"]; then command "$x" ; fi; done
```

- find(1) 和 xargs(1) 聯合:

```
find . -type f -maxdepth 1 -name '*.ext' -print0 | xargs -0 -n 1 command
```

- find(1) 使用"-exec" 選項並執行指令:

```
find . -type f -maxdepth 1 -name '*.ext' -exec command '{}' \;
```

- find(1) 使用"-exec" 選項並執行一個短的 shell 指令碼:

```
find . -type f -maxdepth 1 -name '*.ext' -exec sh -c "command '{}' && echo 'successful'" \;
```

上面的列子確保適當處理怪異的檔名 (如包含空格)。find(1) 更多高階的用法，參見節 10.1.5.

9.4.10 從 GUI 啟動一個程式

對於 指令列介面（command-line interface，CLI），$PATH 環境變數所指定的目錄中第一個匹配相應名稱的程式會被執行。參見節 1.5.3。

對於遵從 freedesktop.org 標準的 圖形使用者介面（graphical user interface，GUI），/usr/share/applications/ 目錄中的 *.desktop 檔案給每個程式的 GUI 選單顯示提供了必要的屬性。遵從 Freedesktop.org xdg 菜單系統的每一個軟體包，透過”/usr/share/applications/” 下”*.desktop” 提供的資料來安裝它的選單。遵從 Freedesktop.org 標準的現代桌面環境，用 xdg-utils 軟體包利用這些資料生成它們的選單。參見”/usr/share/doc/xdg-utils/README”。

舉個例子，chromium.desktop 檔案中為“Chromium 網路瀏覽器”定義了相關屬性，例如程式名“Name”，程式執行路徑和參數“Exec”，所使用的圖示“Icon”等等（參見 桌面調配項規範）。檔案內容如下：

```
[Desktop Entry]
Version=1.0
Name=Chromium Web Browser
GenericName=Web Browser
Comment=Access the Internet
Comment[fr]=Explorer le Web
Exec=/usr/bin/chromium %U
Terminal=false
X-MultipleArgs=false
Type=Application
Icon=chromium
Categories=Network;WebBrowser;
MimeType=text/html;text/xml;application/xhtml_xml;x-scheme-handler/http;x-scheme-handler/ ↩
    https;
StartupWMClass=Chromium
StartupNotify=true
```

這是一個較為簡單的說明。*.desktop 檔案像下面那樣被搜尋。

桌面環境設定 $XDG_DATA_HOME 和 $XDG_DATA_DIR 環境變數。舉個例子，在 GNOME 3 中：

- 未設定 $XDG_DATA_HOME。（將使用預設值 $HOME/.local/share。）
- $XDG_DATA_DIRS 被設定為 /usr/share/gnome:/usr/local/share/:/usr/share/。

基準目錄（參見 XDG Base Directory Specification）和應用程式目錄如下所示。

- $HOME/.local/share/ → $HOME/.local/share/applications/
- /usr/share/gnome/ → /usr/share/gnome/applications/
- /usr/local/share/ → /usr/local/share/applications/
- /usr/share/ → /usr/share/applications/

*.desktop 檔案將按照這個順序在這些 applications 目錄中進行搜尋。

提示
要建立一個使用者自定義的 GUI 選單項，需要在 $HOME/.local/share/applications/ 目錄中新增一個 *.desktop 檔案。

提示
”Exec=...” 行不會由 shell 解析。如果需要設定環境變數，使用 env(1) command。

提示
相似地，如果在這些基準目錄下的 autostart 目錄中建立了一個 *.desktop 檔案，則 *.desktop 檔案中指定的程式會在桌面環境啟動時自動執行。參見 Desktop Application Autostart Specification。

提示
相似地，如果在 $HOME/Desktop 目錄中建立了一個 *.desktop 檔案並且桌面環境被調配為支援桌面圖示啟動器功能，則點選圖示時指定的程式會被執行。請注意，$HOME/Desktop 目錄的實際名稱與語言環境有關。參見 xdg-user-dirs-update(1)。

9.4.11 自定義被啟動的程式

一些程式會被另一個程式自動啟動。下面是自定義該過程的方法。

- 應用程式調配選單：
  - GNOME3 桌面：“設定” → “系統” → “詳細資訊” → “預設應用程式”
  - KDE 桌面: ”K” → ”Control Center 控制中心” → ”KDE Components 元件” → ”Component Chooser 元件選擇器”
  - Iceweasel 瀏覽器：“編輯” → “首選項” → “應用程式”
  - mc(1)：“/etc/mc/mc.ext”
- 例如“$BROWSER”、“$EDITOR”、“$VISUAL”和“$PAGER”這樣的環境變數（參見 environ(7)）
- 用於例如“editor”、“view”、“x-www-browser”、“gnome-www-browser”和“www-browser”這樣的程式的 update-alternatives(1) 系統（參見節 1.4.7）
- “~/.mailcap”和“/etc/mailcap”檔案的內容關聯了程式的 MIME 型別（參見 mailcap(5)）
- “~/.mime.types”和“/etc/mime.types”檔案的內容關聯了 MIME 型別的副檔名（參見 run-mailcap(1)）

提示
update-mime(8) 會更新”/etc/mailcap” 檔案，期間會用到”/etc/mailcap.order” 檔案 (參見 mailcap.order(5)).

提示
debianutils 軟體包提供 sensible-browser(1)、sensible-editor(1) 和 sensible-pager(1)，它們可以分別對要呼叫的編輯器、分頁程式和網路瀏覽器作出明智的選擇。我建議你閱讀那些 shell 指令碼。

提示
為了在 GUI（圖形使用者介面）下執行例如 mutt 這樣的控制檯應用程式來作為你的首選應用程式，你應該像下面那樣建立一個 GUI（圖形使用者介面）應用程式並設定“/usr/local/bin/mutt-term”為你想要啟動的首選應用程式。

```
# cat /usr/local/bin/mutt-term <<EOF
#!/bin/sh
gnome-terminal -e "mutt \$@"
EOF
# chmod 755 /usr/local/bin/mutt-term
```

訊號值	訊號名	操作	註釋
0	---	沒有訊號傳送 (參見 kill(2))	檢查程序是否執行
1	SIGHUP	終止程序	從終端斷開連線（訊號掛起）
2	SIGINT	終止程序	從鍵盤中斷 (CTRL-C)
3	SIGQUIT	終止程序並觸發 dump core	從鍵盤退出 (CTRL-\)
9	SIGKILL	終止程序	不可阻塞的 kill 訊號
15	SIGTERM	終止程序	可被阻塞的終止訊號

Table 9.11: kill 指令常用訊號列表

9.4.12 殺死一個程序

使用 kill(1) 通過程序 ID 來殺死（傳送一個訊號）一個程序。

使用 killall(1) 或 pkill(1) 通過程序命令的名字或其它屬性來做同樣的事情。

9.4.13 單次任務時間安排

執行 at(1) 指令來安排一次性的工作。

```
$ echo 'command -args'| at 3:40 monday
```

9.4.14 定時任務安排

使用 cron(8) 來進行定時任務安排。參見 crontab(1) 和 crontab(5).

你能夠作為一個普通使用者定時執行一個程序，比如，foo 使用"crontab -e" 指令建立一個 crontab(5) 的檔案"/var/spool/cron/crontabs/foo"。

這裡是一個 crontab(5) 檔案的列子。

```
# use /usr/bin/sh to run commands, no matter what /etc/passwd says
SHELL=/bin/sh
# mail any output to paul, no matter whose crontab this is
MAILTO=paul
# Min Hour DayOfMonth Month DayOfWeek command (Day... are OR'ed)
# run at 00:05, every day
5  0  *  * *   $HOME/bin/daily.job >> $HOME/tmp/out 2>&1
# run at 14:15 on the first of every month -- output mailed to paul
15 14 1  * *   $HOME/bin/monthly
# run at 22:00 on weekdays(1-5), annoy Joe. % for newline, last % for cc:
0 22 *   * 1-5 mail -s "It's 10pm" joe%Joe,%%Where are your kids?%.%%
23 */2 1 2 *   echo "run 23 minutes after 0am, 2am, 4am ..., on Feb 1"
```

```
5  4 *   * sun echo "run at 04:05 every Sunday"
# run at 03:40 on the first Monday of each month
40 3 1-7 * *   [ "$(date +%a)" == "Mon" ] && command -args
```

提示
對那些非連續執行的系統，安裝 anacron 軟體包來定時執行週期性的指令，指令在接近機器啟動的時間執行，並允許有特定的時間間隔。參見 anacron(8) 和 anacrontab(5).

提示
對於定時系統維護指令碼，你能夠以 root 帳號定時執行，把這類指令碼放入"/etc/cron.hourly/", "/etc/cron.daily/", "/etc/cron.weekly/", 或"/etc/cron.monthly/". 這些指令碼的執行時間，可以通過"/etc/crontab" 和"/etc/anacrontab" 來定製。

cron 後臺守護程序 (daemon) 不存在時，Systemd 也有按時間計劃執行程式的低階能力。例如，/lib/systemd/system/apt 和 /lib/systemd/system/apt-daily.service 建立每天的 apt 下載行動。參見 systemd.timer(5) 。

9.4.15 基於事件的計劃任務

Systemd 能夠執行計劃程式，不僅基於時間事件，還能夠基於掛載事件。參見節 10.2.3.3 和節 10.2.3.2 的例子。

9.4.16 Alt-SysRq 鍵

按 Alt-SysRq (PrtScr) 組合鍵跟一個字母按鍵，進行不可思議的系統應急控制。

Alt-SysRq 之後的鍵	行為描述
k	**k**ill 殺死在當前虛擬控制檯上的所有程序 (SAK)
s	**s**ync 所有已經掛載的檔案系統來避免資料損壞
u	重新以只讀方式掛載所有已掛載的檔案系統 (**u**mount)
r	在 X 崩潰後，從 **r**aw 模式恢復鍵盤

Table 9.12: 著名的 SAK 命令鍵列表

更多資訊參見 Linux 核心使用者和管理員手冊 » Linux Magic System Request Key Hacks

提示
從 SSH 終端等，你能夠通過向"/proc/sysrq-trigger" 寫入內容來使用 Alt-SysRq 特性。例如，從 root shell 提示字元執行"echo s > /proc/sysrq-trigger; echo u > /proc/sysrq-trigger" 來 **s**yncs 和 **u**mounts 所有已掛載的檔案系統。

目前 (2021) Debian amd64 Linux 核心為 /proc/sys/kernel/sysrq=438=0b110110110:

- 2 = 0x2 - 啟用控制檯日誌級別控制 (開啟)
- 4 = 0x4 - 啟用鍵盤控制 (SAK, unraw) (開啟)
- 8 = 0x8 - 啟用程序除錯轉儲 (debugging dumps of processes) 等。(關閉)
- 16 = 0x10 - 啟用 sync 命令 (開啟)

- 32 = 0x20 - 啟用只讀重新掛載（開啟）
- 64 = 0x40 - 啟用程序訊號 (term, kill, oom-kill) (關閉)
- 128 = 0x80 - 允許重啟、關閉電源（開啟）
- 256 = 0x100 - 允許調整所有 RT（實時）任務優先順序（開啟）

9.5 系統維護技巧

9.5.1 誰在系統裡？

你可以通過下面的方法檢查誰登入在系統裡。

- `who`(1) 顯示誰登入在系統裡面。
- `w`(1) 顯示誰登入在系統裡面，他們在做什麼。
- `last`(1) 顯示使用者最後登入的列表。
- `lastb`(1) 顯示使用者最後錯誤登入的列表。

提示
”`/var/run/utmp`” 和”`/var/log/wtmp`” 儲存這樣的使用者資訊。參見 `login`(1) 和 `utmp`(5).

9.5.2 警告所有人

你可以通過下面的方式使用 `wall`(1) 給登入系統的每一個人傳送資訊。

```
$ echo ”We are shutting down in 1 hour” | wall
```

9.5.3 硬體識別

對於 PCI 類裝置 (AGP, PCI-Express, CardBus, ExpressCard 等), 一開始就使用 `lspci`(8) (也許加上”`-nn`” 選項) 進行硬體識別比較好。

此外，你可以通過閱讀”`/proc/bus/pci/devices`” 裡面的內容或瀏覽”`/sys/bus/pci`” 下面的目錄樹來進行硬體識別 (參見節 1.2.12).

軟體包	流行度	大小	說明
`pciutils`	V:246, I:991	212	Linux PCI 工具: `lspci`(8)
`usbutils`	V:73, I:868	325	Linux USB 工具: `lsusb`(8)
`nvme-cli`	V:14, I:22	1621	Linux NVMe 工具: `nvme`(1)
`pcmciautils`	V:6, I:10	91	Linux PCMCIA 工具: `pccardctl`(8)
`scsitools`	V:0, I:2	346	SCSI 硬體管理工具集: `lsscsi`(8)
`procinfo`	V:0, I:9	132	從”`/proc`”: `lsdev`(8) 獲得系統資訊
`lshw`	V:13, I:90	919	硬體調配資訊: `lshw`(1)
`discover`	V:40, I:957	98	硬體識別系統: `discover`(8)

Table 9.13: 硬體識別工具列表

9.5.4 硬體調配

像 GNOME 和 KDE 這類現代圖形桌面系統，雖然大部分硬體的調配都能夠通過相應的圖形調配工具來管理，但知道一些調配它們的基礎方式，也是一個好的主意。

軟體包	流行度	大小	說明
console-setup	V:90, I:967	428	Linux 控制檯字型和鍵盤表工具
x11-xserver-utils	V:298, I:524	568	X 服務端工具: xset(1), xmodmap(1)
acpid	V:85, I:153	158	管理高階可調配和電源介面 (ACPI) 事件分發的後臺背景程式 (daemon)
acpi	V:10, I:141	47	顯示 ACPI 裝置資訊的工具
sleepd	V:0, I:0	86	在筆記本空閒時，使其進入休眠狀態的後臺背景程式 (daemon)
hdparm	V:181, I:348	256	硬碟存取最佳化 (參見節 9.6.9)
smartmontools	V:205, I:249	2358	使用 S.M.A.R.T. 控制和監控儲存系統
setserial	V:4, I:7	103	串列埠管理工具集
memtest86+	V:1, I:21	12711	記憶體硬體管理工具集
scsitools	V:0, I:2	346	SCSI 硬體管理工具集
setcd	V:0, I:0	37	光碟機存取最佳化
big-cursor	I:0	26	圖形化界面的大滑鼠游標

Table 9.14: 硬體調配工具列表

這裡, ACPI 是一個比 APM 新的電源管理系統框架。

> 提示
> 現代系統的 CPU 頻率調整功能，是由核心模組 acpi_cpufreq 管理的。

9.5.5 系統時間和硬體時間

下面設定系統的硬體時間為：MM/DD hh:mm, CCYY.

```
# date MMDDhhmmCCYY
# hwclock --utc --systohc
# hwclock --show
```

Debian 系統的時間通常顯示為本地時間，但硬體時間通常使用 UTC(GMT) 時間。

如果硬體時間設定為 UTC 時間，請在“/etc/default/rcS”裡面設定“UTC=yes”。

下面是重新調配 Debian 系統使用的時區。

```
# dpkg-reconfigure tzdata
```

如果你希望通過網路來更新系統時間，考慮使用 ntp, ntpdate 和 chrony 這類包提供的 NTP 服務。

> 提示
> 在 systemd 下，是使用 systemd-timesyncd 來替代進行網路時間同步。參見 systemd-timesyncd(8).

看下面。

- 精確時間和日期管理 HOWTO
- NTP 公共服務專案

- ntp-doc 包

提示
在 ntp 包裡面的 ntptrace(8) 能夠跟蹤 NTP 服務鏈至原始源。

9.5.6 終端調配

有幾個元件可以用來調配字元控制檯和 ncurses(3) 系統功能。

- “/etc/terminfo/*/*”檔案 (terminfo(5))
- “$TERM”環境變數 (term(7))
- setterm(1)、stty(1)、tic(1) 和 toe(1)

如果 xterm 的 terminfo 對非 Debian 的 xterm 不起作用，則當你從遠端登入到 Debian 系統時，你需要改變你的終端型別“$TERM”,從“xterm”更改為功能受限的版本 (例如“xterm-r6”)。更多內容參見“/usr/share/doc/libncurses5/F
是“$TERM”中最通用的。

9.5.7 聲音基礎設施

用於現在的 Linux 的音效卡裝置驅動程式由 高階 Linux 聲音體系 (Advanced Linux Sound Architecture，ALSA) 提供。ALSA 提供了相容之前的 開放聲音系統 (Open Sound System，OSS) 的模擬模式。

應用軟體不僅可被配置為不僅直接訪問聲音裝置，也可以透過一些標準化聲音服務端系統來訪問它們。目前，PulseAudio、JACK 和 PipeWire 被用作聲音的服務端系統。參見 Debian 維基聲音頁面 得到最新情況。

每個流行的桌面環境通常都有一個通用的聲音引擎。每個被應用程式使用的聲音引擎都可以選擇連線到不同的聲音伺服器。

提示
使用“cat /dev/urandom > /dev/audio”或 speaker-test(1) 來測試揚聲器 (^C 停止)。

提示
如果你無法聽到聲音，那你的揚聲器可能連線到了一個靜音輸出。現代的聲音系統有許多輸出。alsa-utils 軟體包中的 alsamixer(1) 可以很好地調配聲音和靜音設定。

9.5.8 關閉螢幕保護

關閉螢幕保護，使用下面的指令。

9.5.9 關閉蜂鳴聲

可以把電腦的揚聲器拔掉來關閉蜂鳴聲。把 pcspkr 核心模組刪除，也可以做到這點。

bash(1) 用到的 readline(3) 程式，當遇到告警字元 (ASCII=7) 時，將會發生。下面的操作將阻止發生。

```
$ echo "set bell-style none">> ~/.inputrc
```

軟體包	流行度	大小	說明
alsa-utils	V:323, I:463	2605	調配和使用 ALSA 的工具
oss-compat	V:1, I:17	18	在 ALSA 下相容 OSS，預防“/dev/dsp not found”錯誤
pipewire	V:264, I:316	120	音訊和影片處理引擎多媒體服務端 - 元資料包
pipewire-bin	V:274, I:316	1630	音訊和影片處理引擎多媒體服務端 - 音訊服務和命令列程式
pipewire-alsa	V:99, I:149	205	音訊和影片處理引擎多媒體服務端 - 代替 ALSA 的音訊服務
pipewire-pulse	V:156, I:204	49	音訊和影片處理引擎多媒體服務端 - 代替 PulseAudio 的音訊服務
pulseaudio	V:259, I:314	6472	PulseAudio 服務端
libpulse0	V:408, I:578	975	PulseAudio 客戶端庫
jackd	V:2, I:19	9	JACK Audio Connection Kit. (JACK) 伺服器 (低延遲)
libjack0	V:1, I:10	329	JACK Audio Connection Kit. (JACK) 庫 (低延遲)
libgstreamer1.0-0	V:428, I:593	4454	GStreamer：GNOME 聲音引擎
libphonon4qt5-4	V:72, I:160	593	Phonon：KDE 聲音引擎

Table 9.15: 聲音軟體包

環境	指令
Linux 控制檯	setterm -powersave off
X 視窗 (關閉螢幕保護)	xset s off
X 視窗 (關閉 dpms)	xset -dpms
X 視窗 (螢幕保護 GUI 調配)	xscreensaver-command -prefs

Table 9.16: 關閉螢幕保護指令列表

9.5.10 記憶體使用

對你來說，這裡有兩種可用的方法來得到記憶體的使用情況。

- “/var/log/dmesg”中的核心啟動資訊包含了可用記憶體的精確總大小。
- free(1) 和 top(1) 顯示正在執行的系統中記憶體資源的相關資訊。

下面是一個例子。

```
# grep '\] Memory' /var/log/dmesg
[    0.004000] Memory: 990528k/1016784k available (1975k kernel code, 25868k reserved, 931k ↩
    data, 296k init)
$ free -k
             total       used       free     shared    buffers     cached
Mem:        997184     976928      20256          0     129592     171932
-/+ buffers/cache:     675404     321780
Swap:      4545576          4    4545572
```

你可能會覺得奇怪：“dmesg 告訴你 free 為 990 MB，而 free -k 告訴你 free 為 320 MB。這丟失了超過 600 MB ……”。

別擔心“Mem:”這行中“used”較大的值以及“free”較小的值，放輕鬆，你需要檢視的是下面的那個 (在上面的例子中它們是 675404 和 321780)。

對於我的 MacBook，有 1GB=1048576k 記憶體 (顯示卡系統用掉一些)，我看到的如下。

9.5.11 系統安全性和完整性檢查

糟糕的系統維護可能會暴露你的系統，導致它被外部非法使用。

對於系統安全性和完整性的檢查，你需要從下面這些方面開始。

報告	大小
dmesg 中 total 的大小	1016784k = 1GB - 31792k
dmesg 中的 free	990528k
shell 下的 total	997184k
shell 下的 free	20256k（但有效的為 321780k）

Table 9.17: 報告的記憶體大小

- `debsums` 軟體包，參見 `debsums`(1) 和節 2.5.2。
- `chkrootkit` 軟體包，參見 `chkrootkit`(1)。
- `clamav` 軟體包家族，參見 `clamscan`(1) 和 `freshclam`(1)。
- Debian security FAQ。
- Securing Debian Manual。

軟體包	流行度	大小	說明
`logcheck`	V:6, I:8	110	後臺背景程式（daemon），將系統日誌檔案中的異常通過郵件傳送給管理員
`debsums`	V:4, I:36	98	實用程式，使用 MD5 校驗碼對已安裝軟體包的檔案進行校驗
`chkrootkit`	V:7, I:17	925	rootkit 檢測軟體
`clamav`	V:9, I:45	27455	Unix 的反病毒實用程式——指令列介面
`tiger`	V:1, I:2	7800	報告系統安全漏洞
`tripwire`	V:2, I:2	12168	檔案和目錄完整性檢測軟體
`john`	V:1, I:9	471	先進的密碼破解工具
`aide`	V:1, I:1	293	高階入侵環境檢測——靜態二進位制
`integrit`	V:0, I:0	2659	檔案完整性驗證程式
`crack`	V:0, I:1	149	密碼猜測程式

Table 9.18: 用於系統安全性和完整性檢查的工具

下面是一個簡單的指令碼，用來檢測典型的所有人可寫的錯誤檔案許可權。

```
# find / -perm 777 -a \! -type s -a \! -type l -a \! \( -type d -a -perm 1777 \)
```

注意
由於 `debsums` 軟體包使用本地儲存的 MD5 校驗碼，因此面對惡意攻擊，也不能完全相信系統安全性檢測工具。

9.6 資料儲存技巧

使用 live CD 或 debian-installer CD 以救援模式啟動你的系統，可以讓你簡單地重新調配你的啟動裝置的資料儲存。

如果裝置在 GUI（圖形使用者介面）桌面系統下被自動掛載，在對它們進行操作前，你可能需要從命令列手工 `umount`(8) 這些裝置。

9.6.1 硬碟空間使用情況

硬碟空間的使用情況可以通過 mount、coreutils 和 xdu 軟體包提供的程式來評估：

- mount(8) 顯示所有掛載的檔案系統 (= 磁碟).
- df(1) 報告檔案系統使用的硬碟空間。
- du(1) 報告目錄樹使用的硬碟空間。

提示
你可以將 du(8) 的輸出傳輸給 xdu(1x)，來使用它的圖形互動式演示，例如“du -k . |xdu”、“sudo du -k -x / |xdu”等等。

9.6.2 硬碟分割槽調配

對於硬碟分割槽調配，儘管 fdisk(8) 被認為是標準的調配，但是 parted(8) 工具還是值得注意的。

老的 PC 使用經典的主引導記錄（Master Boot Record，MBR）方案，將硬碟分割槽資料儲存在第一個扇區，即 LBA 扇區 0（512 位元組）。

一些帶有統一可擴充套件韌體介面（Unified Extensible Firmware Interface，UEFI）的近代 PC，包括基於 Intel 的 Mac，使用 全域性唯一標識分割槽表（GUID Partition Table，GPT）方案，硬碟分割槽資料不儲存在第一個扇區。

儘管 fdisk(8) 一直是硬碟分割槽的標準工具，但現在 parted(8) 替代了它。

軟體包	流行度	大小	說明
util-linux	V:880, I:999	5283	多種系統工具，包含 fdisk(8) 和 cfdisk(8)
parted	V:411, I:565	122	GNU Parted，硬碟分割槽調整程式
gparted	V:15, I:103	2175	基於 libparted 的 GNOME 分割槽編輯程式
gdisk	V:330, I:506	885	用於 GPT/MBR 並存的硬碟的分割槽編輯程式
kpartx	V:21, I:34	77	為分割槽建立裝置對映的程式

Table 9.19: 硬碟分割槽管理軟體包

注意
儘管 parted(8) 聲稱也能用來建立和調整檔案系統，但使用維護最好的專門工具來做這些事會更為安全，例如 mkfs(8)（mkfs.msdos(8)、mkfs.ext2(8)、mkfs.ext3(8)、mkfs.ext4(8)……）和 resize2fs(8)。

注
為了在 GPT 和 MBR 之間切換，你需要直接刪除開頭的幾個塊中的內容（參見節 9.8.6）並使用“parted /dev/sdx mklabel gpt”或“parted /dev/sdx mklabel msdos”來設定它。請注意，這裡使用的“msdos”是用於 MBR。

9.6.3 使用 UUID 存取分割槽

儘管重新配置你的分割槽或可移動儲存介質的啟用順序可能會給分割槽產生不同的名字，但你可以使用同一個 UUID 來訪問它們。如果你有多個硬碟並且你的 BIOS/UEFI 沒有給它們一致的裝置名的話，使用 UUID 是不錯的選擇。

- mount(8) 指令帶有“-U”選項可以使用UUID 來掛載一個塊裝置，而不必使用他的檔名稱，例如“/dev/sda3”。

- “/etc/fstab”（參見 fstab(5)）可以使用 UUID。
- 引載載入程式（節 3.1.2）也可以使用 UUID。

提示
你可以使用 blkid(8) 來檢視一個特定塊裝置的 UUID。
你也可以使用”lsblk -f” 來檢測 UUID 並檢視其它資訊。

9.6.4 LVM2

LVM2 是一個用於 Linux 核心的邏輯卷管理器。使用 LVM2 的話，硬碟分割槽可以建立在邏輯捲上來替代物理硬碟。
LVM 有下列需求。

- Linux 核心中的裝置對映支援（Debian 核心預設支援）
- 使用者自定義裝置對映支援庫（libdevmapper* 軟體包）
- 使用者自定義 LVM2 工具（lvm2 軟體包）

請從下面的 man 手冊開始瞭解 LVM2。

- lvm(8)：LVM2 機制的基礎知識（列出了所有 LVM2 指令）
- lvm.conf(5)：LVM2 的組態檔案
- lvs(8)：報告邏輯卷的相關資訊
- vgs(8)：報告卷組的相關資訊
- pvs(8)：報告物理卷的相關資訊

9.6.5 檔案系統調配

對於 ext4 檔案系統, e2fsprogs 包提供下面的工具。

- mkfs.ext4(8) 建立新的 ext4 檔案系統
- fsck.ext4(8) 檢查和修復現有 ext4 檔案系統
- tune2fs(8) 調配 ext4 檔案系統的超級塊
- debugfs(8) 互動式的除錯 ext4 檔案系統. (它有 undel 指令來恢復已經刪除的檔案.)

mkfs(8) 和 fsck(8) 指令是由 e2fsprogs 包提供的,是各種檔案系統相關程式的前端。(mkfs.fstype 和 fsck.fstype).
對於 ext4 檔案系統, 它們是 mkfs.ext4(8) 和 fsck.ext4(8) (它們被符號連結到 mke2fs(8) 和 e2fsck(8)).

Linux 支援的每一個檔案系統，有相似的指令。

提示
Ext4 檔案系統是 Linux 系統上預設的檔案系統，強烈推薦使用這個檔案系統，除非你有特殊的理由不使用。
Btrfs 狀態能夠在 Debian wiki on btrfs 和 kernel.org wiki on btrfs 發現。它被期望作為 ext4 檔案系統之後的下一個預設檔案系統。
一些工具可以在沒有 Linux 核心支援的情況下存取檔案系統 (參見節 9.8.2).

軟體包	流行度	大小	說明
e2fsprogs	V:767, I:999	1501	ext2/ext3/ext4 檔案系統工具
btrfs-progs	V:45, I:72	5078	Btrfs 檔案系統工具
reiserfsprogs	V:12, I:25	473	Reiserfs 檔案系統工具
zfsutils-linux	V:29, I:30	1755	OpenZFS 檔案系統工具
dosfstools	V:191, I:537	315	FAT 檔案系統工具. (Microsoft: MS-DOS, Windows)
exfatprogs	V:28, I:357	301	exFAT 檔案系統工具，由三星維護。
exfat-fuse	V:6, I:128	73	FUSE 讀寫 exFAT 檔案系統（微軟）驅動。
exfat-utils	V:4, I:115	231	exFAT 檔案系統工具，由 exfat-fuse 的作者維護。
xfsprogs	V:22, I:96	3493	XFS 檔案系統工具. (SGI: IRIX)
ntfs-3g	V:200, I:509	1470	FUSE 讀寫 NTFS 檔案系統（微軟：Windows NT⋯⋯）驅動。
jfsutils	V:0, I:8	1577	JFS 檔案系統工具. (IBM: AIX, OS/2)
reiser4progs	V:0, I:2	1367	Reiser4 檔案系統工具
hfsprogs	V:0, I:5	394	HFS 和 HFS Plus 檔案系統工具. (Apple: Mac OS)
zerofree	V:5, I:131	25	把 ext2/3/4 檔案系統上空閒塊設定為零的程式

Table 9.20: 檔案系統管理包列表

9.6.6 檔案系統建立和完整性檢查

mkfs(8) 在 Linux 系統上建立檔案系統。fsck(8) 指令在 Linux 系統上提供檔案系統完整性檢查和修復功能。

在檔案系統建立後，Debian 現在預設不周期性的執行 fsck。

注意
在已經掛載的檔案系統上執行 fsck ，一般是不安全的.

提示
在"/etc/mke2fs.conf" 裡設定"enable_periodic_fsck" 並使用"tune2fs -c0 /dev/*partition_name*" 設定最大掛載數為 0，便可以在重啟時，讓 root 檔案系統包括在內的所有檔案系統上，安全的執行 fsck(8) 指令. 參見 mke2fs.conf(5) 和 tune2fs(8).
從啟動腳本里面執行的 fsck(8) 指令結果，可以在"/var/log/fsck/" 目錄下檢視。

9.6.7 通過掛載選項優化檔案系統

“/etc/fstab”中包含了基礎的靜態檔案系統調配。例如，

```
«file system»                    «mount point» «type» «options»    «dump» «pass»
proc                                     /proc proc   defaults          0 0
UUID=709cbe4c-80c1-56db-8ab1-dbce3146d2f7 /     ext4   errors=remount-ro 0 1
UUID=817bae6b-45d2-5aca-4d2a-1267ab46ac23 none  swap   sw                0 0
/dev/scd0                         /media/cdrom0  udf,iso9660 user,noauto  0 0
```

提示
UUID（參見節 9.6.3）可以替代一般的塊裝置名稱（例如“/dev/sda1”、“/dev/sda2”⋯⋯）來識別一個塊裝置。

從 Linux 2.6.30 起，核心的預設行為是提供"relatime" 選項。

參見 fstab(5) 和 mount(8)。

9.6.8 通過超級塊（superblock）優化檔案系統

一個檔案系統的特性可以使用 tune2fs(8) 指令通過超級塊來優化。

- 執行“sudo tune2fs -l /dev/hda1”可以顯示“/dev/hda1”上的檔案系統超級塊內容。
- 執行”sudo tune2fs -c 50 /dev/hda1” 改變”/dev/hda1” 檔案系統的檢查 (在啟動時執行 fsck) 頻率為每 50 次啟動.
- 執行“sudo tune2fs -j /dev/hda1”會給檔案系統新增日誌功能，即“/dev/hda1”的檔案系統從 ext2 轉換為 ext3。（對未掛載的檔案系統這麼做。）
- 執行“sudo tune2fs -O extents,uninit_bg,dir_index /dev/hda1 && fsck -pf /dev/hda1”在“/dev/hda1”上將它從 ext3 轉換為 ext4。（對未掛載的系統這麼做。）

提示
儘管 tune2fs(8) 的名字是這樣的，但它不僅能用於 ext2 檔案系統，也能用於 ext3 和 ext4 檔案系統。

9.6.9 硬碟最佳化

警告
在你折騰硬碟調配之前，請檢查你的硬體並閱讀 hdparm(8) 的 man 手冊頁，因為這可能會對資料完整性造成相當大的危害。

你可以通過“hdparm -tT /dev/hda”來測試“/dev/hda”硬碟的存取速度。對於一些使用 (E)IDE 連線的硬碟，你可以使用“hdparm -q -c3 -d1 -u1 -m16 /dev/hda”來啟用“(E)IDE 32 位支援”、啟用“using_dma flag”、設定“interrupt-unmask flag”並設定“multiple 16 sector I/O”（危險！），從而加速硬碟存取速度。

你可以通過“hdparm -W /dev/sda”來測試“/dev/sda”硬碟的寫入快取功能。你可以使用“hdparm -W 0 /dev/sda”關閉寫入快取功能。

現代高速 CD-ROM 光碟機，你可以使用”setcd -x 2” 降低速度，來讀取不當壓縮的 CDROM 光碟。

9.6.10 固態硬碟最佳化

固態硬碟（Solid state drive，SSD） 目前可以被自動檢測。

在 /etc/fstab 裡面，將易失性資料路徑掛載為”tmpfs”，可以減少不必要的磁碟訪問來阻止磁碟損耗。

9.6.11 使用 SMART 預測硬碟故障

你可以使用相容 SMART 的 smartd(8) 後臺背景程式（daemon）來監控和記錄你的硬碟。

1. 在 BIOS 中啟用 SMART 功能。
2. 安裝 smartmontools 軟體包。
3. 通過 df(1) 列出硬碟驅動並識別它們。
   - 讓我們假設要監控的硬碟為“/dev/hda”。
4. 檢查“smartctl -a /dev/hda”的輸出，看 SMART 功能是否已啟用。
   - 如果沒有，通過“smartctl -s on -a /dev/hda”啟用它。

5. 通過下列方式執行 smartd(8) 後臺背景程式 (daemon)。
   - 消除 /etc/default/smartmontools” 檔案中“start_smartd=yes” 的註釋。
   - 透過“sudo systemctl restart smartmontools” 重新啟動 smartd(8) 後臺守護程序 (daemon)。

提示
smartd(8) 後臺背景程式 (daemon) 可以使用 /etc/smartd.conf 檔案進行自定義，檔案中包含了相關的警告。

9.6.12 通過 $TMPDIR 指定臨時儲存目錄

應用程式一般在臨時儲存目錄“/tmp” 下建立臨時檔案。如果“/tmp” 沒有足夠的空間，你可以通過 $TMPDIR 變數來為程式指定臨時儲存目錄。

9.6.13 通過 LVM 擴充可用儲存空間

在安裝時建立在 Logical Volume Manager 邏輯儲區管理 (LVM) (Linux 特性) 上的分割槽，它們可以容易的通過合併擴充或刪除擴充的方式改變大小，而不需要在多個儲存裝置上進行大量的重新調配。

9.6.14 通過掛載另一個分割槽來擴充可用儲存空間

如果你有一個空的分割槽 (例如“/dev/sdx”)，你可以使用 mkfs.ext4(1) 將它格式化，並使用 mount(8) 將它掛載到你需要更多空間的目錄。(你需要複製原始資料內容。)

```
$ sudo mv work-dir old-dir
$ sudo mkfs.ext4 /dev/sdx
$ sudo mount -t ext4 /dev/sdx work-dir
$ sudo cp -a old-dir/* work-dir
$ sudo rm -rf old-dir
```

提示
你也可以選擇掛載一個空硬碟映像檔案 (參見節 9.7.5) 作為一個迴圈裝置 (參見節 9.7.3)。實際的硬碟使用量會隨著實際儲存資料的增加而增加。

9.6.15 通過“mount --bind” 掛載另一個目錄來擴充套件可用儲存空間

如果你在另一個分割槽裡有一個帶有可用空間的空目錄 (例如“/path/to/emp-dir”)，你可以通過帶有“--bind” 選項的 mount(8)，將它掛載到一個你需要更多空間的目錄 (例如“work-dir”)。

```
$ sudo mount --bind /path/to/emp-dir work-dir
```

9.6.16 透過 overlay 掛載 (overlay-mounting) 另一個目錄來擴充套件可用儲存空間

如果你在另一個分割槽表中有可用的空間 (例如，“/path/to/empty” 和“/path/to/work”)，你可以在其中建立一個目錄並堆疊到你需要空間的那個舊的目錄 (例如，“/path/to/old”)，要這樣做，你需要用於 Linux 3.18 版核心或更新版本 (對應 Debian Stetch 9.0 或更新版本) 的 OverlayFS。

```
$ sudo mount -t overlay overlay \
  -olowerdir=/path/to/old-dir,upperdir=/path/to/empty,workdir=/path/to/work
```

“/path/to/empty” 和“/path/to/work” 應該位於可讀寫的分割槽，從而能夠寫入“/path/to/old”。

9.6.17 使用符號連結擴充可用儲存空間

注意
這是一個已棄用的做法。某些軟體在遇到“軟連結目錄”時可能不會正常工作。請優先使用上文所述的“掛載”的途徑。

如果你在另一個分割槽裡有一個帶有可用空間的空目錄（例如“/path/to/emp-dir”），你可以使用 ln(8) 建立目錄的一個符號連結。

```
$ sudo mv work-dir old-dir
$ sudo mkdir -p /path/to/emp-dir
$ sudo ln -sf /path/to/emp-dir work-dir
$ sudo cp -a old-dir/* work-dir
$ sudo rm -rf old-dir
```

警告
別對由系統管理的目錄（例如“/opt”）使用“連結到目錄”，這樣的連結在系統升級時可能會被覆蓋。

9.7 磁碟映像

我們在這裡討論磁碟影響的操作。

9.7.1 製作磁碟映像檔案

一個未掛載裝置（例如，第二個 SCSI 或序列 ATA 裝置“/dev/sdb”）的磁碟映像檔案“disk.img”可以使用 cp(1) 或 dd(1) 通過下列方式建立。

```
# cp /dev/sdb disk.img
# dd if=/dev/sdb of=disk.img
```

傳統 PC 中位於主 IDE 硬碟第一扇區的主引導記錄（MBR）（參見節 9.6.2）的磁碟映像可以使用 dd(1) 通過下列方式建立。

```
# dd if=/dev/hda of=mbr.img bs=512 count=1
# dd if=/dev/hda of=mbr-nopart.img bs=446 count=1
# dd if=/dev/hda of=mbr-part.img skip=446 bs=1 count=66
```

- “mbr.img”：帶有分割槽表的 MBR
- “mbr-nopart.img”：不帶分割槽表的 MBR
- “mbr-part.img”：僅 MBR 的分割槽表

如果你使用 SCSI 或序列 ATA 裝置作為啟動硬碟，你需要使用“/dev/sda”替代“/dev/hda”。

如果你要建立原始硬碟的一個硬碟分割槽的映像，你需要使用“/dev/hda1”等替代“/dev/hda”。

9.7.2 直接寫入硬碟

磁碟映像檔案“disk.img”可以通過下列方式寫入到一個匹配大小的未掛載裝置（例如，第二個 SCSI 裝置“/dev/sdb”。

```
# dd if=disk.img of=/dev/sdb
```

相似地，硬碟分割槽映像檔案“partition.img”可以通過下列方式寫入到匹配大小的未掛載分割槽（例如，第二個 SCSI 裝置的第一個分割槽“/dev/sdb1”）。

```
# dd if=partition.img of=/dev/sdb1
```

9.7.3 掛載磁碟映像檔案

可以使用迴圈裝置通過下列方式掛載和解除安裝包含單個分割槽映像的磁碟映像“partition.img”。

```
# losetup -v -f partition.img
Loop device is /dev/loop0
# mkdir -p /mnt/loop0
# mount -t auto /dev/loop0 /mnt/loop0
...hack...hack...hack
# umount /dev/loop0
# losetup -d /dev/loop0
```

可以簡化為如下步驟。

```
# mkdir -p /mnt/loop0
# mount -t auto -o loop partition.img /mnt/loop0
...hack...hack...hack
# umount partition.img
```

可以使用 迴圈裝置 掛載包含多個分割槽的磁碟映像“disk.img”的每個分割槽。因為迴圈裝置預設不管理分割槽，因此我們需要通過下列方式重新設定它。

```
# modinfo -p loop # verify kernel capability
max_part:Maximum number of partitions per loop device
max_loop:Maximum number of loop devices
# losetup -a # verify nothing using the loop device
# rmmod loop
# modprobe loop max_part=16
```

現在迴圈裝置可以管理多達 16 個分割槽。

```
# losetup -v -f disk.img
Loop device is /dev/loop0
# fdisk -l /dev/loop0

Disk /dev/loop0: 5368 MB, 5368709120 bytes
255 heads, 63 sectors/track, 652 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x452b6464

      Device Boot      Start         End      Blocks   Id  System
/dev/loop0p1               1         600     4819468+  83  Linux
/dev/loop0p2             601         652      417690   83  Linux
# mkdir -p /mnt/loop0p1
# mount -t ext4 /dev/loop0p1 /mnt/loop0p1
# mkdir -p /mnt/loop0p2
# mount -t ext4 /dev/loop0p2 /mnt/loop0p2
...hack...hack...hack
```

```
# umount /dev/loop0p1
# umount /dev/loop0p2
# losetup -d /dev/loop0
```

或者，你也可以使用 kpartx 軟體包中的 kpartx(8) 建立 裝置對映裝置來達到相同的效果。

```
# kpartx -a -v disk.img
...
# mkdir -p /mnt/loop0p2
# mount -t ext4 /dev/mapper/loop0p2 /mnt/loop0p2
...
...hack...hack...hack
# umount /dev/mapper/loop0p2
...
# kpartx -d /mnt/loop0
```

注
你也可以使用迴圈裝置利用偏移量來跳過 MBR 等，來掛載此類磁碟映像的單個分割槽。但這更加容易出錯。

9.7.4 清理磁碟映像檔案

使用下面的方式，一個磁碟映像檔案"disk.img" 能夠清理掉所有已經刪除的檔案，成為一個乾淨的稀疏映像"new.img"。

```
# mkdir old; mkdir new
# mount -t auto -o loop disk.img old
# dd bs=1 count=0 if=/dev/zero of=new.img seek=5G
# mount -t auto -o loop new.img new
# cd old
# cp -a --sparse=always ./ ../new/
# cd ..
# umount new.img
# umount disk.img
```

如果“disk.img”位於 ext2、ext3 或 ext4，你也可以像下面那樣使用 zerofree 軟體包中的 zerofree(8)。

```
# losetup -f -v disk.img
Loop device is /dev/loop3
# zerofree /dev/loop3
# cp --sparse=always disk.img new.img
```

9.7.5 製作空的磁碟映像檔案

按下面的方式使用 dd(1)，可以製作一個大小為 5GiB 的空磁碟映像檔案。

```
$ dd bs=1 count=0 if=/dev/zero of=disk.img seek=5G
```

專業的 fallocate(8) 可以在這裡被使用，用來替代使用 dd(1)。

按下面的方式使用環回裝置，你能夠在這個磁碟映像"disk.img" 上建立 ext4 檔案系統。

```
# losetup -f -v disk.img
Loop device is /dev/loop1
# mkfs.ext4 /dev/loop1
...hack...hack...hack
# losetup -d /dev/loop1
```

```
$ du  --apparent-size -h disk.img
5.0G  disk.img
$ du -h disk.img
83M disk.img
```

對於"disk.img", 它的檔案大小是 5.0 GiB，而它實際磁碟使用僅僅是 83MiB. 這個差距可能是由於 ext4 裡面有稀疏檔案.

提示
稀疏檔案的實際磁碟使用會隨著資料的寫入而增加。

迴環裝置 或 裝置對映 裝置上使用類似的操作，在這些裝置按節 9.7.3 掛載後, 你能夠使用 parted(8) 或 fdisk(8) 對這個磁碟映像"disk.img" 進行分割槽，能夠使用 mkfs.ext4(8), mkswap(8) 在上面建立檔案系統等.

9.7.6 製作 ISO9660 映象檔案

" 源目錄" 下的目錄樹可以通過如下所示的 cdrkit 提供的 genisoimage(1) 指令來製作 ISO9660 映象檔案，"cd.iso"。

```
#  genisoimage -r -J -T -V volume_id -o cd.iso source_directory
```

類似的,可引導的 ISO9660 映象檔案,"cdboot.iso", 能夠從 debian-installer 類似目錄樹"source_directory" 製作，方式如下。

```
#  genisoimage -r -o cdboot.iso -V volume_id \
   -b isolinux/isolinux.bin -c isolinux/boot.cat \
   -no-emul-boot -boot-load-size 4 -boot-info-table source_directory
```

這裡的 Isolinux boot loader (參見節 3.1.2) 是用於啟動的.

按下面的方式，你可以直接從光碟機裝置計算 md5sum 值，並製作 ISO9660 映象。

```
$ isoinfo -d -i /dev/cdrom
CD-ROM is in ISO 9660 format
...
Logical block size is: 2048
Volume size is: 23150592
...
# dd if=/dev/cdrom bs=2048 count=23150592 conv=notrunc,noerror | md5sum
# dd if=/dev/cdrom bs=2048 count=23150592 conv=notrunc,noerror > cd.iso
```

警告
為了得到正確結果，你必須小心避免 Linux ISO9600 檔案系統預讀 bug。

9.7.7 直接寫入檔案到 CD/DVD-R/RW

提示
對於由 cdrkit 提供的 wodim(1) 來講，DVD 僅僅是一個大的 CD。

你能夠通過如下所示的指令找到可用的裝置。

```
# wodim --devices
```

然後將空的 CD-R 插入 CD 驅動器並且把 ISO9660 映象檔案，”cd.iso” 寫入到裝置中，例如用如下所示的 wodim(1) 將資料寫入到”/dev/hda” 裝置。

```
# wodim -v -eject dev=/dev/hda cd.iso
```

如果用 CD-RW 代替 CD-R，用如下所示的指令來替代。

```
# wodim -v -eject blank=fast dev=/dev/hda cd.iso
```

提示
如果你的桌面系統自動掛載 CDs，在使用 wodim(1) 之前在終端裡面用”sudo umount /dev/hda” 解除安裝它。

9.7.8 掛載 ISO9660 映象檔案

如果”cd.iso” 包含一個 ISO9660 映象, 下面的指令手工掛載這個檔案到”/cdrom”.

```
# mount -t iso9660 -o ro,loop cd.iso /cdrom
```

提示
現代桌面系統能夠自動掛載可移動介質，如按 ISO9660 格式化的 CD(參見節 10.1.7).

9.8 二進位制資料

這裡，我們討論直接操作儲存介質上的二進位制資料。

9.8.1 檢視和編輯二進位制資料

最基礎的檢視二進位制資料的方法是使用”od -t x1” 指令。

軟體包	流行度	大小	說明
coreutils	V:879, I:999	18307	基礎軟體包，有 od(1) 來匯出檔案 (HEX, ASCII, OCTAL, ⋯)
bsdmainutils	V:12, I:332	17	工具軟體包，有 hd(1) 來匯出檔案 (HEX, ASCII, OCTAL, ⋯)
hexedit	V:0, I:9	73	二進位制瀏覽和編輯器 (HEX, ASCII)
bless	V:0, I:2	924	全功能的十六進位制編輯器 (GNOME)
okteta	V:0, I:12	1585	全功能的十六進位制編輯器 (KDE4)
ncurses-hexedit	V:0, I:1	130	二進位制瀏覽和編輯器 (HEX, ASCII, EBCDIC)
beav	V:0, I:0	137	二進位制瀏覽和編輯器 (HEX, ASCII, EBCDIC, OCTAL, ...)

Table 9.21: 檢視和修改二進位制資料的軟體包列表

提示
HEX 是十六進位制英文hexadecimal首字母縮略詞，基數 radix 是 16。OCTAL 是八進位制英文octal 首字母縮略詞，基數 radix是 8。ASCII 是美國資訊交換標準程式碼 American Standard Code for Information Interchange 的英文縮寫，即正常的英語文字程式碼。EBCDIC 是擴充二進位制編碼十進位制交換碼 Extended Binary Coded Decimal Interchange Code 的英文縮寫，在 IBM 大型機 作業系統上使用。

9.8.2 不掛載磁碟操作檔案

有工具可以在沒有掛載磁碟的情況下讀寫檔案。

軟體包	流行度	大小	說明
mtools	V:9, I:64	390	不掛載磁碟的 MSDOS 檔案工具
hfsutils	V:0, I:5	184	不掛載磁碟的 HFS 和 HFS+ 檔案工具

Table 9.22: 不掛載磁碟操作檔案的軟體包列表

9.8.3 資料冗餘

Linux 核心所提供的RAID軟體系統提供核心檔案系統級別的資料冗餘來實現高水平的儲存可靠性。

有在應用程式級別增加資料冗餘來實現高水平儲存可靠性的工具。

軟體包	流行度	大小	說明
par2	V:9, I:91	268	奇偶校驗檔案卷設定，用於檢查和修復檔案
dvdisaster	V:0, I:1	1422	CD/DVD 媒體資料損失/劃傷/老化的保護
dvbackup	V:0, I:0	413	使用 MiniDV 行動式攝像機的備份工具 (提供 rsbep(1))

Table 9.23: 向檔案新增資料冗餘的工具列表

9.8.4 資料檔案恢復和診斷分析

有用於資料檔案恢復和診斷分析的工具。

軟體包	流行度	大小	說明
testdisk	V:2, I:29	1413	分割槽掃描和磁碟恢復的實用程式
magicrescue	V:0, I:2	255	通過查詢幻數 magic 位元組來恢復檔案的工具 (譯註: 請 man file 來了解幻數)
scalpel	V:0, I:3	88	簡潔、高效能的檔案提取
myrescue	V:0, I:2	83	恢復損壞硬碟中的資料
extundelete	V:0, I:8	147	恢復刪除 ext3/4 檔案系統上的檔案的實用程式
ext4magic	V:0, I:4	233	恢復刪除 ext3/4 檔案系統上的檔案的實用程式
ext3grep	V:0, I:2	293	幫助恢復 ext3 檔案系統上刪除的檔案的工具
scrounge-ntfs	V:0, I:2	50	NTFS 檔案系統的資料恢復程式
gzrt	V:0, I:0	33	gzip 恢復工具包
sleuthkit	V:3, I:24	1671	診斷分析工具 (Sleuthkit)
autopsy	V:0, I:1	1026	SleuthKit 的圖形化介面
foremost	V:0, I:5	102	恢復資料的診斷程式
guymager	V:0, I:0	1021	基於 Qt 的診斷影象工具
dcfldd	V:0, I:3	114	增強版的 dd，用於診斷和安全

Table 9.24: 資料檔案恢復和診斷分析軟體包列表

提示
在 e2fsprogs 軟體包裡有 debugfs(8) 命令，使用該指令裡的 list_deleted_inodes 和 undel 指令，你能夠恢復 ext2 檔案系統上刪除的檔案。

9.8.5 把大檔案分成多個小檔案

當一個檔案太大而不能備份的時候，你應該在備份之前先把它分割為多個小於 2000MiB 的小檔案，稍後再把這些小檔案合併為初始的檔案。

```
$ split -b 2000m large_file
$ cat x* >large_file
```

注意
為了防止檔名衝突，請確保沒有任何以"x" 開頭的檔案。

9.8.6 清空檔案內容

為了清除諸如日誌檔案之類的檔案的內容，不要用 rm(1) 指令去刪除檔案然後建立新的空檔案，因為這個檔案可能在指令執行的期間還在被使用。以下是清除檔案內容的正確方法。

```
$ :>file_to_be_cleared
```

9.8.7 樣子文件

下面的指令創建樣子文件或空文件。

```
$ dd if=/dev/zero    of=5kb.file bs=1k count=5
$ dd if=/dev/urandom of=7mb.file bs=1M count=7
$ touch zero.file
$ : > alwayszero.file
```

你將發現下列文件。

- "5kb.file" 是 5KB 零。
- "7mb.file" 是 7MB 隨機數據。
- "zero.file" 也許是一個 0 字節的文件。如果這個文件之前就存在，則它的 mtime 會被更新，而它的內容和長度保持不變。
- "alwayszero.file" 一定是一個 0 字節文件。如果這個文件之前存在，則它的 mtime 會被更新，而它的內容會被清零。

9.8.8 擦除整塊硬碟

有幾種方法來完全擦除裝置上整個硬碟上資料，比如說，在"/dev/sda" 上的 USB 記憶體盤。

注意
在執行這裡的指令之前，你應該用 mount(8) 指令來檢視 USB 記憶棒的掛載位置。"/dev/sda" 指向的裝置可能是裝有整個系統的 SCSI 硬碟或者 serial-ATA 硬碟。

如下所示是通過資料歸 0 的方式來擦除硬碟上所有資料的。

```
# dd if=/dev/zero of=/dev/sda
```

如下是用隨機資料重寫的方式來擦除所有資料的。

```
# dd if=/dev/urandom of=/dev/sda
```

如下是用隨機資料重寫的方式來高效擦除所有資料。

```
# shred -v -n 1 /dev/sda
```

你或者可以使用 badblocks(8) 加上 -t random 選項。

因為 dd(1) 指令在許多可引導的 Linux CDs (例如 Debian 安裝光碟) 上的 shell 環境下都是可用的，你能夠在裝有系統的硬碟上，例如"/dev/hda", "/dev/sda" 等等裝置上執行擦除指令來完全清除已經安裝的系統。

9.8.9 擦除硬碟上的未使用的區域

硬碟（或 USB 記憶棒）上未使用的區域，例如“/dev/sdb1”可能仍然包含可被擦除的資料，因為他們本身只是解除了從檔案系統的連結，這些可以通過重寫來清除。

```
# mount -t auto /dev/sdb1 /mnt/foo
# cd /mnt/foo
# dd if=/dev/zero of=junk
dd: writing to ‘junk’: No space left on device
...
# sync
# umount /dev/sdb1
```

警告
這對您的 USB 記憶棒來說通常已經足夠好了，但這還不完美。大部分已擦除的檔名和它們的屬性可能隱藏並留在檔案系統中。

9.8.10 恢復已經刪除但仍然被開啟的檔案

即使你不小心刪除了某個檔案，只要這個檔案仍然被一些應用程式所使用（讀或者寫），恢復此檔案是可能的。

嘗試下列例子

```
$ echo foo > bar
$ less bar
$ ps aux | grep ’ less[ ]’
bozo    4775  0.0  0.0  92200   884 pts/8    S+   00:18   0:00 less bar
$ rm bar
$ ls -l /proc/4775/fd | grep bar
lr-x------ 1 bozo bozo 64 2008-05-09 00:19 4 -> /home/bozo/bar (deleted)
$ cat /proc/4775/fd/4 >bar
$ ls -l
-rw-r--r-- 1 bozo bozo 4 2008-05-09 00:25 bar
$ cat bar
foo
```

當你安裝了 lsof 軟體包的時候，在另外一個終端執行如下指令。

```
$ ls -li bar
2228329 -rw-r--r-- 1 bozo bozo 4 2008-05-11 11:02 bar
$ lsof |grep bar|grep less
less 4775 bozo 4r REG 8,3 4 2228329 /home/bozo/bar
$ rm bar
$ lsof |grep bar|grep less
less 4775 bozo 4r REG 8,3 4 2228329 /home/bozo/bar (deleted)
$ cat /proc/4775/fd/4 >bar
$ ls -li bar
2228302 -rw-r--r-- 1 bozo bozo 4 2008-05-11 11:05 bar
$ cat bar
foo
```

9.8.11 查詢所有硬連結

有硬連結的檔案，能夠使用”`ls -li`” 確認。

```
$ ls -li
total 0
2738405 -rw-r--r-- 1 root root 0 2008-09-15 20:21 bar
2738404 -rw-r--r-- 2 root root 0 2008-09-15 20:21 baz
2738404 -rw-r--r-- 2 root root 0 2008-09-15 20:21 foo
```

”`baz`” 和”`foo`” 的連結數為”2” (>1)，表示他們有硬連結。它們的 inode 號都是”2738404”. 這表示它們是同樣的硬連結檔案。如果你不想偶然碰巧發現硬連結檔案，你可以通過 inode 號來查詢它。比如說, 按下面的方式查詢”2738404” 。

```
# find /path/to/mount/point -xdev -inum 2738404
```

9.8.12 不可見磁碟空間消耗

所有開啟的檔案被刪除後，仍然消耗磁碟空間，儘管他們不能夠被普通的 du(1) 所看見。這些被刪除的檔案和他們的大小，可以通過下面的方式列出。

```
# lsof -s -X / |grep deleted
```

9.9 資料加密提示

在可以物理存取您的 PC 的情況下，任何人都可以輕易獲得 root 許可權，存取您的 PC 上的所有檔案 (見節 4.6.4)。這意味著登入密碼系統在您的 PC 被偷盜時並不能保證您私人和敏感資料的安全。您必須部署資料加密技術來實現。儘管 GNU 隱私守護 (見節 10.3) 可以對檔案進行加密，但它需要一些使用者端的工作。

Dm-crypt 透過原生的 Linux 核心模組，使用 device-mapper，用很少的使用者操作實現本地自動資料加密。

軟體包	流行度	大小	說明
cryptsetup	V:34, I:79	410	可用於加密的塊裝置的實用程式 (dm-crypt / 3LUKS)
cryptmount	V:2, I:3	231	可用於加密的塊裝置著重於正常使用者掛載/解除安裝的實用程式 (dm-crypt / LUKS)
fscrypt	V:0, I:1	5520	可用於加密檔案系統的實用程式 (fscrypt)
libpam-fscrypt	I:0	5519	可用於加密檔案系統的實用程式 (fscrypt)

Table 9.25: 資料加密工具列表

注意
資料加密會消耗 CPU 時間等資源，如果它的密碼丟失，加密的資料會變成無法訪問。請權衡其利弊。

注
通過 debian-installer (lenny 或更新版)，整個 Debian 系統能夠被安裝到一個加密的磁碟上，使用 dm-crypt/LUKS 和 initramfs.

提示
請參閱節 10.3 使用者空間加密實用程式：GNU Privacy Guard。

9.9.1 使用 dm-crypt/LUKS 加密移動磁碟

您可以用dm-crypt/LUKS加密大容量可移動裝置上資料，例如掛載在“/dev/sdx”上的 USB 記憶棒。你只需按如下步驟簡單地把它格式化。

```
# fdisk /dev/sdx
... "n" "p" "1" "return" "return" "w"
# cryptsetup luksFormat /dev/sdx1
...
# cryptsetup open /dev/sdx1 secret
...
# ls -l /dev/mapper/
total 0
crw-rw---- 1 root root  10, 60 2021-10-04 18:44 control
lrwxrwxrwx 1 root root       7 2021-10-04 23:55 secret -> ../dm-0
# mkfs.vfat /dev/mapper/secret
...
# cryptsetup close secret
```

然後，它就可以正常的在現代桌面環境下使用 udisks2 軟體包，掛載到"/media/*username/disk_label*"，只不過它會要求輸入密碼 (參見節 10.1.7)。不同的是寫入的資料都是加密的。密碼條目可以自動使用鑰匙環 (參見節 10.3.6)。

你可以把它格式化成其他格式的檔案系統，例如用"mkfs.ext4 /dev/mapper/sdx1" 把它格式化為 ext4。如果使用 btrfs，則需要安裝 udisks2-btrfs 軟體包。對於這些檔案系統，可能需要配置檔案的所有者和許可權。

9.9.2 使用 dm-crypt/LUKS 掛載加密的磁碟

舉個列子，用 dm-crypt/LUKS 在"/dev/sdc5" 上建立的加密磁碟可以用如下步驟掛載到"/mnt":

```
$ sudo cryptsetup open /dev/sdc5 ninja --type luks
Enter passphrase for /dev/sdc5: ****
$ sudo lvm
lvm> lvscan
  inactive          '/dev/ninja-vg/root' [13.52 GiB] inherit
  inactive          '/dev/ninja-vg/swap_1' [640.00 MiB] inherit
  ACTIVE            '/dev/goofy/root' [180.00 GiB] inherit
  ACTIVE            '/dev/goofy/swap' [9.70 GiB] inherit
lvm> lvchange -a y /dev/ninja-vg/root
lvm> exit
  Exiting.
$ sudo mount /dev/ninja-vg/root /mnt
```

9.10 核心

對於支援的架構，Debian 使用軟體包來分發模組化的 Linux 核心.
如果你正在閱讀本文件，你可能不需要親自編譯核心。

9.10.1 核心參數

許多 Linux 特性可以按下面的方式，通過核心參數來調配。

- 核心參數通過 bootloader 初始化 (參見節 3.1.2)
- 對通過 sysfs 存取的核心參數，在執行時通過 `sysctl`(8) 修改 (參見節 1.2.12)
- 當一個模組被啟用時，通過 `modprobe`(8) 參數來設定模組參數。(參見節 9.7.3)

細節參見”The Linux kernel user’s and administrator’s guide » The kernel’s command-line parameters” 。

9.10.2 核心標頭檔案

大部分普通程式編譯時不需要核心標頭檔案，如果你直接使用它們來編譯，甚至會導致編譯中斷。在 Debian 系統上，普通程式編譯依賴 `libc6-dev` 軟體包 (由 `glibc` 原始碼包建立) 提供的，在”`/usr/include/linux`” 和”`/usr/include/asm`” 裡的標頭檔案。

注
對於編譯一些核心相關的程式，比如說從外部原始碼編譯的核心模組和 automounter 後臺守護 (daemon) 程式 (amd)，你必須包含相應的核心標頭檔案到路徑裡，比如”`-I/usr/src/linux-particular-version/include/`”, 到你的命令列。

9.10.3 編譯核心和相關模組

Debian 有它自己的方式來編譯核心和相關模組。

軟體包	流行度	大小	說明
`build-essential`	I:480	17	建立 Debian 軟體包所必須的軟體包: `make`, `gcc`, ⋯
`bzip2`	V:163, I:969	112	bz2 檔案壓縮和解壓縮工具
`libncurses5-dev`	I:72	6	ncurses 開發者庫和文件
`git`	V:347, I:545	46734	git: Linux 核心使用的分散式版本控制系統
`fakeroot`	V:28, I:487	224	為非 root 使用者建立軟體包提供一個偽造的 root 環境
`initramfs-tools`	V:433, I:990	113	建立 initramfs 的工具 (Debian 規範)
`dkms`	V:76, I:163	195	動態核心模組支援 dynamic kernel module support (DKMS) (通用)
`module-assistant`	V:1, I:20	406	製作模組軟體包的幫助工具 (Debian 規範)
`devscripts`	V:6, I:40	2658	Debian Package maintainer Debian 包維護者的幫助指令碼 (Debian 規範)

Table 9.26: Debian 系統核心編譯需要安裝的主要軟體包列表

如果你在節 3.1.2 使用 `initrd` , 請一定閱讀 `initramfs-tools`(8), `update-initramfs`(8), `mkinitramfs`(8) 和 `initramfs.conf`(5) 裡的相關資訊。

警告
在編譯 Linux 核心原始碼時，請不要放置從"/usr/include/linux" 和"/usr/include/asm" 到原始碼樹 (比如："/usr/src/linux*") 裡目錄的符號連結。(一些過期的文件建議這樣做.)

注
當在 Debian stable 版裡編譯最新的 Linux 核心時, 可能需要使用一些從 Debian unstable 版裡 backported 向後移植的最新版本的工具。
module-assistant(8) (或者它的短形式 m-a) 幫助使用者，從一個或多個定製的核心，容易的構建和安裝模組軟體包。
dynamic kernel module support (DKMS) 動態核心模組支援 是一個新的分散式獨立框架，被設計用來允許單個的核心模組在不改變整個核心的情況下升級。這可以用於維護核心程式碼樹外部的模組。這也使你升級核心時，重新編譯模組變得非常簡單。

9.10.4 編譯核心原始碼：Debian 核心團隊推薦

從上游核心原始碼編譯個性化的核心二進位制包，你應當使用由它提供的"deb-pkg" 物件。

```
$ sudo apt-get build-dep linux
$ cd /usr/src
$ wget https://mirrors.edge.kernel.org/pub/linux/kernel/v6.x/linux-version.tar.xz
$ tar --xz -xvf linux-version.tar.xz
$ cd linux-version
$ cp /boot/config-version .config
$ make menuconfig
 ...
$ make deb-pkg
```

提示
linux-source-*version* 軟體包使用"/usr/src/linux-*version*.tar.bz2" 提供有 Debian 補丁的 Linux 核心原始碼。

從 Debian 核心原始碼軟體包編譯特定的二進位制包,你應當使用"debian/rules.gen" 裡的"binary-arch_*architectu*
物件。

```
$ sudo apt-get build-dep linux
$ apt-get source linux
$ cd linux-3.*
$ fakeroot make -f debian/rules.gen binary-arch_i386_none_686
```

進階資訊參見：

- Debian Wiki: KernelFAQ
- Debian Wiki: DebianKernel
- Debian Linux 核心手冊：https://kernel-handbook.debian.net

9.10.5 硬體驅動和韌體

硬體驅動是執行在目標系統上主 CPU 上的程式碼。大部分硬體驅動現在是自由軟體，已經包含在普通的 Debian 核心軟體包裡，放在 main 區域。

- GPU 驅動
  - Intel GPU 驅動 (`main`)
  - AMD/ATI GPU 驅動 (`main`) 和/
  - NVIDIA GPU 驅動 (nouveau 驅動放在 `main` , 由廠家支援的二進位制驅動，放在 `non-free`.)

韌體是載入在連線在目標系統裝置上的程式碼或資料 (比如說，CPU microcode、GPU 執行的渲染程式碼或 FPGA / CPLD 資料……) 部分韌體包是作為自由軟體存在，但是很多韌體包由於包含有沒有原始碼的資料，二進位制不是作為自由軟體存在。安裝這些韌體資料是必需的，這樣裝置才能按期望執行。

- 韌體資料軟體包，含有載入到目標裝置易失性儲存器上的資料。
  - firmware-linux-free (`main`)
  - firmware-linux-nonfree (`non-free-firmware`)
  - firmware-linux-* (`non-free-firmware`)
  - *-firmware (`non-free-firmware`)
  - intel-microcode (`non-free-firmware`)
  - amd64-microcode (`non-free-firmware`)
- 韌體更新程式軟體包，更新在目標裝置非易失性儲存器上的資料。
  - fwupd (`main`): 從 Linux Vendor Firmware Service 下載韌體資料的韌體更新後臺守護程序 (daemon)。
  - gnome-firmware (`main`): fwupd 的 GTK 前端
  - plasma-discover-backend-fwupd (`main`): fwupd 的 Qt 前端

請注意，從 Debian 12 Bookworm 開始，使用者使用由官方安裝介質裡面提供的 `non-free-firmware` 軟體包來提供完善的安裝體驗。`non-free-firmware` 區域在節 2.1.5 裡面描述。

也請注意到，fwupd 從 Linux Vendor Firmware Service 下載的韌體資料並載入到正在執行的 Linux 核心，可能是 `non-free`。

9.11 虛擬化系統

通過使用虛擬系統，我們能在單個機器上同時執行多個系統。

提示
參見 Debian wiki 的系統虛擬化。

9.11.1 虛擬化和模擬器工具

有幾個 虛擬化 和模擬器工具平臺。

- 完全的 硬體模擬 軟體包，比如透過 games-emulator 元軟體包安裝的軟體包
- 大部分 CPU 層的模擬，加上一些 I/O 裝置的模擬，比如 QEMU
- 大部分 CPU 層的虛擬化，加上一些 I/O 裝置的模擬，比如 Kernel-based Virtual Machine (KVM)
- 作業系統級的容器虛擬化，加上核心級的支援，比如 LXC (Linux Containers), Docker, `systemd-nspawn`(1), ...
- 作業系統級的檔案系統訪問虛擬化，加上在檔案路徑上的系統庫呼叫，比如 chroot
- 作業系統級的檔案系統訪問虛擬化，加上在檔案所有者許可權上的系統庫呼叫，比如 fakeroot

- 作業系統 API 模擬器，比如 Wine
- 直譯器級的虛擬化，加上它的執行選擇和執行時庫忽略，比如 Python 的 virtualenv 和 venv

容器虛擬化使用節 4.7.5，是節 7.6 的後端技術。

這裡有一些幫你搭建虛擬化系統的軟體包。

軟體包	流行度	大小	說明
coreutils	V:879, I:999	18307	GNU 核心工具包含 chroot(8)
systemd-container	V:53, I:60	1328	systemd container/nspawn 工具包含 systemd-nspawn(1)
schroot	V:5, I:7	2579	在 chroot 下執行 Debian 二進位制包的特異工具
sbuild	V:1, I:3	242	從 Debian 原始碼構建 Debian 二進位制包的工具
debootstrap	V:5, I:55	314	搭建一個基本的 Debian 系統 (用 sh 寫的)
cdebootstrap	V:0, I:1	115	搭建一個 Debian 系統 (用 C 寫的)
cloud-image-utils	V:1, I:17	66	雲映象管理工具
cloud-guest-utils	V:3, I:12	71	雲客戶機工具
virt-manager	V:11, I:44	2296	虛擬機器管理器: 用於管理虛擬機器的桌面應用
libvirt-clients	V:46, I:65	1241	libvirt 的庫程式
lxd	V:0, I:0	52119	LXD: 系統容器和虛擬機器管理器
podman	V:13, I:15	41928	podman：在 Pods 裡面執行基於 OCI 容器的引擎
podman-docker	V:0, I:0	248	在 Pods 裡面執行基於 OCI 容器的引擎 -- docker 封裝器
docker.io	V:40, I:43	150003	docker: Linux 容器執行時
games-emulator	I:0	21	games-emulator: Debian 的遊戲模擬器
bochs	V:0, I:0	6956	Bochs: IA-32 PC 模擬器
qemu	I:15	97	QEMU: 快速的通用處理器模擬器
qemu-system	I:22	66	QEMU: 全功能系統的模擬二進位制
qemu-user	V:1, I:6	93764	QEMU: 使用者模式的模擬二進位制
qemu-utils	V:14, I:107	10635	QEMU: 工具集
qemu-system-x86	V:35, I:91	58128	KVM: x86 硬體上有 硬體輔助虛擬化的全虛擬化
virtualbox	V:6, I:8	126272	VirtualBox:i386 和 amd64 上 x86 的虛擬化解決方案
gnome-boxes	V:1, I:7	6691	Boxes：用於訪問虛擬機器系統的簡單的 GNOME 應用程式
xen-tools	V:0, I:2	719	用於管理 debian XEN 虛擬伺服器的工具
wine	V:13, I:60	133	Wine: Windows 應用程式程式設計介面實現（標準套件）
dosbox	V:1, I:15	2696	DOSBox: 有 Tandy/Herc/CGA/EGA/VGA/SVGA 顯示卡, 聲音和 DOS 的 x86 模擬器
lxc	V:9, I:12	25889	Linux 容器使用者層工具
python3-venv	I:86	6	venv 建立虛擬的 python 環境（系統庫）
python3-virtualenv	V:9, I:51	356	virtualenv 建立隔離的虛擬 python 環境
pipx	V:3, I:16	3308	pipx 在隔離的環境中安裝 python 應用程式

Table 9.27: 虛擬化工具列表

參見維基百科 Comparison of platform virtual machines 來獲得不同平臺的虛擬化解決方案的詳細比較資訊。

9.11.2 虛擬化工作流

注
自從 lenny 之後，預設的 Debian 核心就是支援 KVM 的。

典型的虛擬化工作流涉及以下幾個步驟。

- 建立空檔案系統 (目錄樹或磁碟映像)。
  - 目錄樹可以通過"`mkdir -p /path/to/chroot`" 建立。
  - 原始的磁碟映像檔案能夠使用 dd(1) 建立 (參見節 9.7.1 和節 9.7.5).
  - `qemu-img`(1) 能夠建立和轉化 QEMU 支援的磁碟映像檔案.
  - 原始的格式和 VMDK 檔案格式, 能夠作為虛擬化工具的通用格式.
- 使用 `mount`(8) 掛載磁碟映像到檔案系統 (可選).
  - 對於原始磁碟映像檔案, 把它作為迴環裝置 或 裝置對映 裝置掛載. (參見節 9.7.3).
  - 對於 QEMU 支援的磁碟映像, 把它們作為 network block device 網路塊裝置掛載 (參見節 9.11.3).
- 在目標檔案系統上部署需要的系統資料。
  - 使用 `debootstrap` 和 `cdebootstrap` 之類的程式來協助處理這個過程 (參見節 9.11.4).
  - 在全功能系統模擬器下使用作業系統安裝器。
- 在虛擬化環境下執行一個程式。
  - chroot 提供基本的虛擬化環境, 足夠能在裡面編譯程式, 執行控制檯應用, 執行後臺程式 daemon.
  - QEMU 提供跨平臺的 CPU 模擬器.
  - QEMU 和 KVM 通過 hardware-assisted virtualization 硬體輔助虛擬化來提供全功能系統的模擬.
  - VirtualBox 可以在 i386 和 amd64 上, 使用或者不使用 hardware-assisted virtualization 硬體輔助虛擬化來提供全功能系統模擬.

9.11.3 掛載虛擬磁碟映像檔案

對於原始磁碟映像檔案, 參見節 9.7.

對於其它虛擬磁碟映像檔案, 你能夠用使用 network block device 網路塊裝置 協議的 `qemu-nbd`(8) 來匯出他們, 並使用核心模組 `nbd` 來掛載它們.

`qemu-nbd`(8) 支援 QEMU 所支援的磁碟格式: QEMU 支援下列磁碟格式: raw, qcow2, qcow, vmdk, vdi, bochs, cow (user-mode Linux copy-on-write), parallels, dmg, cloop, vpc, vvfat (virtual VFAT) 和主機裝置.

網路塊裝置 能夠用和迴環裝置一樣的方式支援分割槽 (參見節 9.7.3). 你能夠按下面的方式掛載"`disk.img`" 的第一個分割槽.

```
# modprobe nbd max_part=16
# qemu-nbd -v -c /dev/nbd0 disk.img
...
# mkdir /mnt/part1
# mount /dev/nbd0p1 /mnt/part1
```

提示
你可以給 `qemu-nbd`(8) 使用"`-P 1`" 選項來匯出"`disk.img`" 的第一個分割槽.

9.11.4 Chroot 系統

如果你希望從終端控制檯嘗試一個新的 Debian 環境，我推薦你使用 chroot。這使你能夠執行 unstable 和 testing 的控制檯應用程式，不會有通常的相關風險，並且不需要重啟。chroot(8) 是最基礎的方法。

注意
下面的列子假設根源系統和 chroot 系統都共享相同的 amd64 CPU 架構。

雖然你能夠手工使用 debootstrap(1) 來建立一個 chroot(8) 環境，這要求瑣碎的工作。

sbuild 軟體包從原始碼構建一個 Debian 軟體包，使用 schroot 軟體包管理的 chroot 環境。它和幫助指令碼 sbuild-createchro 一起。讓我們按如下所示的方式執行它，來學會它是怎樣工作的。

```
$ sudo mkdir -p /srv/chroot
$ sudo sbuild-createchroot -v --include=eatmydata,ccache unstable /srv/chroot/unstable- ↩
    amd64-sbuild http://deb.debian.org/debian
 ...
```

你能夠看到 debootstrap(8) 是如何在"/srv/chroot/unstable-amd64-sbuild" 下部署 unstable 環境的系統資料, 用於一個最小的構建系統。

你可以使用 schroot(1) 來登入到這個環境。

```
$ sudo schroot -v -c chroot:unstable-amd64-sbuild
```

你可以看到一個執行在 unstable 環境的系統 shell 是如何建立的。

注
"/usr/sbin/policy-rc.d" 檔案總是用 101 退出，阻止在 Debian 系統上自動啟動後臺守護程式。參見"/usr/share/doc/init-system-helpers/README.policy-rc.d.gz"。

注
一些在 chroot 下的程式，需要訪問比上面根源系統上的 sbuild-createchroot 能夠提供的檔案之外的更多檔案才能夠執行。例如，"/sys", "/etc/passwd", "/etc/group", "/var/run/utmp", "/var/log/wtmp" 等等. 也許需要使用 bind-mounted 或複製.

提示
sbuild 軟體包幫助建立一個 chroot 系統來構建一個軟體包，在 chroot 內使用 schroot 作為它的後端。它是一個檢查構建依賴 (build-dependencies) 的理想系統。更多資訊參見 sbuild at Debian wiki 和在"Guide for Debian Maintainers" 中的 sbuild 配置樣例 。

提示
systemd-nspawn(1) 命令使用 chroot 類似的方法幫助執行一個命令，或者輕量級容器內的作業系統。它更強大，因為它使用名稱空間來完全虛擬化程序樹、程序間通訊、主機名、域名，並可選網路和使用者資料庫。參見 systemd-nspawn。

9.11.5 多桌面系統

如果你希望嘗試任一作業系統的一個新的 GUI 桌面環境，我推薦在 Debian 穩定版系統上使用 QEMU或者KVM，這些軟體應用虛擬化技術安全的執行多桌面系統。這能讓你執行任何桌面應用，包括 Debian 不穩定版和測試版上的桌面應用，並且沒有與之相關的通常意義上的風險，並且不需要重啟。

因為單純的 QEMU 工具是非常慢的，當主機系統支援 KVM 的時候，建議使用它來加速。

虛擬機器管理器，也被稱為 `virt-manager`，它是一個便利的管理 KVM 虛擬機器的 GUI（圖形使用者介面）工具，它呼叫 libvirt。

按下面的方法，能夠建立一個可以用於QEMU 的包含有 Debian 系統的虛擬磁碟映像”`virtdisk.qcow2`”，這個 Debian 系統使用 debian 安裝器: 小 CD安裝。

```
$ wget https://cdimage.debian.org/debian-cd/5.0.3/amd64/iso-cd/debian-503-amd64-netinst.iso
$ qemu-img create -f qcow2 virtdisk.qcow2 5G
$ qemu -hda virtdisk.qcow2 -cdrom debian-503-amd64-netinst.iso -boot d -m 256
...
```

提示
在虛擬化下執行 Ubuntu 和 Fedora 之類的其它 GNU/Linux 發行版，是一個不錯的學習其調配技巧的方法。其它專有作業系統也可以在這個 GNU/Linux 虛擬化下很好的執行。

在 Debian wiki: SystemVirtualization 參看更多技巧。

Chapter 10

資料管理

以下是關於在 Debian 系統上管理二進位制和文字資料的工具及其相關提示。

10.1 共享，拷貝和存檔

警告
為避免競爭情況，不應當對正在進行寫操作的裝置和檔案，多個程序進行不協調的寫操作。採用 `flock`(1) 的 檔案鎖定 機制可用於避免這種情況。

資料的安全和它的受控共享有如下幾個方面。

- 存檔檔案的建立
- 遠端儲存存取
- 複製
- 跟蹤修改歷史
- 促進資料共享
- 防止未經授權的檔案存取
- 檢測未經授權的檔案修改

這些可以通過使用工具集來實現。

- 存檔和壓縮工具
- 複製和同步工具
- 網路檔案系統
- 移動儲存媒介
- 安全 shell
- 認證體系
- 版本控制系統工具
- 雜湊演算法和加密工具

10.1.1 存檔和壓縮工具

以下是 Debian 系統上可用的存檔和壓縮工具的預覽。

軟體包	流行度	大小	副檔名	指令	描述
tar	V:909, I:999	3077	.tar	tar(1)	標準的歸檔工具（預設）
cpio	V:433, I:998	1199	.cpio	cpio(1)	Unix System V 風格的歸檔器，與 find(1) 一起使用
binutils	V:173, I:629	144	.ar	ar(1)	建立靜態庫的歸檔工具
fastjar	V:1, I:14	183	.jar	fastjar(1)	Java 歸檔工具（類似 zip）
pax	V:8, I:15	170	.pax	pax(1)	新的 POSIX 歸檔工具，介於 tar 和 cpio 之間
gzip	V:877, I:999	252	.gz	gzip(1), zcat(1), …	GNU LZ77 壓縮工具（預設）
bzip2	V:163, I:969	112	.bz2	bzip2(1), bzcat(1), …	Burrows-Wheeler block-sorting 壓縮工具有著比 gzip(1) 更高的壓縮率 (跟 gzip 有著相似的語法但速度比它慢)
lzma	V:1, I:17	149	.lzma	lzma(1)	LZMA 壓縮工具有著比 gzip(1) 更高的壓縮率（不推薦）
xz-utils	V:359, I:980	1258	.xz	xz(1), xzdec(1), …	XZ 壓縮工具有著比 bzip2(1) 更高的壓縮率（壓縮速度慢於 gzip 但是比 bzip2 快; LZMA 壓縮工具的替代品）
zstd	V:182, I:451	2158	.zstd	zstd(1), zstdcat(1), …	Zstandard 快速無失真壓縮工具
p7zip	V:21, I:472	8	.7z	7zr(1), p7zip(1)	有著更高壓縮率的 7-zip 檔案歸檔器（LZMA 壓縮）
p7zip-full	V:121, I:477	12	.7z	7z(1), 7za(1)	有著更高壓縮率的 7-Zip 檔案歸檔器（LZMA 壓縮和其他）
lzop	V:15, I:140	164	.lzo	lzop(1)	LZO 壓縮工具有著比 gzip(1) 更高的壓縮和解壓縮速度 (跟 gzip 有著相似的語法但壓縮率比它低)
zip	V:48, I:380	616	.zip	zip(1)	InfoZip: DOS 歸檔器和壓縮工具
unzip	V:103, I:771	379	.zip	unzip(1)	InfoZIP: DOS 解檔器和解壓縮工具

Table 10.1: 存檔和壓縮工具列表

警告
除非你知道將會發生什麼，否則不要設定"$TAPE" 變數。它會改變 tar(1) 的行為。

- gzipped tar(1) 歸檔器用於副檔名是".tgz" 或者".tar.gz" 的檔案。
- xz-compressed tar(1) 歸檔器用於副檔名是".txz" 或者".tar.xz" 的檔案。
- FOSS 工具，例如 tar(1)，中的主流壓縮方法已經按如下所示的遷移: gzip → bzip2 → xz
- cp(1),scp(1) 和 tar(1) 工具可能並不適用於一些特殊的檔案。cpio(1) 工具的適用範圍是最廣的。
- cpio(1) 是被設計為與 find(1) 和其它指令一起使用，適合於建立備份指令碼的場景，因此，指令碼的檔案選擇部分能夠被獨立測試。
- Libreoffice 資料檔案的內部結構是".jar" 檔案，它也可以使用 unzip 工具來開啟。
- 事實上跨平臺支援最好的存檔工具是 zip。按照“zip -rX”的方式呼叫可以獲得最大的相容性。如果最大檔案大小需要納入考慮範圍，請同時配合“-s”選項使用。

10.1.2 複製和同步工具

以下是 Debian 系統上的可用的簡單複製和備份工具的預覽。

軟體包	流行度	大小	工具	功能
coreutils	V:879, I:999	18307	GNU cp	複製本地檔案和目錄 (”-a” 參數實現遞迴)
openssh-client	V:857, I:995	4959	scp	複製遠端檔案和目錄 (客戶端，”-r” 參數實現遞迴)
openssh-server	V:726, I:817	1804	sshd	複製遠端檔案和目錄 (遠端伺服器)
rsync	V:240, I:552	781		單向遠端同步和備份
unison	V:3, I:15	14		雙向遠端同步和備份

Table 10.2: 複製和同步工具列表

在複製檔案的時候，rsync(8) 比其他工具提供了更多的特性。

- 差分傳輸演算法只會傳送原始檔與已存在的目標檔案之間的差異部分
- 快速檢查演算法 (預設) 會查詢大小或者最後的修改時間有變化的檔案
- ”--exclude” 和”--exclude-from” 選項類似於 tar(1)
- 在源目錄中新增反斜槓的語法能夠避免在目標檔案中建立額外的目錄級別。

提示
在表格 10.14 中的版本控制系統 (VCS) 可以被認為是多路拷貝和同步工具。

10.1.3 歸檔語法

以下是用不同的工具壓縮和解壓縮整個”./source” 目錄中的內容。

GNU tar(1):

```
$ tar -cvJf archive.tar.xz ./source
$ tar -xvJf archive.tar.xz
```

或者，如下所示。

```
$ find ./source -xdev -print0 | tar -cvJf archive.tar.xz --null -T -
```

cpio(1):

```
$ find ./source -xdev -print0 | cpio -ov --null > archive.cpio; xz archive.cpio
$ zcat archive.cpio.xz | cpio -i
```

10.1.4 複製語法

如下是用不同的工具複製整個”./source” 目錄中的內容。

- 本地複製: ”./source” 目錄 → ”/dest” 目錄
- 遠端複製：本地主機上的”./source” 目錄 → ”user@host.dom” 主機上的”/dest” 目錄

rsync(8):

```
# cd ./source; rsync -aHAXSv . /dest
# cd ./source; rsync -aHAXSv . user@host.dom:/dest
```

你能夠選擇使用“源目錄上的反斜槓”語法。

```
# rsync -aHAXSv ./source/ /dest
# rsync -aHAXSv ./source/ user@host.dom:/dest
```

或者，如下所示。

```
# cd ./source; find . -print0 | rsync -aHAXSv0 --files-from=- . /dest
# cd ./source; find . -print0 | rsync -aHAXSv0 --files-from=- . user@host.dom:/dest
```

GNU cp(1) 和 openSSH scp(1):

```
# cd ./source; cp -a . /dest
# cd ./source; scp -pr . user@host.dom:/dest
```

GNU tar(1):

```
# (cd ./source && tar cf - . ) | (cd /dest && tar xvfp - )
# (cd ./source && tar cf - . ) | ssh user@host.dom '(cd /dest && tar xvfp - )'
```

cpio(1):

```
# cd ./source; find . -print0 | cpio -pvdm --null --sparse /dest
```

你能夠在所有包含”.” 的例子裡用”foo” 替代”.”，這樣就可以從”./source/foo” 目錄複製檔案到”/dest/foo” 目錄。

在所有包含”.” 的列子裡,你能夠使用絕對路徑”/path/to/source/foo” 來代替”.”,這樣可以去掉”cd ./source;”. 如下所示，這些檔案會根據工具的不同，拷貝到不同的位置。

- ”/dest/foo”: rsync(8), GNU cp(1), 和 scp(1)
- ”/dest/path/to/source/foo”: GNU tar(1), 和 cpio(1)

提示
rsync(8) 和 GNU cp(1) 可以用”-u” 選項來忽略接受端上更新的檔案。

10.1.5 查詢檔案的語法

find(1) 被用作從歸檔中篩選檔案也被用作拷貝指令 (參見節 10.1.3和節 10.1.4) 或者用於 xargs(1) (參見節 9.4.9)。通過 find 的指令列參數能夠使其功能得到加強。

以下是 find(1) 基本語法的總結。

- find 條件參數的運算規則是從左到右。
- 一旦輸出是確定的，那麼運算就會停止。
- “邏輯 **OR**”（由條件之間的”-o” 參數指定的）優先順序低於” 邏輯 **AND**”（由”-a” 參數指定或者條件之間沒有任何引數）。
- ” 邏輯 **NOT**” (由條件前面的”!” 指定) 優先順序高於“邏輯 **AND**”。
- ”-prune” 總是回傳邏輯 **TRUE** 並且如果這個目錄是存在的，將會搜尋除這個目錄以外的檔案。

- ”-name” 選項匹配帶有 shell 萬用字元 (參見節 1.5.6) 的檔名但也匹配帶有類似”*” 和”?” 元字元的 .”。(新的 POSIX 特性)
- ”-regex” 匹配整個檔案路徑，預設採用 emacs 風格的 **BRE** (參見節 1.6.2)。
- ”-size” 根據檔案大小來匹配 (值前面帶有”+” 號匹配更大的檔案，值前面帶有”-” 號匹配更小的檔案)
- ”-newer” 參數匹配比參數名中指定的檔案還要新的檔案。
- ”-print0” 參數總是回傳邏輯 **TRUE** 並將完整檔名 (null terminated) 列印到標準輸出裝置上。

如下是 find(1) 語法格式。

```
# find /path/to \
    -xdev -regextype posix-extended \
    -type f -regex ”.*\.cpio|.*~” -prune -o \
    -type d -regex ”.*/\.git” -prune -o \
    -type f -size +99M -prune -o \
    -type f -newer /path/to/timestamp -print0
```

這些指令會執行如下動作。

1. 查詢”/path/to” 下的所有檔案
2. 限定全域性查詢的檔案系統並且使用的是 **ERE** (參見節 1.6.2)
3. 通過停止處理的方式來排除匹配”.*\.cpio” 或”.*~” 正規表達式的檔案
4. 通過停止處理的方式來排除匹配”.*/\.git” 正規表達式的目錄
5. 通過停止處理的方式來排除比 99MB (1048576 位元組單元) 更大的檔案
6. 顯示檔名，滿足以上搜索條件並且比”/path/to/timestamp” 新的檔案

請留心以上例子中的”-prune -o” 排除檔案的習慣用法。

注
對於非 Debian 系的 Unix-like 系統，有些參數可能不被 find(1) 指令所支援。在這種情況下，應該考慮調整匹配方法並用”-print” 替代”-print0”。你可能同樣需要更改其他相關的指令。

10.1.6 歸檔媒體

為重要的資料存檔尋找 儲存裝置 時，你應該注意它們的侷限性。對於小型的個人資料備份，我使用品牌公司的 CD-R 和 DVD-R 然後把它放在陰涼、乾燥、清潔的地方。(專業的一般使用磁帶存檔介質)

注
防火安全是對於紙質文件來說的，大多數的計算機資料儲存媒介耐熱性比紙差。我經常依賴儲存在多個安全地點的加密拷貝。

網上（主要是來源於供應商資訊）可以檢視儲存介質的最大使用壽命。

- 大於 100 年：用墨水的無酸紙
- 100 年：光碟儲存（CD/DVD，CD/DVD-R）
- 30 年：磁帶儲存（磁帶，軟盤)

- 20 年：相變光碟儲存（CD-RW）

這不包括由於人為導致的機械故障等等。

網上（主要來源於供應商資訊）可以檢視儲存介質的最大的寫次數。

- 大於 250,000 次：硬碟驅動器
- 大於 10,000 次：快閃記憶體
- 1,000 次：CD/DVD-RW
- 1 次：CD/DVD-R，紙

注意
這裡的儲存壽命和寫次數的資料不應該被用來決定任何用於關鍵資料的儲存媒介，請翻閱製造商提供的特定產品的說明。

提示
因為 CD/DVD-R 和紙只能寫一次，它們從根本上阻止了因為重寫導致的資料意外丟失。這是優點！

提示
如果你需要更快更頻繁的進行大資料備份，那麼通過高速網路連線的遠端主機上的硬碟來實現備份，可能是唯一可行的方法。

提示
如果你在使用一個可重複寫入的介質作為你的備份介質，使用支援只讀快照的 btrfs 或 zfs 檔案系統，也許是一個好注意。

10.1.7 可移動儲存裝置

可移動儲存裝置可能是以下的任何一種。

- USB 快閃記憶體盤
- 硬碟驅動器
- 光碟驅動器
- 數碼相機
- 數字音樂播放器

它們可以通過以下的方式來進行連線。

- USB
- IEEE 1394 / FireWire
- PC 卡

像 GNOME 和 KDE 這樣的現代桌面環境能夠在”`/etc/fstab`” 檔案中沒有匹配條目的時候，自動掛載這些可移動裝置。

- udisks2 包提供了守護程序和相關的實用程式來掛載和解除安裝這些裝置。
- D-bus 建立事件來觸發自動處理。
- PolicyKit 提供了所需的特權。

提示
umount(8) 在自動掛載裝置的時候可能會帶有"uhelper=" 參數。

提示
只有當這些可移動裝置沒有在"/etc/fstab" 檔案中列出時，桌面環境下才會自動掛載。

現代桌面環境下的掛載點被選為"/media/*username/disk_label*"，它可以被如下所示的來定製。

- FAT 格式的檔案系統使用 mlabel(1) 指令
- ISO9660 檔案系統使用帶有"-V" 選項的 genisoimage(1) 指令
- ext2/ext3/ext4 檔案系統使用帶有"-L" 選項的 tune2fs(1) 指令

提示
掛載時可能需要提供編碼選項（參見節 8.1.3）。

提示
在圖形介面選單上移除檔案系統，可能會移除它的動態裝置節點例如"/dev/sdc"。如果你想要保留它的裝置節點，你應該在指令列提示字元上輸入 umount(8) 指令來解除安裝它。

10.1.8 選擇用於分享資料的檔案系統

當你通過可移動儲存裝置與其他系統分享資料的時候，你應該先把它格式化為被兩種作業系統都支援的通用的 檔案系統。下面是檔案系統的列表。

提示
檢視節 9.9.1來獲得關於使用裝置級加密的跨平臺的資料共享的資訊。

FAT 檔案系統被絕大多數的現代作業系統支援，它對於通過可行動硬碟進行的資料交換是非常有用的。

當格式化像裝有 FAT 檔案系統的跨平臺資料共享的可移動裝置時，以下應該是保險的選擇。

- 用 fdisk(8)，cfdisk(8) 或者 parted(8) 指令（參見節 9.6.2）把它們格式化為單個的主分割槽並對把它做如下標記。
  - 標記小於 2GB 的 FAT 裝置為字元"6"。
  - 標記更大的 FAT32 裝置為字元"c"。
- 如下所示是用 mkfs.vfat(8) 指令格式化主分割槽的。
  - 它的裝置名字，例如"/dev/sda1" 用於 FAT16 裝置
  - 明確的選項和它的裝置名，例如"-F 32 /dev/sda1" 用於 FAT32 裝置

檔案系統名	典型使用場景
FAT12	軟盤（<32MiB）上跨平臺的資料分享
FAT16	在小硬碟 (<2GiB) 上的跨平臺的資料分享
FAT32	在大硬碟（<8TiB，被 MS Windows95 OSR2 以上的作業系統所支援) 上的跨平臺的資料分享
exFAT	在大硬碟類裝置上跨平臺共享資料（<512TiB，被 WindowsXP, Mac OS X Snow Leopard 10.6.5 和 Linux 核心 5.4 版本以上的作業系統所支援)
NTFS	在大硬碟類裝置上的跨平臺共享資料 (在 MS Windows NT 和後續版本原生支援；在 Linux 上，通過使用 FUSE 的 NTFS-3G 支援。)
ISO9660	在 CD-R 和 DVD+/-R 上的跨平臺的靜態資料分享
UDF	CD-R 和 DVD+/-R (新) 上的增量資料寫入
MINIX	軟盤上磁碟空間高利用率的 unix 檔案資料儲存
ext2	在裝有老舊 linux 系統的硬碟上的資料分享
ext3	在裝有老舊 linux 系統的硬碟上的資料分享
ext4	在裝有較新的 linux 系統的硬碟上的資料分享
btrfs	使用只讀快照在裝有較新的 Linux 系統的硬碟上共享資料

Table 10.3: 典型使用場景下可移動儲存裝置可選擇的檔案系統列表

當使用 FAT 或 ISO9660 檔案系統分享資料時，如下是需要注意的安全事項。

- 用 `tar`(1)，或 `cpio`(1) 指令壓縮檔案，目地是為了保留檔名，符號連結，原始的檔案許可權和檔案所有者資訊。
- 用 `split`(1) 指令把壓縮檔案分解成若干小於 2GiB 的小檔案，使其免受檔案大小限制。
- 加密壓縮檔案保護其內容免受未經授權的存取。

注
因為 FAT 檔案系統的設計，最大的檔案大小為 `(2^32 - 1) bytes = (4GiB -1 byte)`。對於一些老舊的 32 位系統上的應用程式而言，最大的檔案大小甚至更小 `(2^31 -1) bytes = (2GiB -1 byte)`。Debian 沒有遇到後者的問題。

注
微軟系統本身並不建議在超過 200MB 的分割槽或者驅動器上使用 FAT。他們的"Overview of FAT, HPFS, and NTFS File Systems 這篇文章突出顯示了微軟系統的缺點，例如低效的磁碟空間利用。當然了，我們在 Linux 系統上還是應該使用 ext4 檔案系統。

提示
有關檔案系統和存取檔案系統的更多資訊，請參考"Filesystems HOWTO"。

10.1.9 網路上的資料分享

當使用網路來分享資料的時候，你應該使用通用的服務。這裡有一些提示。

儘管對於檔案分享來說，通過網路掛載檔案系統和傳輸檔案是相當方便的，但這可能是不安全的。它們的網路連線必須通過如下所示的加強安全性。

- 用 SSL/TLS 加密
- 建立 SSH 通道
- 建立 VPN 通道
- 網路之間需要有安全的防火牆

參見節 6.5 和節 6.6。

網路服務	典型使用場景描述
SMB/CIFS 用 Samba 掛載網路檔案系統	通過“Microsoft Windows 網路”分享檔案，參見 smb.conf(5) 和官方 Samba 3.x.x 指導和參考手冊 (The Official Samba 3.x.x HOWTO and Reference Guide) 或 samba-doc 軟體包
NFS 用 Linux 核心掛載網路檔案系統	通過“Unix/Linux 網路”分享檔案，參見 exports(5) 和 Linux NFS-HOWTO
HTTP 服務	在 web 伺服器/客戶端之間分享檔案
HTTPS 服務	在有加密的安全套接層 (SSL) 或者安全傳輸層 (TLS) 的網路伺服器/客戶端中分享檔案
FTP 服務	在 FTP 伺服器/客戶端之間分享檔案

Table 10.4: 典型使用場景下可選擇的網路服務列表

10.2 備份和恢復

我們都熟知計算機有時會出問題，或者由於人為的錯誤導致系統和資料損壞。備份和恢復操作是成功的系統管理中非常重要的一部分。可能有一天你的電腦就會出問題。

提示
保持你的備份系統簡潔並且經常備份你的系統，有備份資料比你採用的備份方法的技術先進要重要的多。

10.2.1 備份和恢復策略

有 3 個關鍵的因素決定實際的備份和恢復策略。

1. 知道要備份和恢復什麼。
   - 你自己建立的資料檔案：在”~/” 下的資料
   - 你使用的應用程式建立的資料檔案:在”/var/” 下的資料 (除了”/var/cache/”,”/var/run/” 和”/var/tmp/”)
   - 系統組態檔案：在”/etc/” 下的資料
   - 本地程式：在”/usr/local/” 或”/opt/” 下的資料
   - 系統安裝資訊：關鍵步驟 (分割槽,...) 的純文字備忘錄
   - 驗證資料結果：通過實驗性的恢復操作來預先驗證
     - 使用者程序的 Cron 工作,檔案在”/var/spool/cron/crontabs” 目錄,並且重啟 cron(8)。參見節 9.4.14來獲得關於 cron(8) 和 crontab(1) 的資訊。
     - 使用者程序的 Systemd 計時器工作：檔案在”~/.config/systemd/user” 目錄。參見 systemd.timer(5) 和 systemd.service(5)。
     - 使用者程序的自動啟動工作：檔案在”~/.config/autostart” 目錄。參見 Desktop Application Autostart Specification。
2. 知道怎樣去備份和恢復。
   - 安全的資料儲存：保護其免於覆蓋和系統故障
   - 經常備份：有計劃的備份
   - 冗餘備份：資料映象
   - 傻瓜式操作：單個簡單指令備份
3. 評估涉及的風險和成本。
   - 資料丟失的風險

  - 資料至少是應該在不同的磁碟分割槽上，最好是在不同的磁碟和機器上，來承受檔案系統發生的損壞。重要資料最好儲存在一個只讀檔案系統上。1
- 資料非法訪問的風險
  - 敏感的身份資料，比如”/etc/ssh/ssh_host_*_key”, ”~/.gnupg/*”, ”~/.ssh/*”, ”~/.local/share/keyr ”/etc/passwd”, ”/etc/shadow”, ”popularity-contest.conf”, ”/etc/ppp/pap-secrets”, and ”/etc/e 應當使用加密備份。2 (參見節 9.9。)
  - 即使在信任的系統上，也不能夠硬編碼系統登入密碼或者加密密碼到任何腳本里面。(參見節 10.3.6。)
- 資料丟失的方式及其可能性
  - 硬體（特別是硬碟）將會損壞
  - 檔案系統可能會損壞，裡面的資料可能被丟失
  - 對違規安全訪問而言，遠端儲存系統不能夠被信任
  - 弱的密碼保護能夠被輕鬆的破解
  - 檔案許可權系統可以被破解
- 備份所需的資源：人力，硬體，軟體，…
  - 使用 cron 任務或者 systemd 計時器任務來自動化排程備份工作

提示
你能夠用”debconf-set-selections debconf-selections” 命令恢復 debconf 調配資料，可以用”dpkg --set-selection <dpkg-selections.list” 指令恢復 dpkg 篩選資料。

注
除非你知道自己做的是什麼，否則不要備份 /proc, /sys, /tmp, 和 /run 目錄下的偽檔案系統（參見節 1.2.12 和節 1.2.13)。它們是龐大且無用的資料。

注
當備份資料的時候，你可能希望停止一些應用程式的背景程式例如 MTA（參見節 6.2.4）。

10.2.2 實用備份套件

以下是 Debian 系統上值得注意的實用備份程式套件的列表。

備份工具有各自的專用的用途。

- Mondo Rescue 是一個備份系統，它能夠方便的從備份 CD/DVD 等裝置中快速恢復整個系統，而不需要經過常規的系統安裝過程。
- Bacula，Amanda 和 BackupPC 是全功能的備份實用套件，主要用於聯網的定期備份。
- Duplicity 和 Borg 是簡單的備份工具用於典型的工作站。

10.2.3 備份技巧

對於一個個人工作站，為伺服器環境設計的全功能備份套件工具也行不是最合適的。與此同時，已有的用於工作站的備份工具有一些不足。

這裡有一些技巧讓備份更加容易，只需使用者做最小的工作。這些技巧可以同任意備份工具一起使用。

出於演示的目的，讓我們假設基本使用者和組名為 penguin，建立一個備份和快照指令碼例子”/usr/local/bin/bkss.sh

1一個只能寫一次的媒介，例如 CD/DVD-R，能防止覆蓋事故。（參見節 9.8 怎樣在 shell 命令列寫入儲存媒介。GNOME 桌面圖形環境可以讓你輕鬆的透過選單：“位置 → CD/DVD 燒錄” 來實現寫入操作。）

2這些資料中的一些，不能夠透過在系統裡面輸入同樣的字串來重新生成。

軟體包	流行度	大小	說明
bacula-common	V:9, I:10	2119	Bacula: 網路資料備份，恢復和核查-常見的支援檔案
bacula-client	V:0, I:2	154	Bacula: 網路資料備份，恢復和核查-客戶端元軟體包
bacula-console	V:0, I:3	104	Bacula: 網路資料備份，恢復和核查-文字終端
bacula-server	I:0	154	Bacula: 網路資料備份，恢復和核查-伺服器端元軟體包
amanda-common	V:0, I:2	9897	Amanda: 馬里蘭大學開發的高階自動化網路磁碟歸檔器（庫）
amanda-client	V:0, I:2	1092	Amanda: 馬里蘭大學開發的高階自動化網路磁碟歸檔器（客戶端）
amanda-server	V:0, I:0	1077	Amanda: 馬里蘭大學開發的高階自動化網路磁碟歸檔器（伺服器端）
backuppc	V:2, I:2	3178	BackupPC 是用於備份 PC 機資料（基於磁碟）的高效能的企業級工具
duplicity	V:28, I:48	1973	(遠端) 增量備份
deja-dup	V:26, I:41	4992	duplicity 的 GUI（圖形使用者介面）前端
borgbackup	V:11, I:20	3301	(遠端) 去重備份
borgmatic	V:2, I:3	509	borgbackup 備份軟體的輔助軟體
rdiff-backup	V:4, I:10	1203	(遠端) 增量備份
restic	V:2, I:6	21373	(遠端) 增量備份
backupninja	V:2, I:3	360	輕量，可擴展的 **meta-backup** 系統
flexbackup	V:0, I:0	243	(遠端) 增量備份
slbackup	V:0, I:0	151	(遠端) 增量備份
backup-manager	V:0, I:1	566	指令列備份工具
backup2l	V:0, I:0	115	用於可掛載媒介 (基於磁碟的) 的低維護的備份/恢復工具

Table 10.5: 實用備份程式套件列表

```
#!/bin/sh -e
SRC="$1" # source data path
DSTFS="$2" # backup destination filesystem path
DSTSV="$3" # backup destination subvolume name
DSTSS="${DSTFS}/${DSTSV}-snapshot" # snapshot destination path
if [ "$(stat -f -c %T "$DSTFS")" != "btrfs" ]; then
  echo "E: $DESTFS needs to be formatted to btrfs" >&2 ; exit 1
fi
MSGID=$(notify-send -p "bkup.sh $DSTSV" "in progress ...")
if [ ! -d "$DSTFS/$DSTSV" ]; then
  btrfs subvolume create "$DSTFS/$DSTSV"
  mkdir -p "$DSTSS"
fi
rsync -aHxS --delete --mkpath "${SRC}/" "${DSTFS}/${DSTSV}"
btrfs subvolume snapshot -r "${DSTFS}/${DSTSV}" ${DSTSS}/$(date -u --iso=min)
notify-send -r "$MSGID" "bkup.sh $DSTSV" "finished!"
```

這裡，只使用基本工具 rsync(1) 來幫助備份，儲存空間使用 Btrfs 來高效利用。

> 提示
> 提示：這個作者在他的工作站上使用他自己的類似 shell 指令碼"bss: Btrfs Subvolume Snapshot Utility"。

10.2.3.1 GUI（圖形使用者介面）備份

這裡是一個建立單擊 GUI（圖形使用者介面）圖示備份的例子。

- 準備一個 USB 儲存裝置用來備份。

  - 格式化 USB 儲存裝置為一個分割槽，使用 btrfs 檔案系統，卷標名為”BKUP”。這個 U 盤也能夠加密 (參見節 9.9.1)。
  - 把這個插入你的系統。桌面系統將自動掛載它到”/media/penguin/BKUP”。
  - 執行”sudo chown penguin:penguin /media/penguin/BKUP”，讓它可以由使用者寫。
- 建立如下的”~/.local/share/applications/BKUP.desktop” 檔案，按照節 9.4.10 裡寫的技術：

```
[Desktop Entry]
Name=bkss
Comment=Backup and snapshot of ~/Documents
Exec=/usr/local/bin/bkss.sh /home/penguin/Documents /media/penguin/BKUP Documents
Type=Application
```

對於每一次圖示單擊，你的資料從”~/Documents” 備份到 USB 儲存裝置，並建立了一個只讀快照。

10.2.3.2 掛載事件觸發的備份

這裡是一個由掛載事件觸發的自動備份例子。

- 按節 10.2.3.1 的方式準備一個 USB 儲存裝置用於備份。
- 建立一個如下的 systemd 服務單元檔案”~/.config/systemd/user/back-BKUP.service”：

```
[Unit]
Description=USB Disk backup
Requires=media-%u-BKUP.mount
After=media-%u-BKUP.mount

[Service]
ExecStart=/usr/local/bin/bkss.sh %h/Documents /media/%u/BKUP Documents
StandardOutput=append:%h/.cache/systemd-snap.log
StandardError=append:%h/.cache/systemd-snap.log

[Install]
WantedBy=media-%u-BKUP.mount
```

- 用下面的方式啟用這個 systemd 單元配置：

```
 $ systemctl --user enable bkup-BKUP.service
```

對於每一次掛載事件，你的資料從”~/Documents” 備份到 USB 儲存裝置，並建立了一個只讀快照。

這裡，當前 systemd 在記憶體中的 systemd 掛載單元的名字，使用者使用”systemctl --user list-units --type=moun
來呼叫服務管理器來查詢。

10.2.3.3 時間事件觸發的備份

這裡是一個由時間事件觸發的自動備份例子。

- 按節 10.2.3.1 的方式準備一個 USB 儲存裝置用於備份。
- 建立一個如下的 systemd 時間單元檔案”~/.config/systemd/user/snap-Documents.timer”：

```
[Unit]
Description=Run btrfs subvolume snapshot on timer
Documentation=man:btrfs(1)

[Timer]
OnStartupSec=30
```

```
OnUnitInactiveSec=900

[Install]
WantedBy=timers.target
```

- 建立一個如下的 systemd 服務單元檔案"~/.config/systemd/user/snap-Documents.service":

```
[Unit]
Description=Run btrfs subvolume snapshot
Documentation=man:btrfs(1)

[Service]
Type=oneshot
Nice=15
ExecStart=/usr/local/bin/bkss.sh %h/Documents /media/%u/BKUP Documents
IOSchedulingClass=idle
CPUSchedulingPolicy=idle
StandardOutput=append:%h/.cache/systemd-snap.log
StandardError=append:%h/.cache/systemd-snap.log
```

- 用下面的方式啟用這個 systemd 單元配置:

```
$ systemctl --user enable snap-Documents.timer
```

對於每一次時間事件，你的資料從"~/Documents" 備份到 USB 儲存裝置，並建立了一個只讀快照。

這裡,當前 systemd 在記憶體中的 systemd 使用者時間單元的名字,使用"systemctl --user list-units --type=time 來呼叫服務管理器來查詢。

對於現在的桌面系統，這個 systemd 方案，比起傳統的 Unix at(1)、cron(8) 或 anacron 方式，能夠提供更精緻的細粒度控制。

10.3 資料安全基礎

資料安全基礎設施是資料加密，訊息摘要和簽名工具的結合。

軟體包	流行度	大小	指令	說明
gnupg	V:553, I:909	885	gpg(1)	GNU 隱私衛士 - OpenPGP 加密和簽名工具
gpgv	V:893, I:999	922	gpgv(1)	GNU 隱私衛士 - 簽名驗證工具
paperkey	V:1, I:13	58	paperkey(1)	從 OpenPGP 私鑰裡面，僅僅匯出私密資訊
cryptsetup	V:34, I:79	410	cryptsetup(8), ...	dm-crypt 塊裝置加密支援 LUKS 工具
coreutils	V:879, I:999	18307	md5sum(1)	計算與校驗 MD5 訊息摘要
coreutils	V:879, I:999	18307	sha1sum(1)	計算與校驗 SHA1 訊息摘要
openssl	V:839, I:995	2294	openssl(1ssl)	使用"openssl dgst" (OpenSSL) 計算資訊摘要
libsecret-tools	V:0, I:10	41	secret-tool(1)	儲存和取回密碼 (CLI)
seahorse	V:78, I:267	7987	seahorse(1)	金鑰管理工具 (GNOME)

Table 10.6: 資料安全基礎工具列表

參見節 9.9 的 dm-crypt 和 fscrypt，它們透過 Linux 核心模組實現了自動資料加密架構。

指令	說明
gpg --gen-key	生成一副新的金鑰對
gpg --gen-revoke my_user_ID	生成 my_user_ID 的一份吊銷證書
gpg --edit-key user_ID	互動式的編輯金鑰，輸入”help” 來獲得幫助資訊
gpg -o file --export	把所有的金鑰輸出到檔案
gpg --import file	從檔案匯入金鑰
gpg --send-keys user_ID	傳送 user_ID 的公鑰到公鑰伺服器
gpg --recv-keys user_ID	從公鑰伺服器下載 user_ID 的公鑰
gpg --list-keys user_ID	列出 user_ID 的所有金鑰
gpg --list-sigs user_ID	列出 user_ID 的簽字
gpg --check-sigs user_ID	檢查 user_ID 金鑰簽字
gpg --fingerprint user_ID	檢查 user_ID 的指紋
gpg --refresh-keys	更新本地金鑰

Table 10.7: GNU 隱私衛士金鑰管理指令的列表

程式碼	信任描述
-	沒有所有者信任簽名/沒有計算
e	信任計算失敗
q	沒有足夠的資訊用於計算
n	從不信任這個鍵
m	最低限度的信任
f	完全信任
u	最終信任

Table 10.8: 信任碼含義列表

10.3.1 GnuPG 金鑰管理

如下是 GNU 隱私衛士 基本的金鑰管理命令。

信任碼含義.

如下指令上傳我的”1DD8D791” 公鑰到主流的公鑰伺服器”hkp://keys.gnupg.net”。

```
$ gpg --keyserver hkp://keys.gnupg.net --send-keys 1DD8D791
```

預設良好的公鑰伺服器在”~/.gnupg/gpg.conf” （舊的位置在”~/.gnupg/options”）檔案中設定，此檔案包含了以下資訊。

```
keyserver hkp://keys.gnupg.net
```

從鑰匙伺服器獲得無名鑰匙。

```
$ gpg --list-sigs --with-colons | grep '^sig.*\[User ID not found\]' |\
          cut -d ':' -f 5| sort | uniq | xargs gpg --recv-keys
```

有一個錯誤在 OpenPGP 公鑰伺服器 (先前的版本 0.9.6)，會將鍵中斷為 2 個以上的子鍵。新的 gnupg (>1.2.1-2) 軟體包能夠處理這些中斷的子鍵。參見 gpg(1) 下的”--repair-pks-subkey-bug” 選項.

10.3.2 在檔案上使用 GnuPG

這裡有一些在檔案上使用 GNU 隱私衛士 指令的例子。

指令	說明
`gpg -a -s file`	ASCII 封裝的簽名檔案 file.asc
`gpg --armor --sign file`	同上
`gpg --clearsign file`	生成明文簽字資訊
`gpg --clearsign file|mail foo@example.org`	傳送一份明文簽字到 `foo@example.org`
`gpg --clearsign --not-dash-escaped patchfile`	明文簽名的補丁檔案
`gpg --verify file`	驗證明文檔案
`gpg -o file.sig -b file`	生成一份分離的簽字
`gpg -o file.sig --detach-sign file`	同上
`gpg --verify file.sig file`	使用 file.sig 驗證檔案
`gpg -o crypt_file.gpg -r name -e file`	公鑰加密，從檔案裡面獲得名字，生成二進位制的 crypt_file.gpg
`gpg -o crypt_file.gpg --recipient name --encrypt file`	同上
`gpg -o crypt_file.asc -a -r name -e file`	公鑰加密，從檔案中獲得名字，生成 ASCII 封裝的 crypt_file.asc
`gpg -o crypt_file.gpg -c file`	將檔案對稱加密到 crypt_file.gpg
`gpg -o crypt_file.gpg --symmetric file`	同上
`gpg -o crypt_file.asc -a -c file`	對稱加密，從檔案到 ASCII 封裝的 crypt_file.asc
`gpg -o file -d crypt_file.gpg -r name`	解密
`gpg -o file --decrypt crypt_file.gpg`	同上

Table 10.9: 在檔案上使用的 GNU 隱私衛士的指令列表

10.3.3 在 Mutt 中使用 GnuPG

增加下面內容到”~/.muttrc”，在自動啟動時，避免一個慢的 GnuPG，在索引選單中按”S” 來允許它使用。

```
macro index S ”:toggle pgp_verify_sig\n”
set pgp_verify_sig=no
```

10.3.4 在 vim 中使用 GnuPG

gnupg 外掛可以讓你對副檔名為”.gpg”, ”.asc”, 和”.pgp” 的檔案可靠的執行 GnuPG。[3]

```
$ sudo aptitude install vim-scripts
$ echo ”packadd! gnupg” >> ~/.vim/vimrc
```

10.3.5 MD5 校驗和

md5sum(1) 提供了製作摘要檔案的一個工具, 它使用 rfc1321 裡的方式製作摘要檔案.

```
$ md5sum foo bar >baz.md5
$ cat baz.md5
d3b07384d113edec49eaa6238ad5ff00  foo
c157a79031e1c40f85931829bc5fc552  bar
$ md5sum -c baz.md5
foo: OK
bar: OK
```

注
MD5 校驗和的 CPU 計算強度是比 GNU Privacy Guard (GnuPG) 加密簽名要少的. 在通常情況下, 只有頂級的摘要檔案才需要加密簽名來確保資料完整性.

10.3.6 密碼金鑰環

在 GNOME 系統,GUI (圖形使用者介面) 工具 seahorse(1) 管理密碼,安全的在金鑰環 ~/.local/share/keyrings/* 裡面儲存它們。

secret-tool(1) 能夠從命令列儲存密碼到鑰匙環。

讓我們儲存 LUKS/dm-crypt 加密磁碟映象用到的密碼

```
$ secret-tool store --label=’LUKS passphrase for disk.img’ LUKS my_disk.img
Password: ********
```

這個儲存的密碼能夠被獲取並給到其它程式，比如 cryptsetup(8)。

```
$ secret-tool lookup LUKS my_disk.img | \
  cryptsetup open disk.img disk_img --type luks --keyring -
$ sudo mount /dev/mapper/disk_img /mnt
```

提示
無論何時，你需要在一個腳本里面提供密碼時，使用 secret-tool 來避免將密碼直接硬編碼到腳本里面。

---

[3]如果你使用”~/.vimrc” 代替”~/.vim/vimrc”，請進行相應的取代。

10.4 原始碼合併工具

這裡有許多原始碼合併工具。如下的是我感興趣的工具。

軟體包	流行度	大小	指令	說明
patch	V:99, I:699	248	patch(1)	給原檔案打補丁
vim	V:91, I:370	3743	vimdiff(1)	在 vim 中並排比較兩個檔案
imediff	V:0, I:0	169	imediff(1)	全屏互動式兩路/三路合併工具
meld	V:8, I:30	3500	meld(1)	比較和移植檔案 (GTK)
wiggle	V:0, I:0	176	wiggle(1)	應用被拒絕的補丁
diffutils	V:862, I:996	1735	diff(1)	逐行比較兩個檔案
diffutils	V:862, I:996	1735	diff3(1)	逐行比較和合並三個檔案
quilt	V:2, I:22	871	quilt(1)	管理系列補丁
wdiff	V:7, I:51	648	wdiff(1)	在文字檔案中，顯示單詞的不同
diffstat	V:13, I:120	74	diffstat(1)	通過 diff 生成一個改變柱狀圖
patchutils	V:16, I:118	232	combinediff(1)	從兩個增量補丁建立一個積累補丁
patchutils	V:16, I:118	232	dehtmldiff(1)	從一個 HTML 頁面提取出一個 diff
patchutils	V:16, I:118	232	filterdiff(1)	從一個 diff 檔案裡面提取或者排除 diff 檔案
patchutils	V:16, I:118	232	fixcvsdiff(1)	修復由 CVS patch(1) 錯誤建立的 diff 檔案
patchutils	V:16, I:118	232	flipdiff(1)	交換兩個補丁的順序
patchutils	V:16, I:118	232	grepdiff(1)	顯示哪些檔案是由匹配正規表達式的補丁修改
patchutils	V:16, I:118	232	interdiff(1)	顯示在兩個統一格式 diff 檔案（基於同一個檔案的兩個不同 diff 檔案）之間的差異
patchutils	V:16, I:118	232	lsdiff(1)	顯示哪些檔案由補丁修改
patchutils	V:16, I:118	232	recountdiff(1)	重新計算通用內容 diff 檔案的數量和偏移
patchutils	V:16, I:118	232	rediff(1)	修復手工編輯 diff 檔案的數量和偏移
patchutils	V:16, I:118	232	splitdiff(1)	隔離出增量補丁
patchutils	V:16, I:118	232	unwrapdiff(1)	識別已經被分詞的補丁
dirdiff	V:0, I:1	167	dirdiff(1)	顯示目錄樹之間的不同並移植改變
docdiff	V:0, I:0	553	docdiff(1)	逐詞逐字的比較兩個檔案
makepatch	V:0, I:0	100	makepatch(1)	生成擴展補丁檔
makepatch	V:0, I:0	100	applypatch(1)	套用擴展補丁檔

Table 10.10: 原始碼合併工具列表

10.4.1 從原始碼檔案匯出差異

下面的操作，匯出兩個原始檔的不同，並根據檔案的位置，建立通用 diff 檔案"file.patch0" 或"file.patch1".

```
$ diff -u file.old file.new > file.patch0
$ diff -u old/file new/file > file.patch1
```

10.4.2 原始碼檔案移植更新

diff 檔案（通常被叫作 patch 補丁檔案），用於傳送一個程式更新。通過下面的方式，接收到的部分，應用這個更新到其它檔案。

```
$ patch -p0 file < file.patch0
$ patch -p1 file < file.patch1
```

10.4.3 互動式移植

如果一個原始碼，你有兩個版本，你可以透過下面的方式，使用 imediff(1) 執行兩方互動式移植。

```
$ imediff -o file.merged file.old file.new
```

如果一個原始碼，你有三個版本，你可以透過下面的方式，使用 imediff(1) 執行互動式三方移植。

```
$ imediff -o file.merged file.yours file.base file.theirs
```

10.5 Git

Git 是這些天選擇的用於 版本控制系統 version control system (VCS) 的工具，因為 Git 能夠同時在本地和遠端原始碼管理上，做任何事情。

通過 Debian Salsa service，Debian 能夠提供免費的 Git 服務。在 http://wiki.debian.org/Salsa 能找到它的說明文件。

下面是一些 Git 相關軟體包。

軟體包	流行度	大小	指令	說明
git	V:347, I:545	46734	git(7)	Git 快速、可擴展、分佈式的版本控制系統
gitk	V:5, I:33	1838	gitk(1)	有歷史功能的 Git 圖形倉庫瀏覽器
git-gui	V:1, I:18	2429	git-gui(1)	Git 圖形界面（無歷史功能）
git-email	V:0, I:10	1087	git-send-email(1)	從 Git 用電子郵件發送收集到的補丁
git-buildpackage	V:1, I:9	1988	git-buildpackage(1)	用 Git 自動製作 Debian 包
dgit	V:0, I:1	484	dgit(1)	Debian 檔案庫的 git 互動操作
imediff	V:0, I:0	169	git-ime(1)	互動式的分開 git 提交的輔助工具
stgit	V:0, I:0	601	stg(1)	封裝的 git (Python)
git-doc	I:12	13208	N/A	Git 官方文檔
gitmagic	I:0	721	N/A	”Git 魔術”，易於理解的 Git 手冊

Table 10.11: git 相關包和指令列表

10.5.1 調配 Git 客戶端

你可以在”~/.gitconfig” 裏面設置幾個 Git 接下來需要使用的全局調配，比如說你的名字和電子郵件地址。

```
$ git config --global user.name "Name Surname"
$ git config --global user.email yourname@example.com
```

你也可以按如下所示定製 Git 的預設行為。

```
$ git config --global init.defaultBranch main
$ git config --global pull.rebase true
$ git config --global push.default current
```

如果你習慣使用 CVS 或 Subversion 指令，你也許希望設置如下幾個指令別名。

```
$ git config --global alias.ci "commit -a"
$ git config --global alias.co checkout
```

你能夠通過如下方式檢查你的整體組態。

```
$ git config --global --list
```

10.5.2 基本的 Git 命令

Git 操作涉及幾個資料。

- 工作樹目錄保持面向使用者的檔案，你可以對這些檔案做修改。
  - 需要被記錄的改變，必須明確的被選擇並暫存到索引。這是 `git add` 和 `git rm` 命令。
- 索引保持暫存檔案。
  - 在接下來的請求之前，暫存檔案將被提交到本地倉庫。這個是 `git commit` 命令。
- 本地倉庫保持已經提交的的檔案。
  - Git 記錄提交資料的連結歷史並在倉庫裡面將它們作為分支組織。
  - 本地倉庫透過 `git push` 命令傳送資料到遠端倉庫。
  - 本地倉庫能夠透過 `git fetch` 和 `git pull` 命令從遠端倉庫接收資料。
    * `git pull` 命令在 `git fetch` 後執行 `git merge` 或 `git rebase` 命令。
    * 這裡，`git merge` 聯合兩個獨立分支的歷史結尾到一個點。(在沒有定製的 `git pull`，這個是預設的，同時對上游作者釋出分支到許多人時，也是好的)
    * 這裡，`git rebase` 建立一個遠端分支的序列歷史的單個分支，跟著本地分支。(這是定製 `pull.rebase true` 的情況，對我們其餘的用途有用。)
- 遠端倉庫保持已經提交的檔案。
  - 到遠端倉庫的通訊，使用安全的通訊協議，比如 SSH 或 HTTPS。

工作樹是在 `.git/` 目錄之外的檔案。在 `.git/` 目錄裡面的檔案，包括索引、本地倉庫資料和一些 git 配置的文字檔案。

這裡是主要的 Git 命令概覽。

Git 命令	功能
`git init`	建立 (本地) 儲存庫
`git clone URL`	克隆遠端儲存庫到本地倉庫工作目錄樹
`git pull origin main`	透過遠端倉庫 `origin` 更新本地 `main` 分支
`git add .`	增加工作樹裡面的檔案僅作為預先存在的索引檔案
`git add -A .`	增加工作樹裡面的所有檔案到索引 (包括已經刪除的)
`git rm filename`	從工作樹和索引中刪除檔案
`git commit`	提交在索引中的暫存改變到本地儲存庫
`git commit -a`	新增工作樹裡的所有的改變到索引並提交它們到本地倉庫 (新增 + 提交)
`git push -u origin branch_name`	使用本地 `branch_name` 分支更新遠端倉庫 `origin` (初始啟用)
`git push origin branch_name`	使用本地 `branch_name` 分支更新遠端倉庫 `origin` (隨後呼叫)
`git diff treeish1 treeish2`	顯示 *treeish1* 提交和 *treeish2* 提交的不同
`gitk`	VCS 儲存庫分支歷史樹的圖形介面顯示

Table 10.12: 主要的 Git 命令

10.5.3 Git 技巧

下面是一些 Git 技巧。

警告
不要使用帶空格的標簽字串。即使一些工具，如 `gitk`(1) 允許你使用它，但會阻礙其它 `git` 指令。

Git 命令列	功能
`gitk --all`	參看完整的 Git 歷史和操作，比如重置 HEAD 到另外一個提交、挑選補丁、建立標籤和分支……
`git stash`	得到一個乾淨的工作樹，不會丟失資料
`git remote -v`	檢查遠端設定
`git branch -vv`	檢查分支設定
`git status`	顯示工作樹狀態
`git config -l`	列出 git 設定
`git reset --hard HEAD; git clean -x -d -f`	反轉所有工作樹的改變並完全清理它們
`git rm --cached filename`	反轉由 `git add filename` 改變的暫存索引
`git reflog`	獲取參考日誌（對從刪除的分支中恢復提交有用）
`git branch new_branch_name HEAD@{6}`	從 reflog 資訊建立一個新的分支
`git remote add new_remote URL`	增加一個由 URL 指向的遠端倉庫 `new_remote`
`git remote rename origin upstream`	遠端倉庫的名字從 `origin` 重新命名到 `upstream`
`git branch -u upstream/branch_name`	設定遠端跟蹤到遠端倉庫 `upstream` 和它的分支名 `branch_name`。
`git remote set-url origin https://foo/bar.git`	改變 `origin` 的 URL
`git remote set-url --push upstream DISABLED`	禁止推送到 `upstream`（編輯 `.git/config` 來重新啟用）
`git remote update upstream`	獲取 `upstream` 倉庫中所有遠端分支更新
`git fetch upstream foo:upstream-foo`	建立本地（可能是孤立的）`upstream-foo` 分支，作為 `upstream` 倉庫中 `foo` 分支的一個複製
`git checkout -b topic_branch ; git push -u topic_branch origin`	製作一個新的 `topic_branch` 並把它推送到 `origin`
`git branch -m oldname newname`	本地分支改名
`git push -d origin branch_to_be_removed`	刪除遠端分支（新的方式）
`git push origin :branch_to_be_removed`	刪除遠端分支（老的方式）
`git checkout --orphan unconnected`	建立一個新的 `unconnected` 分支
`git rebase -i origin/main`	從 `origin/main` 重新排序、刪除、壓縮提交到一個乾淨的分支歷史
`git reset HEAD^; git commit --amend`	壓縮最後兩個提交為一個
`git checkout topic_branch ; git merge --squash topic_branch`	壓縮整個 `topic_branch` 到一個提交
`git fetch --unshallow --update-head-ok origin '+refs/heads/*:refs/heads/*'`	反轉一個淺克隆到一個所有分支的完整克隆
`git ime`	分開最後的提交到一系列單個逐一檔案的小提交。（要求 `imediff`）
`git repack -a -d; git prune`	本地倉庫重新打包到一個單獨的包中（這可能限制從刪除分支裡面恢復丟失資料等機會）

Table 10.13: Git 技巧

注意
如果一個本地分支推送到一個已經變基或者壓縮過的倉庫，推送這樣的分支有風險，並要求 `--force` 選項。這通常對 `main` 分支來說不可接受，但對於一個移植到 `main` 分支前的特定分支，是可以接受的。

注意
從指令列通過"`git-xyz`" 直接呼叫 `git` 子指令的方式，從 2006 年早期開始就被取消。

提示
如果有一個可執行檔案 `git-foo` 在路徑環境變數 `$PATH` 裡面, 在命令列輸入沒有中劃線的"`git foo`"，則將呼叫 `git-foo`. 這是 `git` 命令的一個特性。

10.5.4 Git 參考

看下面。

- man 手冊：git(1) (`/usr/share/doc/git-doc/git.html`)
- Git 使用者手冊 (`/usr/share/doc/git-doc/user-manual.html`)
- git 介紹教學 (`/usr/share/doc/git-doc/gittutorial.html`)
- git 介紹教學：第二部 (`/usr/share/doc/git-doc/gittutorial-2.html`)
- GIT 每一天 20 個左右的命令 (`/usr/share/doc/git-doc/giteveryday.html`)
- Git 魔術 (`/usr/share/doc/gitmagic/html/index.html`)

10.5.5 其它的版本控制系統

版本控制系統（VCS）有時被認為是修訂控制系統 (RCS), 或者是軟體配置管理程式 (SCM)。

這裡是 Debian 系統上著名的其它非 Git 的 VCS 彙總。

軟體包	流行度	大小	工具	VCS 型別	描述
mercurial	V:5, I:33	2019	Mercurial	分散式	mercurial 主要是用 Python 寫的還有一部分是 C 寫的
darcs	V:0, I:5	34070	Darcs	分散式	有智慧代數補丁的 DVCS (慢)
bzr	I:8	28	GNU Bazaar	分散式	受 tla 啟發並且是用 Python 寫的 DVCS (歷史)
tla	V:0, I:1	1022	GNU arch	分散式	主要由 Tom Lord 寫的 DVCS (成為歷史的)
subversion	V:12, I:74	4837	Subversion	遠端	”比 CVS 做的好“，遠端 VCS 的新標準 (歷史)
cvs	V:4, I:30	4753	CVS	遠端	以前的遠端 VCS 標準 (歷史)
tkcvs	V:0, I:1	1498	CVS, ⋯	遠端	VCS (CVS,Subversion,RCS) 儲存庫樹的圖形介面顯示
rcs	V:2, I:13	564	RCS	本地	” 比 Unix SCCS 做的好” (歷史)
cssc	V:0, I:1	2044	CSSC	本地	Unix SCCS 的克隆 (歷史)

Table 10.14: 其它版本控制系統工具列表

Chapter 11

資料轉換

下面是關於 Debian 系統上可用的格式轉化工具及其相關提示的資訊。
基於標準的工具，是非常好用的，但支援的專有資料格式有限.

11.1 文字資料轉換工具

如下是文字資料轉換工具。

軟體包	流行度	大小	關鍵詞	說明
libc6	V:923, I:999	12987	字元集	使用 iconv(1) 的不同語言環境 (locale) 之間的文字編碼轉換器 (基礎的)
recode	V:2, I:18	601	字元集 + 換行	不同語言環境 (locale) 之間的文字編碼轉換器 (多功能的，更多別名和特性)
konwert	V:1, I:48	134	字元集	不同語言環境 (locale) 之間的文字編碼轉換器 (高檔的)
nkf	V:0, I:9	360	字元集	日語字元集翻譯
tcs	V:0, I:0	518	字元集	字元集翻譯
unaccent	V:0, I:0	35	字元集	代替重音字元，使用和它們相當的非重音字元
tofrodos	V:1, I:18	51	換行	在 DOS 和 Unix 之間的文字格式轉換: fromdos(1) 和 todos(1)
macutils	V:0, I:0	312	換行	在 Macintosh 和 Unix 之間的文字格式轉換: frommac(1) 和 tomac(1)

Table 11.1: 文字資料轉化工具列表

11.1.1 用 iconv 指令來轉換文字檔案

> 提示
> iconv(1) 是 libc6 軟體包的一部分並且它可以在類 Unix 的系統上轉換字元的編碼。

你能夠通過如下的指令用 iconv(1) 來轉換文字檔案的編碼。

```
$ iconv -f encoding1 -t encoding2 input.txt >output.txt
```

編碼值是大小寫不敏感的，且會在匹配時忽略“-”和“_”。可以使用“iconv -l”指令檢查支援的編碼。

編碼值	用法
ASCII	美國資訊交換標準程式碼，7 位程式碼不帶重音符號
UTF-8	用於所有現代作業系統的多語言標準
ISO-8859-1	舊的西歐語言標準，ASCII + 重音符號
ISO-8859-2	舊的東歐語言標準，ASCII + 重音符號
ISO-8859-15	舊的帶有歐元符號的西歐語言標準 (ISO-8859-1)
CP850	code page 850，用於西歐語言的微軟 DOS 的帶有圖形的字元，ISO-8859-1 的變體
CP932	code page 932, 日語 Microsoft Windows 的 Shift-JIS 變體
CP936	code page 936，用於簡體中文的微軟作業系統風格的 GB2312，GBK 或者 GB18030 的變體
CP949	code page 949，用於韓語的微軟作業系統風格的 EUC-KR 或者 Unified Hangul Code 的變體
CP950	code page 950，用於繁體中文的微軟作業系統風格的 Big5 的變體
CP1251	code page 1251，用於西裡爾字母的微軟作業系統風格的編碼
CP1252	code page 1252，用於西歐語言的微軟作業系統風格的 ISO-8859-15 的變體
KOI8-R	用於西裡爾字母的舊俄語 UNIX 標準
ISO-2022-JP	日文郵件的標準編碼，只使用 7 位位元組
eucJP	老的日文 UNIX 標準的 8 位位元組，和 Shift-JIS 完全不同
Shift-JIS	日文 JIS X 0208 附錄 1 標準 (參見 CP932)

Table 11.2: 編碼值和用法的列表

注
一些編碼只支援資料轉換，它不能作為語言環境的值 (節 8.1)。

像 ASCII 和 ISO-8859 這樣適用於單位元組的字元集，字元編碼和字元集幾乎指的是同一件事情。

對於多字元的字元集，比如說，用於日文的 JIS X 0213，或用於差不多所有語言的 Universal Character Set (UCS, Unicode, ISO-10646-1) , 有多種編碼方案來序列化它們的位元組資料。

- 日文的 EUC 和 ISO/IEC 2022 (也被稱為 JIS X 0202)
- Unicode 的 UTF-8、UTF-16/UCS-2 和 UTF-32/UCS-4 編碼

對於以上這些，字元集和字元編碼之間有著明顯的區別。

對某些計算機廠家而言，code page 是作為字元編碼表的同義詞來使用。

注
請注意，大部分編碼系統共享 ASCII 的 7 位字元的同樣編碼，但也有一些列外。如果你從通常所說的 shift-JIS 編碼格式，轉化老的日文 C 語言程式和 URL 資料，到 UTF-8 格式, 你需要使用"CP932" 作為編碼名來代替"shift-JIS" 來得到期望的結果：0x5C → "\" 和 0x7E → "~". 否則，這些將被轉化為錯誤的字元。

提示
recode(1) 也可能被使用並且不僅僅是 iconv(1)，fromdos(1)，todos(1)，frommac(1) 和 tomac(1) 功能的結合。想要獲得更多資訊，請參見"info recode"。

11.1.2 用 iconv 檢查檔案是不是 UTF-8 編碼

你能夠通過如下指令用 iconv(1) 來檢查一個文字檔案是不是用 UTF-8 編碼的。

```
$ iconv -f utf8 -t utf8 input.txt >/dev/null || echo ”non-UTF-8 found”
```

提示
在上面的例子中使用”--verbose” 參數來找到第一個 non-UTF-8 字元。

11.1.3 使用 iconv 轉換檔名

這裡是一個示例腳步，在同一目錄下，將在老的作業系統系統下建立的檔名編碼，轉換為現代 UTF-8.

```
#!/bin/sh
ENCDN=iso-8859-1
for x in *;
 do
 mv ”$x” ”$(echo ”$x” | iconv -f $ENCDN -t utf-8)”
done
```

”$ENCDN” 變數定義了在老的作業系統下，檔名使用的原始編碼，見表格 11.2.

對於更加複雜的情況，請使用適當的編碼作為 mount(8) 的選項 (參見節 8.1.3) 來掛載包含有這樣檔名的檔案系統 (比如說，磁碟上的一個分割槽)，使用”cp -a” 指令來拷貝它的整個內容到另外一個使用 UTF-8 掛載的檔案系統上。

11.1.4 換行符轉換

文字檔案的格式，特別是行尾 (換行符) 編碼，有平臺獨立性。

平臺	換行符編碼	控制碼	十進位制	**16** 進位制
Debian (unix)	LF	^J	10	0A
MSDOS 和 Windows	CR-LF	^M^J	13 10	0D 0A
蘋果的 Macintosh	CR	^M	13	0D

Table 11.3: 不同平臺的換行符樣式列表

換行符轉換程式, fromdos(1), todos(1), frommac(1), 和 tomac(1), 是相當方便. recode(1) 也是有用的。

注
在 Debian 系統上的一些資料，如 python-moinmoin 軟體包的 wiki 頁面資料，使用 MSDOS 式樣的 CR-LF 作為換行符編碼。所以，上面的規則僅僅是一個通用規則。

注
大部分編輯器 (比如：vim, emacs, gedit, …) 能夠透明處理 MSDOS 式樣的換行符檔案。

提示
對於混合 MSDOS 和 Unix 式樣的檔案，統一到 MSDOS 換行符式樣，使用”sed -e ’/\r$/!s/$/\r/’” 代替 todos(1) 比較好。(例如，在使用 diff3(1) 移植兩個 MSDOS 式樣的檔案後.) 這是因為 todos 給所有的行增加 CR.

功能	**bsdmainutils**	**coreutils**
把製表符擴展成空格	"col -x"	expand
不把空格擴展成製表符	"col -h"	unexpand

Table 11.4: bsdmainutils 和 coreutils 包中的用於轉換 TAB 的指令列表

11.1.5 TAB 轉換

這裡有一些轉換 TAB 程式碼的專業工具。

indent 包中的 indent(1) 指令能夠重新格式化 C 程式中的空格。

例如 vim 和 emacs 這樣的編輯軟體，可以被用來擴展 TAB。就拿 vim 來說，你能夠按順序輸入":set expandtab" 和":%retab" 指令來擴展 TAB。你也可以按順序輸入 :%set noexpandtab" 和":%retab" 指令來復原。

11.1.6 帶有自動轉換功能的編輯器

像 vim 這樣的現代智慧編輯器軟體是相當聰明的並且能夠處理任何編碼系統以及任何檔案格式。你應該在支援 UTF-8 編碼的控制檯上並在 UTF-8 環境下使用這些編輯器來獲得最好的相容性。

以 latin1（iso-8859-1）編碼儲存的舊西歐語言的 Unix 文字檔案，"u-file.txt"，能通過如下所示的用 vim 輕易的編輯。

```
$ vim u-file.txt
```

這是可能的因為 vim 的檔案編碼自動檢測機制先假定檔案是 UTF-8 編碼，如果失敗了，則假定它是 latin1 編碼。

以 latin2（iso-8859-2）編碼儲存的舊波蘭語的 Unix 文字檔案，"pu-file.txt"，能通過如下所示的用 vim 編輯。

```
$ vim '+e ++enc=latin2 pu-file.txt'
```

以 eucJP 編碼儲存的舊日語的 Unix 文字檔案，"ju-file.txt"，能通過如下所示的用 vim 編輯。

```
$ vim '+e ++enc=eucJP ju-file.txt'
```

以所謂的 shift-JIS 編碼 (更確切的說法是：CP932) 儲存的舊日語 MS-Windows 文字檔案，"jw-file.txt"，能通過如下所示的用 vim 編輯。

```
$ vim '+e ++enc=CP932 ++ff=dos jw-file.txt'
```

當一個檔案用 vim 開啟的時候帶有"++enc" 和"++ff" 選項，在 Vim 指令列輸入":w" 指令會以原格式儲存檔案並且會覆蓋原檔案。你也可以在 Vim 指令列指定儲存檔名及其格式，例如，":w ++enc=utf8 new.txt"。

請查閱 vim 線上幫助中的 mbyte.txt，" 多位元組文字支援" 和表格 11.2來獲得"++enc" 使用的本地值的資訊。

emacs 家族的程式能夠實現同樣的功能。

11.1.7 提取純文字

如下所示讀入 web 頁面並把它轉化成文字檔案。當從 Web 中拷貝調配或者是在 web 頁面中應用類似 grep(1) 的基礎 Unix 文字工具時，以下指令是非常有用的。

```
$ w3m -dump https://www.remote-site.com/help-info.html >textfile
```

同樣，你可以使用如下所示的工具從其他格式提取純文字資料。

11.1.8 高亮並格式化純文字資料

你可以通過如下所示的來高亮並格式化純文字資料。

軟體包	流行度	大小	關鍵詞	功能
w3m	V:14, I:188	2837	html→text	用”w3m -dump” 指令把 HTML 轉化為文字的轉換器
html2text	V:3, I:53	274	html→text	高階的 HTML 到文字檔案的轉換器 (ISO8859-1)
lynx	V:24, I:330	1948	html→text	用”lynx -dump” 指令把 HTML 轉化為文字的轉化器
elinks	V:3, I:21	1654	html→text	用”elinks -dump” 指令把 HTML 轉化為文字的轉換器
links	V:3, I:29	2314	html→text	用”links -dump” 指令把 HTML 轉化為文字的轉換器
links2	V:1, I:12	5492	html→text	用”links2 -dump” 指令把 HTML 轉化為文字的轉換器
catdoc	V:14, I:153	686	MSWord→text,TeX	轉化 MSWord 檔案到純文字或 TeX 檔案
antiword	V:1, I:7	589	MSWord→text,ps	轉化 MSWord 檔案到純文字或 ps 檔案
unhtml	V:0, I:0	40	html→text	從一個 HTML 檔案裡面刪除標記標籤
odt2txt	V:2, I:38	60	odt→text	從開放文件格式到文字格式的轉化器

Table 11.5: 用於提取純文字資料的工具列表

軟體包	流行度	大小	關鍵詞	說明
vim-runtime	V:18, I:397	36525	高亮	用”:source $VIMRUNTIME/syntax/html.vim” Vim 巨集指令轉化原始碼到 HTML
cxref	V:0, I:0	1190	c→html	從 C 程式到 latex 和 HTML 的轉換器 (C 語言)
src2tex	V:0, I:0	622	高亮	轉換許多原始碼到 TeX (C 語言)
source-highlight	V:0, I:5	2115	高亮	轉換原始碼到帶有高亮顯示的 HTML, XHTML, LaTeX, Texinfo, ANSI 顏色轉義序列和 DocBook 檔案 (C++)
highlight	V:0, I:5	1373	高亮	轉化許多原始碼到帶有高亮顯示的 HTML, XHTML, RTF, LaTeX, TeX or XSL-FO 檔案。(C++)
grc	V:0, I:5	208	text→ 有顏色的	用於任何文字的通用顏色生成器 (Python)
pandoc	V:8, I:45	194495	text→any	通用標記轉化器 (Haskell)
python3-docutils	V:13, I:51	1804	text→any	重構文字文件到 XML (Python)
markdown	V:0, I:9	58	text→html	Markdown 文字文件到 (X)HTML (Perl)
asciidoctor	V:0, I:7	98	text→any	AsciiDoc 文字文件格式化到 XML/HTML (Ruby)
python3-sphinx	V:6, I:23	2755	text→any	基於文件釋出系統 (Python) 重構文字
hugo	V:0, I:5	69551	text→html	基於 Markdown 的靜態站點發布系統 (Go)

Table 11.6: 高亮純文字資料的工具列表

11.2 XML 資料

擴展標記語言 Extensible Markup Language (XML) 是一種標記語言，用於含有結構化資訊的文件。

在 XML.COM 檢視介紹資訊.

- ” 什麼是 XML?”
- ” 什麼是 XSLT?”
- ” 什麼是 XSL-FO?”
- ” 什麼是 XLink?”

11.2.1 XML 的基本提示

XML 文字看起來有些像 HTML. 它能夠使我們管理一個文件的多個格式。一個簡單的 XML 系統是 `docbook-xsl` 軟體包，在這裡使用。

每一個 XML 檔案使用下面的標準 XML 宣告開始。

```
<?xml version=”1.0” encoding=”UTF-8”?>
```

XML 元素的基本語法是按下面的方式標記。

```
<name attribute=”value”>content</name>
```

內容為空的 XML 元素，使用下面的短格式標記。

```
<name attribute=”value” />
```

上面列子中的”`attribute=”value”`” 是可選的。

XML 裡面的註釋部分，是按下面的方式標記。

```
<!-- comment -->
```

不同於增加標記，XML 至少要求使用預定義實體裡的內容來轉化下列字元。

預定義實體	轉化的字元
`&quot;`	”：引號
`&apos;`	’：撇號
`<`	<：小於號
`>`	>：大於號
`&`	&：& 號

Table 11.7: XML 預定義實體列表

注意
“<”或“&”不能在屬性（attributes）或元素（elements）中使用。

注
當 SGML 式樣的使用者定義實體，比如”`&some-tag;`”, 被使用的時候，第一個定義會覆蓋其它的。實體定義在”`<!ENTITY some-tag ”entity value”>`” 裡表示。

注
只要 XML 標記是一致使用某一標籤名集合（一些資料作為內容或屬性值），使用 Extensible Stylesheet Language Transformations (XSLT) 來轉換到另外一個 XML，是一個微不足道的任務。

11.2.2 XML 處理

有許多工具可以用於處理 XML 檔案，比如說：可擴充樣式表語言 Extensible Stylesheet Language (XSL).

一旦你建立了一個好的成形的 XML 檔案，基本上來講，你就可以使用 可擴充樣式表語言轉換 Extensible Stylesheet Language Transformations (XSLT)，將其轉換成任何格式。

格式化物件的可擴充樣式表語言 Extensible Stylesheet Language for Formatting Objects (XSL-FO) 是用來作為格式化的解決方案. `fop` 軟體包比 Debian `main` 檔案庫要新，因為它依賴 Java 程式語言. LaTeX 程式碼通常是從 XML 使用 XSLT 生成，LaTe 圖形化界面是用來建立 DVI, PostScript 和 PDF 這類可列印的檔案。

軟體包	流行度	大小	關鍵詞	說明
`docbook-xml`	I:398	2134	xml	DocBook 的 XML 文件型別定義（DTD）
`docbook-xsl`	V:13, I:143	14851	xml/xslt	使用 XSLT 將 DocBook XML 處理成各種輸出格式的 XSL 樣式表
`xsltproc`	V:16, I:79	162	xslt	XSLT 指令列處理器 (XML → XML, HTML, 純文字，等等)
`xmlto`	V:0, I:14	130	xml/xslt	使用 XSLT 將 XML 轉換到任意格式的轉換器
`fop`	V:0, I:12	285	xml/xsl-fo	轉換 Docbook XML 檔案到 PDF
`dblatex`	V:2, I:10	4635	xml/xslt	使用 XSLT 將 Docbook 檔案轉換為 DVI, PostScript, PDF 文件
`dbtoepub`	V:0, I:0	37	xml/xslt	DocBook XML 到.epub 轉換

Table 11.8: XML 工具列表

由於 XML 是 標準通用標記語言 Standard Generalized Markup Language (SGML)的一個子集, 用於處理 SGML 的擴充工具，也能夠處理 XML，比如說 文件式樣語言和規範語言 Document Style Semantics and Specification Language (DSSSL).

軟體包	流行度	大小	關鍵詞	說明
`openjade`	V:1, I:26	2396	dsssl	ISO/IEC 10179:1996 標準 DSSSL 處理器 (最新的)
`docbook-dsssl`	V:0, I:13	2605	xml/dsssl	使用 DSSSL 處理 DocBook XML 到各種輸出格式的 DSSSL 樣式表
`docbook-utils`	V:0, I:9	287	xml/dsssl	DocBook 檔案的工具包，包括使用 DSSSL 的轉換成其它格式 (HTML, RTF, PS, man, PDF) 的 `docbook2*` 指令
`sgml2x`	V:0, I:0	90	SGML/dsssl	SGML 和 XML 使用 DSSSL 樣式表的轉換器

Table 11.9: DSSSL 工具列表

提示
GNOME 的 `yelp` 往往能夠方便的直接讀取 DocBook XML 檔案，這是因為它可以從 X 獲得適當的渲染。

11.2.3 XML 資料提取

使用下面的方法，你能夠從其它格式提取 HTML 或 XML 資料。

軟體包	流行度	大小	關鍵詞	說明
man2html	V:0, I:1	142	man 手冊頁 → html	從 man 手冊頁到 HTML 的轉換器 (支援 CGI)
doclifter	V:0, I:0	451	troff → xml	troff 到 DocBook XML 的轉換器
texi2html	V:0, I:5	1847	texi → html	從 Texinfo 到 HTML 的轉換器
info2www	V:1, I:2	74	info → html	從 GNU info 到 HTML 的轉換器 (支援 CGI)
wv	V:0, I:5	733	MSWord → 任何格式	從微軟 Word 格式到 HTML, LaTeX, 等格式的檔案轉換器。
unrtf	V:0, I:3	148	rtf → html	從 RTF 到 HTML 等的轉換器
wp2x	V:0, I:0	200	WordPerfect → 任意格式	WordPerfect 5.0 和 5.1 檔案到 TeX, LaTeX, troff, GML 和 HTML

Table 11.10: XML 資料提取工具列表

11.2.4 XML 資料檢查

對於非 XML 的 HTML 檔案，你能夠轉換它們為 XHTML，XHTML 是一個相當成型的 XML 例項。XHTML 能夠被 XML 工具處理.

XML 檔案的語法和在它們中發現的 URL 的完整性，能夠被檢查。

軟體包	流行度	大小	功能	說明
libxml2-utils	V:21, I:211	180	xml ↔ html ↔ xhtml	使用 xmllint(1) 的 XML 指令列工具 (語法檢查，重新格式化，梳理, ⋯)
tidy	V:1, I:9	84	xml ↔ html ↔ xhtml	HTML 語法檢查和重新格式化
weblint-perl	V:0, I:1	32	檢查	用於 HTML 的小巧的語法檢查器
linklint	V:0, I:0	343	連結檢查	快速的網站維護工具及連結檢查器

Table 11.11: XML 美化列印工具列表

一旦適當的 XML 生成，基於標記的內容等，你能夠使用 XSLT 技術提取資料。

11.3 排版

Unix 上的 troff 程式最初是由 AT&T 公司開發的，可以被用做簡單排版。現在被用來建立手冊頁。

Donald Knuth 發明的 Tex 是非常強大的排版工具也是實際上的標準。最初是由 Leslie Lamport 開發的 LaTex 使得使用者可以更為方便的利用 Tex 的強大功能。

軟體包	流行度	大小	關鍵詞	說明
texlive	V:3, I:35	56	(La)TeX	用於排版、預覽和列印的 Te 圖形化界面
groff	V:2, I:38	20720	troff	GNU troff 文字格式化系統

Table 11.12: 排版工具的列表

11.3.1 roff 排版

傳統意義上，roff 是 Unix 上主要的文字處理系統。參見 roff(7), groff(7), groff(1), grotty(1), troff(1), groff_mdoc(7), groff_man(7), groff_ms(7), groff_me(7), groff_mm(7) 和"info groff"。

安裝好 groff 軟體包以後，你輸入"-me" 巨集指令就能看到一份不錯的指導手冊，它的位置是"/usr/share/doc/groff/"。

提示
"`groff -Tascii -me -`" 輸出帶有 ANSI 轉義碼的純文字。如果你想要 manpage 的輸出帶有許多"^H" 和"_"，那麼使用替代指令"`GROFF_NO_SGR=1 groff -Tascii -me -`"。

提示
如果想要移除 `groff` 生成的文字檔案中的"^H" 和"_"，使用"`col -b -x`" 來過濾它。

11.3.2 TeX/LaTeX

Tex Live 軟體提供了全部的 Te 圖形化界面。`texlive` 元包只是 TeX Live 中的一部分，但是它足夠應付日常任務。

這裡有許多可用的 TeX 和 LaTeX 的參考資料。

- The teTeX HOWTO: The Linux-teTeX Local Guide
- `tex`(1)
- `latex`(1)
- `texdoc`(1)
- `texdoctk`(1)
- "The TeXbook", 作者 Donald E. Knuth, (Addison-Wesley)
- "LaTeX - A Document Preparation System", 作者 Leslie Lamport, (Addison-Wesley)
- "The LaTeX Companion", 作者 Goossens, Mittelbach, Samarin, (Addison-Wesley)

這是最強大的排版環境。許多 SGML 處理器把它作為其後臺字處理工具。`lyx` 軟體包提供的 Lyx 和 `texmacs` 軟體包提供的 GNU TeXmacs 都為 LaTeX 提供了非常不錯的所見即所得的編輯環境，然而許多人使用 Emacs 和 Vim 作為其原始碼編輯器。

有許多線上資源存在。

- TEX Live Guide - TEX Live 2007 ("`/usr/share/doc/texlive-doc-base/english/texlive-en/live.html`") (`texlive-doc-base` 包)
- Latex/Lyx 的一個簡單指引
- 使用 LaTeX 進行文書處理

當文件變得更大時，TeX 有時會出錯。你必須在"`/etc/texmf/texmf.cnf`" 中增加 pool 的大小 (更確切的說話是編輯是"`/etc/texmf/texmf.d/95NonPath`" 並且執行 `update-texmf`(8)) 來修復此問題。

注
"The TeXbook" 的 TeX 原始碼可以從 texbook.tex 的 www.ctan.org tex-archive 站點上下載。此檔案包含了絕大多數所需的宏指令。我聽說把文件中的第 7 到第 10 行註釋了並且新增"`\input manmac \proofmodefalse`"，就可以用 `tex`(1) 來處理此文件。我強烈建議去購買這本書 (還有 Donald E. Knuth 寫的其他書) 而不是使用線上版本，但是線上版本中的原始碼確實是學習 Tex 輸入很好的例子！

11.3.3 漂亮的列印手冊頁

你能夠用如下任意一個指令在印表機上漂亮的列印手冊頁。

```
$ man -Tps some_manpage | lpr
```

11.3.4 建立手冊頁

儘管用純 troff 格式寫手冊頁（manpage）是可能的，這裡還是有一些輔助的程式包用於建立手冊頁。

軟體包	流行度	大小	關鍵詞	說明
docbook-to-man	V:0, I:8	191	SGML→man 手冊頁	從 DocBook SGML 到 roff 手冊頁巨集指令的轉換器
help2man	V:0, I:7	542	text→man 手冊頁	通過 --help 參數自動生成手冊頁的工具
info2man	V:0, I:0	134	info→man 手冊頁	轉換 GNU info 到 POD 或手冊頁的轉換器
txt2man	V:0, I:0	112	text→man 手冊頁	把純粹的 ASCII 文字轉化為手冊頁格式

Table 11.13: 建立手冊頁的工具列表

11.4 可印刷的資料

在 Debian 系統中，可列印的資料是 PostScript 格式的。對於非 PostScript 印表機，通用 Unix 列印系統 (CUPS) 使用 Ghostscript 作為其後臺光柵處理程式。

在最近的 Debian 系統中，可印刷的資料，也可以用 PDF 格式表示。

PDF 檔案能夠使用 GUI（圖形使用者介面）的檢視工具顯示，它的排版條目也可以被填充到 GUI（圖形使用者介面）檢視工具，比如 Evince 和 Okular (參見節 7.4); 以及現代瀏覽器，比如 Chromium。

PDF 檔案能夠使用某些影象工具編輯，比如 LibreOffice、Scribus 和 Inkscape (參見節 11.6)。

11.4.1 Ghostscript

處理可印刷的資料的核心是 Ghostscript PostScript 直譯器，它能夠生成光柵影象。

軟體包	流行度	大小	說明
ghostscript	V:159, I:579	179	GPL Ghostscript PostScript/PDF 直譯器
ghostscript-x	V:2, I:39	87	GPL Ghostscript PostScript/PDF 直譯器-X 顯示支援
libpoppler102	V:16, I:136	4274	PDF 渲染庫 (xpdf PDF 瀏覽器的分支)
libpoppler-glib8	V:274, I:482	484	PDF 渲染庫（基於 Glib 的共享庫)
poppler-data	V:126, I:605	13086	用於 PDF 渲染庫的 CMaps (CJK 支援：Adobe-*)

Table 11.14: Ghostscript PostScript 直譯器列表

提示
"gs -h" 能夠顯示 Ghostscript 的調配資訊。

11.4.2 合併兩個 PS 或 PDF 檔案

你能夠使用 Ghostscript 中的 gs(1) 來合併兩個 PostScript(PS) 或可移植文件格式（PDF）檔案。

```
$ gs -q -dNOPAUSE -dBATCH -sDEVICE=pswrite -sOutputFile=bla.ps -f foo1.ps foo2.ps
$ gs -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sOutputFile=bla.pdf -f foo1.pdf foo2.pdf
```

注
PDF 是用途很廣的跨平臺可印刷的資料格式，它本質上是帶有一些額外特性和擴充的壓縮了的 PS 格式。

提示
對於指令列來說，psmerge(1) 和 psutils 包中的其他指令在處理 PostScript 文件時是很有用的。pdftk 包中的 pdftk(1) 在處理 PDF 文件的時候同樣是很好用的。

11.4.3 處理可印刷資料的工具

如下是處理可印刷資料的工具列表。

軟體包	流行度	大小	關鍵詞	說明
poppler-utils	V:158, I:467	717	pdf → ps,text, …	PDF 工具: pdftops, pdfinfo, pdfimages, pdftotext, pdffonts
psutils	V:4, I:69	219	ps → ps	PostScript 檔案轉換工具
poster	V:0, I:3	57	ps → ps	用 PostScript 頁製作大型海報
enscript	V:1, I:14	2130	text → ps, html, rtf	轉化 ASCII 文字到 PostScript, HTML, RTF 或 Pretty-Print
a2ps	V:0, I:10	3979	text → ps	’任何文字到 PostScript’ 的轉換器並且也是相當不錯的列印程式
pdftk	I:38	28	pdf → pdf	PDF 文件轉換工具: pdftk
html2ps	V:0, I:2	261	html → ps	從 HTML 到 PostScript 的轉換器
gnuhtml2latex	V:0, I:0	27	html → latex	從 html 到 latex 的轉換器
latex2rtf	V:0, I:4	495	latex → rtf	轉換 LaTeX 文件到能被 Microsoft Word 讀取的 RTF 格式的文件
ps2eps	V:2, I:42	95	ps → eps	從 PostScript 到 EPS（Encapsulated PostScript）的轉換器
e2ps	V:0, I:0	109	text → ps	帶有日文編碼支援的文字到 PostScript 轉換器
impose+	V:0, I:0	118	ps → ps	PostScript 工具
trueprint	V:0, I:0	149	text → ps	漂亮的列印許多源程式（C, C++, Java, Pascal, Perl, Pike, Sh, 和 Verilog）到 PostScript。(C 語言)
pdf2svg	V:0, I:3	30	ps → svg	PDF 到可升級的向量圖形格式的轉換器
pdftoipe	V:0, I:0	65	ps → ipe	從 PDF 到 IPE ‘s XML 格式的轉換器

Table 11.15: 處理可印刷資料的工具列表

11.4.4 用 CUPS 列印

Unix 通用列印系統（CUPS）中的 lp(1) 和 lpr(1) 指令都提供了自定義列印資料的選項。
你可以使用下列指令中的一個來列印 3 份有裝訂頁碼的檔案。

```
$ lp -n 3 -o Collate=True filename
```

```
$ lpr -#3 -o Collate=True filename
```

你能夠通過”-o number-up=2”,”-o page-set=even”,”-o page-set=odd”,”-o scaling=200”,”-o natural-sca
等等印表機選項來進一步定製印表機操作，詳細的文件參見指令列列印和選項。

11.5 郵件資料轉換

下列郵件資料轉換軟體包捕獲了我的眼球。

軟體包	流行度	大小	關鍵詞	說明
sharutils	V:3, I:37	1415	郵件	shar(1), unshar(1), uuencode(1), uudecode(1)
mpack	V:1, I:12	108	MIME	編碼和解碼 MIME 資訊: mpack(1) 和 munpack(1)
tnef	V:0, I:7	110	ms-tnef	解包 MIME 附件型別"application/ms-tnef"，該格式僅由微軟使用
uudeview	V:0, I:3	105	郵件	下列格式的編碼器和解碼器: uuencode, xxencode, BASE64, quoted printable 和 BinHex

Table 11.16: 有助於郵件資料轉換的軟體包列表

提示
如果郵件客戶端可以配置使用 IMAP4 伺服器，網際網路訊息訪問協議 版本 4 (IMAP4) 伺服器可以用來把郵件從專有郵件系統裡面移出來。

11.5.1 郵件資料基礎

郵件 (SMTP) 資料需要被限制為 7 位資料序列。二進位制資料和 8 位文字資料使用 Multipurpose Internet Mail Extensions (MIME) 網際網路多用途郵件擴充套件 和選擇的字符集編碼到 7 位格式。(參見表格 11.2)。

標準的郵件儲存格式是 mbox，它是依據 RFC2822 (由 RFC822 更新) 來的格式. 參見 mbox(5) (由 mutt 軟體包提供).

對於歐洲語言, 由於沒有什麼 8 位字元，"Content-Transfer-Encoding: quoted-printable" 加 ISO-8859-1 字元集通常被用於郵件。如果歐洲文字是被編碼為 UTF-8, 由於幾乎全是 7 位資料，使用"Content-Transfer-Encoding: quoted-printable" 也是合適的。

對於日語，傳統的"Content-Type: text/plain; charset=ISO-2022-JP" 通常被用於郵件來保持文字在 7 位。但是老的微軟系統會在沒有宣告的情況下使用 Shift-JIS 來發送郵件。如果日語文字是用 UTF-8 編碼, 由於含有許多 8 位資料，使用 Base64 是合適的。其它亞洲語言也是類似情形。

注
如果你的非 Unix 郵件資料可以透過一個具備和 IMAP4 服務通訊的非 Debian 客戶端訪問，你可以透過執行你的 IMAP4 服務來將郵件資料移出。

注
如果你使用其它郵件儲存格式，第一步把它們移動到 mbox 格式比較好。像 mutt(1) 這樣多功能的客戶端程式可以便捷的完成這類操作。

你可以使用 procmail(1) 和 formail(1) 把郵箱內容分開成每一封郵件.

每一封郵件能夠使用來自 mpack 軟體包的 munpack(1) 指令（或其它特異的工具）來獲得 MIME 編碼內容。

11.6 圖形資料工具

如下是關於圖形資料轉換、編輯和管理的工具包。

軟體包	流行度	大小	關鍵詞	說明
gimp	V:51, I:252	19303	圖形 (點陣圖)	GNU 圖形處理程式
imagemagick	I:317	74	圖形 (點陣圖)	圖形處理程式
graphicsmagick	V:1, I:12	5565	圖形 (點陣圖)	影象處理程式 (imagemagick 派生出來的)
xsane	V:12, I:143	2339	圖形 (點陣圖)	用於 SANE 的基於 GTK 的前端圖形介面 (現在存取掃描器就很簡單了)
netpbm	V:27, I:326	8526	圖形 (點陣圖)	圖形介面的轉換工具
libheif-examples	V:0, I:2	191	heif→jpeg(bitmap)	轉化 高效能影象檔案格式 (HEIF) 到 JPEG、PNG 或 Y4M 格式，用 heif-convert(1) 命令
icoutils	V:7, I:51	221	png↔ico(bitmap)	MS Windows 符號和游標轉化為 PNG 格式，或者從 PNG 格式轉化為點陣圖格式 (favicon.ico)
scribus	V:1, I:17	30242	ps/pdf/SVG/…	Scribus DTP 編輯器
libreoffice-draw	V:69, I:426	10311	圖形 (向量)	LibreOffice 辦公套件-繪畫
inkscape	V:14, I:114	99800	圖形 (向量)	SVG (可升級向量圖形)編輯器
dia	V:2, I:23	3908	圖形 (向量)	圖表編輯器 (Gtk)
xfig	V:0, I:11	7849	圖形 (向量)	在圖形介面下，互動式的生成影象變得方便
pstoedit	V:2, I:53	1004	ps/pdf→image(向量)	PostScript 和 PDF 檔案到可編輯的向量圖形的轉換器 (SVG)
libwmf-bin	V:7, I:121	151	Windows/image(向量)	Windows 元檔案 (向量圖形資料) 轉換工具
fig2sxd	V:0, I:0	151	fig→sxd(向量)	轉換 XFig 檔案為 OpenOffice.org 繪畫格式
unpaper	V:2, I:17	412	image→image	後處理 OCR 掃描頁面的工具
tesseract-ocr	V:8, I:34	2228	image→text	基於惠普的商業 OCR 引擎的免費 OCR 軟體
tesseract-ocr-eng	V:8, I:34	4032	image→text	OCR 引擎資料：用於英文文字的 tesseract-ocr 語言檔案
gocr	V:0, I:7	545	image→text	免費 OCR 軟體
ocrad	V:0, I:3	578	image→text	免費 OCR 軟體
eog	V:60, I:274	7770	影象 (Exif)	Eye of GNOME 影象瀏覽程式
gthumb	V:4, I:17	5032	影象 (Exif)	影象瀏覽器 (GNOME)
geeqie	V:4, I:15	2256	影象 (Exif)	基於 GTK 的影象瀏覽器
shotwell	V:17, I:252	6263	影象 (Exif)	數碼相片管理器 (GNOME)
gtkam	V:0, I:4	1154	影象 (Exif)	從數碼照相機中檢索多媒體資料的應用 (GTK)
gphoto2	V:0, I:8	947	影象 (Exif)	gphoto2 軟體是指令列方式的管理數碼相機的工具
gwenview	V:33, I:105	11755	影象 (Exif)	圖片瀏覽器 (KDE)
kamera	I:104	998	影象 (Exif)	KDE 上的支援數碼相機的應用軟體
digikam	V:2, I:10	293	影象 (Exif)	用於 KDE 桌面環境的數字照片管理應用
exiv2	V:2, I:27	275	影象 (Exif)	EXIF/IPTC 元資料處理工具
exiftran	V:1, I:15	69	影象 (Exif)	改變數碼照相機的 jpeg 影象格式
jhead	V:0, I:8	132	影象 (Exif)	處理相容 JPEG 檔案 (數碼相機圖片) 的 Exif 中的非圖形部分
exif	V:2, I:41	339	影象 (Exif)	顯示 JPEG 檔案中的 EXIF 資訊的指令列工具
exiftags	V:0, I:3	292	影象 (Exif)	從數碼相機的 JPEG 檔案讀取 Exif 標籤的實用工具
exifprobe	V:0, I:3	499	影象 (Exif)	從數碼圖片中讀取元資料
dcraw	V:1, I:12	583	image(原始的)→ppm	解碼原始的數碼相機圖片
findimagedupes	V:0, I:1	77	image→fingerprint	找到相似或重複的影象
ale	V:0, I:0	839	image→image	合併影象來增加保真度或者用於建立馬賽克
imageindex	V:0, I:1	145	image(Exif)→html	從圖形中建立靜態 HTML 相簿
outguess	V:0, I:1	230	jpeg,png	通用的 Steganographic 工具
librecad	V:1, I:15	8963	DXF	CAD 資料編輯器 (KDE)
blender	V:3, I:27	84492	blend, TIFF, VRML, …	用於動畫的 3D 編輯器
mm3d	V:0, I:0	3881	ms3d, obj, dxf, …	基於 OpenGL 的 3D 模型編輯器

提示
在 aptitude(8) (參考節 2.2.6) 中用正規表達式"~Gworks-with::image" 來查詢更多的影象工具。

雖然像 gimp(1) 這樣的圖形介面程式是非常強大的，但像 imagemagick(1) 這樣的指令列工具在用指令碼自動化處理影象時是很有用的。

實際上的數碼相機的影象是可交換的影象檔案格式(EXIF)，這種格式是在 JPEG 影象檔案格式上新增一些元資料標籤。它能夠儲存諸如日期、時間和相機設定的資訊。

The Lempel-Ziv-Welch (LZW) 無損資料壓縮專利已經過期了。使用 LZW 壓縮方式的 圖形互動格式 (GIF) 工具現在可以在 Debian 系統上自由使用了。

提示
任何帶有可移動記錄介質的數碼相機或掃描器都可以在 Linux 上通過 USB 儲存讀取器來工作，因為它遵循相機檔案系統設計規則並且使用 FAT 檔案系統，參考節 10.1.7。

11.7 不同種類的資料轉換工具

這裡有許多其他用於資料轉換的工具。在 aptitude(8) (參考節 2.2.6) 裡用正規表達式"~Guse::converting"" 來查詢如下的軟體包。

軟體包	流行度	大小	關鍵詞	說明
alien	V:1, I:20	163	rpm/tgz → deb	把外來的軟體包轉換為 Debian 軟體包
freepwing	V:0, I:0	424	EB → EPWING	把" 電子書" (在日本流行) 變成單一的 JIS X 4081 格式 (EPWING V1 的子集)
calibre	V:7, I:29	63180	any → EPUB	電子書轉換器和庫管理

Table 11.18: 不同種類的資料轉換工具列表

你能夠通過如下的指令從 RPM 格式的包中提取資料。

```
$ rpm2cpio file.src.rpm | cpio --extract
```

Chapter 12

編程

這裡我給出一些 Debian 系統中的資訊，幫助學習程式設計的人找出打包的原始碼。下面是值得關注的軟體包和與之對應的文件。

安裝 `manpages` 和 `manpages-dev` 包之後，可以通過運行“`man` 名稱”查看手冊頁中的參考資訊。安裝了 GNU 工具的相關文檔包之後，可以通過運行“`info` 程序名稱”查看參考文檔。某些 GFDL 協議的文檔與 DFSG 並不兼容，所以你可能需要在 `main` 倉庫中包含 `contrib` 和 `non-free` 才能下載並安裝它們。

請考慮使用版本控制系統工具。參見節 10.5。

警告
不要用“`test`”作爲可執行的測試文件的名字，因爲 shell 中內建有“`test`”指令。

注意
你可以把從源代碼編譯得到的程序直接放到“`/usr/local`”或“`/opt`”目錄，這樣可以避免與系統程序撞車。

提示
“歌曲：99 瓶啤酒”的代碼示例可以給你提供實踐各種語言的好範本。

12.1 Shell 腳本

Shell 腳本 是指包含有下面格式的可執行的文本文件。

```
#!/bin/sh
 ... command lines
```

第一行指明瞭讀取並執行這個文件的 shell 解釋器。

讀懂 shell 指令碼的最好辦法是先理解類 UNIX 系統是如何工作的。這裡有一些 shell 程式設計的提示。看看“Shell 錯誤”(https://www.greenend.org.uk/rjk/2001/04/shell.html)，可以從錯誤中學習。

不像 shell 交互模式 (參見節 1.5 和節 1.6)，shell 腳本會頻繁使用參數、條件和循環等。

12.1.1 POSIX shell 兼容性

系統中的許多指令碼都可以透過任意 POSIX shell（參見表格 1.13）來執行。

- 預設的非互動 POSIX shell "/usr/bin/sh" 是一個指向到 /usr/bin/dash 的符號連結，並被許多系統程式使用。
- 預設的互動式 POSIX shell 是 /usr/bin/bash。

避免編寫具有 **bashisms**（bash 化）或者 **zshisms**（zsh 化）語法的 shell 指令碼，確保指令碼在所有 POSIX shell 之間具有可移植性。你可以使用 checkbashisms(1) 對其進行檢查。

好的: **POSIX**	應該避免的: **bashism**
`if [ "$foo" = "$bar" ] ; then …`	`if [ "$foo" == "$bar" ] ; then …`
`diff -u file.c.orig file.c`	`diff -u file.c{.orig,}`
`mkdir /foobar /foobaz`	`mkdir /foo{bar,baz}`
`funcname() { …}`	`function funcname() { …}`
八進位制格式: "\377"	十六進位制格式: "\xff"

Table 12.1: 典型 bashism 語法列表

使用"echo" 指令的時候需要注意以下幾個方面，因為根據內建 shell 和外部指令的不同，它的實現也有差別。

- 避免使用除“-n”以外的任何指令列選項。
- 避免在字串中使用轉義序列，因為根據 shell 不同，計算後的結果也不一樣。

注
儘管“-n”選項並不是 POSIX 語法，但它已被廣泛接受。

提示
如果你想要在輸出字串中嵌入轉義序列，用"printf" 指令替代"echo" 指令。

12.1.2 Shell 參數

特殊的 shell 參數經常在 shell 腳本里面被用到。

shell 參數	值
$0	shell 或 shell 指令碼的名稱
$1	第一個 shell 參數
$9	第 9 個 shell 參數
$#	位置參數數量
"$*"	"$1 $2 $3 $4 …"
"$@"	"$1" "$2" "$3" "$4" …
$?	最近一次指令的退出狀態碼
$$	這個 shell 指令碼的 PID
$!	最近開始的後臺任務 PID

Table 12.2: shell 參數列表

如下所示是需要記憶的基本的參數展開。
以上這些操作中":" 實際上都是可選的。

- 有":" 等於測試的 var 值是存在且非空
- 沒有":" 等於測試的 var 值只是存在的，可以為空

參數表示式形式	如果 **var** 變數已設定那麼值為	如果 **var** 變數沒有被設定那麼值為
`${var:-string}`	"`$var`"	"`string`"
`${var:+string}`	"`string`"	"`null`"
`${var:=string}`	"`$var`"	"`string`" (並執行"`var=string`")
`${var:?string}`	"`$var`"	在 **stderr** 中顯示"`string`" (出錯退出)

Table 12.3: shell 參數展開列表

參數替換形式	結果
`${var%suffix}`	刪除位於 var 結尾的 suffix 最小匹配模式
`${var%%suffix}`	刪除位於 var 結尾的 suffix 最大匹配模式
`${var#prefix}`	刪除位於 var 開頭的 prefix 最小匹配模式
`${var##prefix}`	刪除位於 var 開頭的 prefix 最大匹配模式

Table 12.4: 重要的 shell 參數替換列表

12.1.3 Shell 條件語句

每個指令都會回傳 退出狀態，這可以被條件語句使用。

- 成功：0 ("True")
- 失敗：非 0 ("False")

注
"0" 在 shell 條件語句中的意思是"True"，然而"0" 在 C 條件語句中的含義為"False"。

注
"`[`" 跟 `test` 指令是等價的，它評估到"`]`" 之間的參數來作為一個條件表示式.

如下所示是需要記憶的基礎 條件語法。

- "*`command`* `&&` *`if_success_run_this_command_too`* `||` `true`"
- "*`command`* `||` *`if_not_success_run_this_command_too`* `||` `true`"
- 如下所示是多行指令碼片段

```
if [ conditional_expression ]; then
 if_success_run_this_command
else
 if_not_success_run_this_command
fi
```

這裡末尾的"`|| true`"是需要的，它可以保證這個 shell 指令碼在不小心使用了"`-e`"選項而被呼叫時不會在該行意外地退出。

算術整數的比較在條件表示式中為"`-eq`"，"`-ne`"，"`-lt`"，"`-le`"，"`-gt`" 和"`-ge`"。

表示式	回傳邏輯真所需的條件
`-e` *file*	*file* 存在
`-d` *file*	*file* 存在並且是一個目錄
`-f` *file*	*file* 存在並且是一個普通檔案
`-w` *file*	*file* 存在並且可寫
`-x` *file*	*file* 存在並且可執行
file1 `-nt` *file2*	*file1* 是否比 *file2* 新
file1 `-ot` *file2*	*file1* 是否比 *file2* 舊
file1 `-ef` *file2*	*file1* 和 *file2* 位於相同的裝置上並且有相同的 inode 編號

Table 12.5: 在條件表示式中進行檔案比較

表示式	回傳邏輯真所需的條件
`-z` *str*	*str* 的長度為零
`-n` *str*	*str* 的長度不為零
str1 `=` *str2*	*str1* 和 *str2* 相等
str1 `!=` *str2*	*str1* 和 *str2* 不相等
str1 `<` *str2*	*str1* 排列在 *str2* 之前（取決於語言環境）
str1 `>` *str2*	*str1* 排列在 *str2* 之後（取決於語言環境）

Table 12.6: 在條件表示式中進行字串比較

12.1.4 shell 迴圈

這裡有幾種可用於 POSIX shell 的迴圈形式。

- "`for x in foo1 foo2 …; do command ; done`",該迴圈會將"`foo1 foo2 …`"賦予變數"`x`"並執行"`command`"。
- "`while condition ; do command ; done`"，當"`condition`"為真時，會重複執行"`command`"。
- "`until condition ; do command ; done`"，當"`condition`"為假時，會重複執行"`command`"。
- "`break`"可以用來退出迴圈。
- "`continue`"可以用來重新開始下一次迴圈。

提示
C 語言中的數值迭代可以用 `seq`(1) 實現來生成"`foo1 foo2 ...`"。

提示
參見節 9.4.9。

12.1.5 Shell 環境變數

普通的 shell 命令列提示下的一些常見的環境變數，可能在你的指令碼的執行環境中不存在。

- 對於"`$USER`", 使用"`$(id -un)`"
- 對於"`$UID`", 使用"`$(id -u)`"
- 對於"`$HOME`"，使用"`$(getent passwd "$(id -u)"|cut -d ":" -f 6)`" (這個也在節 4.5.2 下工作)

12.1.6 shell 指令列的處理順序

shell 大致以下列的順序來處理一個指令碼。

- shell 讀取一行。
- 如果該行包含有"…" 或 '…'，shell 對該行各部分進行分組作為 一個標識（**one token**）（譯註：one token 是指 shell 識別的一個結構單元）.
- shell 通過下列方式將行中的其它部分分隔進 標識（**tokens**）。
  - 空白字元: 空格 *tab* 換行符
  - 元字元: `< > | ; & ( )`
- shell 會檢查每一個不位於 "…" 或 '...' 的 token 中的 保留字來調整它的行為。
  - 保留字: `if then elif else fi for in while unless do done case esac`
- shell 展開不位於 "…" 或 '...' 中的 別名。
- shell 展開不位於 "…" 或 '...' 中的 波浪線。
  - "~" → 當前使用者的家目錄
  - "*~user*" → *user* 的家目錄
- shell 將不位於 '...' 中的 變數展開為它的值。
  - 變數: "$PARAMETER" 或"${PARAMETER}"
- shell 展開不位於 '...' 中的 指令替換。
  - "$(command)" → "command" 的輸出
  - "` command `" → "command" 的輸出
- shell 將不位於 "…" 或 '...' 中的 **glob** 路徑展開為匹配的檔名。
  - * → 任何字元
  - ? → 一個字元
  - […] → 任何位於"…" 中的字元
- shell 從下列幾方面查詢 指令並執行。
  - 函式定義
  - 內建指令
  - "$PATH" 中的可執行檔案
- shell 前往下一行，並按照這個順序從頭再次進行處理。

雙引號中的單引號是沒有效果的。

在 shell 中執行 "`set -x`" 或使用 "`-x`" 選項啟動 shell 可以讓 shell 顯示出所有執行的指令。這對除錯來說是非常方便的。

軟體包	流行度	大小	說明
dash	V:883, I:997	191	小和快的 POSIX 相容 shell，用於 sh
coreutils	V:879, I:999	18307	GNU 核心工具
grep	V:781, I:999	1266	GNU grep、egrep 和 fgrep
sed	V:787, I:999	987	GNU sed
mawk	V:437, I:997	285	小和快的 awk
debianutils	V:907, I:999	223	用於 Debian 的各種工具
bsdutils	V:519, I:999	356	來自 4.4BSD-Lite 的基礎工具
bsdextrautils	V:582, I:698	339	來自 4.4BSD-Lite 的額外的工具
moreutils	V:15, I:38	244	額外的 Unix 工具

Table 12.7: 包含用於 shell 指令碼的小型應用程式的軟體包

12.1.7 用於 shell 指令碼的應用程式

為了使你的 shell 程式在 Debian 系統上儘可能地具有可移植性，你應該只使用 必要的軟體包所提供的應用程式。

- "`aptitude search ~E`"，列出 必要的軟體包。
- "`dpkg -L` *`package_name`* `|grep '/man/man.*/'`"，列出 *package_name* 軟體包所提供的 man 手冊。

> 提示
> 儘管 moreutils 可能不存在 Debian 之外，但它提供了一些有趣的小程式。最值得注意的一個是 sponge(8)，當你想覆蓋原來的檔案時，它會非常好用。

參見節 1.6 的例子。

12.2 解釋性語言中的指令碼

軟體包	流行度	大小	包
dash	V:883, I:997	191	sh: 小和快的 POSIX 相容的 shell，用於 sh
bash	V:837, I:999	7175	sh: 由 bash-doc 包提供的 "info bash"
mawk	V:437, I:997	285	AWK: 小和快的 awk
gawk	V:286, I:351	2906	AWK: 由 gawk-doc 包提供的 "info gawk"
perl	V:702, I:989	673	Perl: perl(1) 以及透過 perl-doc 和 perl-doc-html 提供的 html 文件
libterm-readline-gnu-perl	V:2, I:29	380	GNU ReadLine/History 庫的 Perl 擴充套件：perlsh(1)
libreply-perl	I:0	171	Perl 的 REPL : reply(1)
libdevel-repl-perl	V:0, I:0	237	Perl 的 REPL : re.pl(1)
python3	V:714, I:951	81	Python: python3(1) 以及透過 python3-doc 包提供的 html 文件
tcl	V:25, I:223	20	Tcl: tcl(3) 以及透過 tcl-doc 包提供的更詳細的手冊頁文件
tk	V:20, I:217	20	Tk:tk(3) 以及透過 tk-doc 包提供的更詳細的手冊頁文件
ruby	V:85, I:211	29	Ruby: ruby(1), erb(1), irb(1), rdoc(1), ri(1)

Table 12.8: 直譯器相關軟體包列表

當你希望在 Debian 上自動化執行一個任務，你應當首先使用解釋性語言指令碼。選擇解釋性語言的準則是：

- 使用 dash，如果任務是簡單的，使用 shell 程式聯合 CLI 命令列程式。
- 使用 python3，如果任務不是簡單的，你從零開始寫。
- 使用 perl、tcl、ruby……，如果在 Debian 上有用這些語言寫的現存程式碼，需要為完成任務進行調整。

如果最終程式碼太慢，為提升執行速度，你可以用編譯型語言重寫關鍵部分，從解釋性語言呼叫。

12.2.1 除錯解釋性語言程式碼

大部分直譯器提供基本的語法檢查和程式碼跟蹤功能。

- “**dash -n** *script.sh*” - Shell 指令碼語法檢查
- “**dash -x** *script.sh*” - 跟蹤一個 Shell 指令碼
- “**python -m py_compile** *script.py*” - Python 指令碼語法檢查
- “**python -mtrace --trace** *script.py*” - 跟蹤一個 Python 指令碼
- “**perl -I ../libpath -c** *script.pl*” - Perl 指令碼語法檢查
- “**perl -d:Trace** *script.pl*” - 跟蹤一個 Perl 指令碼

為測試 dash 程式碼，嘗試下節 9.1.4，它提供了和 bash 類似的互動式環境。

為了測試 perl 程式碼，嘗試下 Perl 的 REPL 環境，它為 Perl 提供了 Python 類似的 REPL (=READ + EVAL + PRINT + LOOP) 環境。

12.2.2 使用 shell 指令碼的 GUI 程式

shell 指令碼能夠被改進用來製作一個吸引人的 GUI（圖形使用者介面）程式。技巧是用一個所謂的對話程式來代替使用 echo 和 read 命令的乏味互動。

軟體包	流行度	大小	說明
x11-utils	V:197, I:564	651	xmessage(1): 在一個視窗中顯示一條訊息或疑問（X）
whiptail	V:274, I:996	56	從 shell 指令碼中顯示使用者友好的對話方塊（newt）
dialog	V:11, I:101	1227	從 shell 指令碼中顯示使用者友好的對話方塊（ncurses）
zenity	V:76, I:362	183	從 shell 指令碼中顯示圖形對話方塊（GTK）
ssft	V:0, I:0	75	Shell 指令碼前端工具 (zenity, kdialog, and 帶有 gettext 的 dialog 封裝)
gettext	V:57, I:259	5817	“/usr/bin/gettext.sh”: 翻譯資訊

Table 12.9: 對話（dialog）程式列表

這裡是一個用來演示的 GUI 程式的例子，僅使用一個 shell 指令碼是多麼容易。

這個指令碼使用 zenity 來選擇一個檔案 (預設 /etc/motd) 並顯示它。

這個指令碼的 GUI 啟動器能夠按節 9.4.10 建立。

```
#!/bin/sh -e
# Copyright (C) 2021 Osamu Aoki <osamu@debian.org>, Public Domain
# vim:set sw=2 sts=2 et:
DATA_FILE=$(zenity --file-selection --filename="/etc/motd" --title="Select a file to check ↩
    ") || \
  ( echo "E: File selection error" >&2 ; exit 1 )
# Check size of archive
```

```
if ( file -ib "$DATA_FILE" | grep -qe '^text/' ) ; then
  zenity --info --title="Check file: $DATA_FILE" --width 640  --height 400 \
    --text="$(head -n 20 "$DATA_FILE")"
else
  zenity --info --title="Check file: $DATA_FILE" --width 640  --height 400 \
    --text="The data is MIME=$(file -ib "$DATA_FILE")"
fi
```

這種使用 shell 指令碼的 GUI 程式方案只對簡單選擇的場景有用。如果你寫一個其它任何複雜的程式，請考慮在功能更強的平臺上寫。

12.2.3 定製 GUI（圖形使用者介面）檔案管理器的行為

GUI（圖形使用者介面）檔案管理器在選定的檔案上，能夠用外加的擴充套件軟體包來擴充套件執行一些常見行為。透過增加特定的指令碼，它們也能夠用來定製執行非常特殊的行為。

- 對於 GNOME，參見 NautilusScriptsHowto。
- 對於 KDE，參見 Creating Dolphin Service Menus。
- 對於 Xfce，參見 Thunar - Custom Actions 和 https://help.ubuntu.com/community/ThunarCustomActions。
- 對於 LXDE，參見 Custom Actions。

12.2.4 Perl 短指令碼的瘋狂

為了處理資料，sh 需要生成子程序執行 cut、grep、sed 等，是慢的。從另外一個方面，perl 有內部處理資料能力，是快的。所以 Debian 上的許多系統維護指令碼使用 perl。

讓我們考慮下面一行 AWK 指令碼片段和它在 Perl 中的等價物。

```
awk '($2=="1957") { print $3 }' |
```

這等價於下列的任意一行。

```
perl -ne '@f=split; if ($f[1] eq "1957") { print "$f[2]\n"}' |
```

```
perl -ne 'if ((@f=split)[1] eq "1957") { print "$f[2]\n"}' |
```

```
perl -ne '@f=split; print $f[2] if ( $f[1]==1957 )' |
```

```
perl -lane 'print $F[2] if $F[1] eq "1957"' |
```

```
perl -lane 'print$F[2]if$F[1]eq+1957' |
```

最後一個簡直就是個迷。它用上了下面列出的這些 Perl 的特性。

- 空格為可選項。
- 存在從數字到字串的自動轉換。
- 透過命令列選項：perlrun(1) 的 Perl 執行技巧
- Perl 特異變數：perlvar(1)

靈活性是 Perl 的強項。與此同時，這允許我們建立令人困惑和繁亂的程式碼。所以請小心。

軟體包	流行度	大小	說明
gcc	V:166, I:551	47	GNU C 編譯器
libc6-dev	V:259, I:569	12051	GNU C 庫：開發庫和標頭檔案
g++	V:53, I:501	14	GNU C++ 編譯器
libstdc++-10-dev	V:15, I:172	17537	GNU 標準 C++ 庫版本 3（開發檔案）
cpp	V:331, I:727	30	GNU C 預處理
gettext	V:57, I:259	5817	GNU 國際化工具
glade	V:0, I:5	1209	GTK 使用者介面構建器
valac	V:0, I:4	724	使用 GObject 系統類似 C# 的語言
flex	V:7, I:74	1243	LEX 相容的 fast lexical analyzer generator
bison	V:7, I:80	3116	YACC 相容的 解析器生成器
susv2	I:0	16	通過“單一 UNIX 規範（版本 2）”獲得（英語文檔）
susv3	I:0	16	通過“單一 UNIX 規範（版本 3）”獲得（英語文檔）
susv4	I:0	16	透過“單一 UNIX 規範（版本 4）”獲取（英語文件）
golang	I:20	11	Go 程式語言編譯器
rustc	V:3, I:14	8860	Rust 系統程式語言
haskell-platform	I:1	12	標準的 Haskell 庫和工具
gfortran	V:6, I:63	16	GNU Fortran 95 編譯器
fpc	I:2	103	自由 Pascal

Table 12.10: 編譯相關軟體包列表

12.3 編譯型語言程式碼

這裡，包括了節 12.3.3 和節 12.3.4，用來說明類似編譯器的程式怎樣用 C 語言來編寫，是透過編譯高階描述到 C 語言。

12.3.1 C

你可以通過下列方法設定適當的環境來編譯使用 C 程式語言編寫的程式。

```
# apt-get install glibc-doc manpages-dev libc6-dev gcc build-essential
```

libc6-dev 軟體包，即 GNU C 庫，提供了 C 標準庫，它包含了 C 程式語言所使用的標頭檔案和庫例程。
參考資訊如下。

- “info libc”（C 庫函式參考）
- gcc(1) 和“info gcc”
- each_C_library_function_name(3)
- Kernighan & Ritchie，“C 程式設計語言”，第二版（Prentice Hall）

12.3.2 簡單的 C 程式（gcc）

一個簡單的例子“example.c”可以通過如下方式和“libm”庫一起編譯為可執行程式“run_example”。

```
$ cat > example.c << EOF
#include <stdio.h>
#include <math.h>
#include <string.h>
```

```
int main(int argc, char **argv, char **envp){
        double x;
        char y[11];
        x=sqrt(argc+7.5);
        strncpy(y, argv[0], 10); /* prevent buffer overflow */
        y[10] = '\0'; /* fill to make sure string ends with '\0' */
        printf("%5i, %5.3f, %10s, %10s\n", argc, x, y, argv[1]);
        return 0;
}
EOF
$ gcc -Wall -g -o run_example example.c -lm
$ ./run_example
        1, 2.915, ./run_exam,     (null)
$ ./run_example 1234567890qwerty
        2, 3.082, ./run_exam, 1234567890qwerty
```

為了使用 sqrt(3)，必須使用“-lm”連結來自 libc6 軟體包的庫“/usr/lib/libm.so”。實際的庫檔案位於“/lib/”，檔名為“libm.so.6”，它是指向“libm-2.7.so”的一個連結。

請看一下輸出文字的最後一段。即使指定了“%10s”，它依舊超出了 10 個字元。

使用沒有邊界檢查的指標記憶體操作函式，比如 sprintf(3) 和 strcpy(3), 是不建議使用，是為防止快取溢位洩露而導致上面的溢位問題。請使用 snprintf(3) 和 strncpy(3) 來替代.

12.3.3 Flex —一個更好的 Lex

Flex 是相容 Lex 的快速語法分析程式生成器。

可以使用“info flex”檢視 flex(1) 的教程。

很多簡單的例子能夠在“/usr/share/doc/flex/examples/”下發現。[1]

12.3.4 Bison —一個更好的 Yacc

在 Debian 裡，有幾個軟體包提供 Yacc相容的前瞻性的 LR 解析 或 LALR 解析的生成器。

軟體包	流行度	大小	說明
bison	V:7, I:80	3116	GNU LALR 解析器生成器
byacc	V:0, I:4	258	伯克利（Berkeley）LALR 解析器生成器
btyacc	V:0, I:0	243	基於 byacc 的回溯解析生成器

Table 12.11: 相容 Yacc 的 LALR 解析器生成器列表

可以使用“info bison”檢視 bison(1) 的教學。

你需要提供你自己的的”main()” 和”yyerror()”. 通常，Flex 建立的”main()” 呼叫”yyparse()”，它又呼叫了”yylex()”.

這裡是一個建立簡單終端計算程式的例子。

讓我們建立 example.y:

```
/* calculator source for bison */
%{
#include <stdio.h>
extern int yylex(void);
extern int yyerror(char *);
%}
```

[1] 在當前系統下，為了讓它們工作，需要做一些 調整。

```
/* declare tokens */
%token NUMBER
%token OP_ADD OP_SUB OP_MUL OP_RGT OP_LFT OP_EQU

%%
calc:
 | calc exp OP_EQU    { printf("Y: RESULT = %d\n", $2); }
 ;

exp: factor
 | exp OP_ADD factor  { $$ = $1 + $3; }
 | exp OP_SUB factor  { $$ = $1 - $3; }
 ;

factor: term
 | factor OP_MUL term { $$ = $1 * $3; }
 ;

term: NUMBER
 | OP_LFT exp OP_RGT  { $$ = $2; }
  ;
%%

int main(int argc, char **argv)
{
  yyparse();
}

int yyerror(char *s)
{
  fprintf(stderr, "error: '%s'\n", s);
}
```

讓我們建立 `example.l`:

```
/* calculator source for flex */
%{
#include "example.tab.h"
%}

%%
[0-9]+ { printf("L: NUMBER = %s\n", yytext); yylval = atoi(yytext); return NUMBER; }
"+"    { printf("L: OP_ADD\n"); return OP_ADD; }
"-"    { printf("L: OP_SUB\n"); return OP_SUB; }
"*"    { printf("L: OP_MUL\n"); return OP_MUL; }
"("    { printf("L: OP_LFT\n"); return OP_LFT; }
")"    { printf("L: OP_RGT\n"); return OP_RGT; }
"="    { printf("L: OP_EQU\n"); return OP_EQU; }
"exit" { printf("L: exit\n");   return YYEOF; } /* YYEOF = 0 */
.      { /* ignore all other */ }
%%
```

按下面的方法來從 shell 提示符執行來嘗試這個：

```
$ bison -d example.y
$ flex example.l
$ gcc -lfl example.tab.c lex.yy.c -o example
$ ./example
$ ./example
1 + 2 * ( 3 + 1 ) =
```

```
L: NUMBER = 1
L: OP_ADD
L: NUMBER = 2
L: OP_MUL
L: OP_LFT
L: NUMBER = 3
L: OP_ADD
L: NUMBER = 1
L: OP_RGT
L: OP_EQU
Y: RESULT = 9

exit
L: exit
```

12.4 靜態程式碼分析工具

類似 lint 的工具能夠幫助進行自動化 靜態程式碼分析。

類似 Indent 的工具能夠幫助人進行程式碼檢查，透過一致性的重新格式化原始碼。

類似 Ctags 的工具能夠幫助人進行程式碼檢查，透過利用原始碼中發現的名字生成索引（或標籤）檔案。

提示
配置你喜歡的編輯器 (emacs 或 vim) 使用非同步 lint 引擎外掛幫助你的程式碼寫作。這些外掛透過充分利用 Language Server Protocol 的優點，會變得非常強大。因它們在快速開發，使用它們上游的程式碼代替 Debian 軟體包，是一個好的選擇。

12.5 除錯

除錯是程式中很重要的一部分。知道怎樣去除錯程式，能夠讓你成為一個好的 Debian 使用者, 能夠做出有意義的錯誤報告。

12.5.1 基本的 gdb 使用指令

Debian 上原始的偵錯程式是 gdb(1), 它能讓你在程式執行的時候檢查程式。

讓我們通過如下所示的指令來安裝 gdb 及其相關程式。

```
# apt-get install gdb gdb-doc build-essential devscripts
```

好的 gdb 教程能夠被發現：

- “info gdb”
- 在 /usr/share/doc/gdb-doc/html/gdb/index.html 的“Debugging with GDB”
- “tutorial on the web”

這裡是一個簡單的列子，用 gdb(1) 在” 程式” 帶有”-g” 選項編譯的時候來產生除錯資訊。

軟體包	流行度	大小	說明
vim-ale	I:0	2591	用於 Vim 8 和 NeoVim 的非同步 Lint 引擎
vim-syntastic	I:2	1379	vim 語法檢查利器
elpa-flycheck	V:0, I:1	808	Emacs 現代實時語法檢查
elpa-relint	V:0, I:0	147	Emacs Lisp 正則錯誤發現器
cppcheck-gui	V:0, I:1	7224	靜態 C/C++ 程式碼分析工具 (GUI)
shellcheck	V:2, I:12	18987	shell 指令碼的 lint 工具
pyflakes3	V:2, I:15	20	Python 3 程式被動檢查器
pylint	V:4, I:19	2018	Python 程式碼靜態檢查器
perl	V:702, I:989	673	帶有內部靜態程式碼檢測的直譯器: B::Lint(3perl)
rubocop	V:0, I:0	3247	Ruby 靜態程式碼分析器
clang-tidy	V:2, I:11	21	基於 clang 的 C++ 規則格式檢查工具
splint	V:0, I:2	2320	靜態檢查 C 程式 bug 的工具
flawfinder	V:0, I:0	205	檢查 C/C++ 原始碼和查詢安全漏洞的工具
black	V:3, I:13	639	強硬的 Python 程式碼格式化器
perltidy	V:0, I:4	2493	Perl 指令碼縮排和重新格式化
indent	V:0, I:8	431	C 語言原始碼格式化程式
astyle	V:0, I:2	785	C、C++、Objective-C、C# 和 Java 的原始碼縮排器
bcpp	V:0, I:0	111	美化 C(++)
xmlindent	V:0, I:1	53	XML 流重新格式化
global	V:0, I:2	1895	原始碼檢索和瀏覽工具
exuberant-ctags	V:2, I:20	341	構建原始碼定義的標籤檔案索引
universal-ctags	V:1, I:11	3386	構建原始碼定義的標籤檔案索引

Table 12.12: 靜態程式碼分析工具的列表

軟體包	流行度	大小	包
gdb	V:14, I:96	11637	由 gdb-doc 包提供的“info gdb”
ddd	V:0, I:7	4105	由 ddd-doc 包提供的“info ddd”

Table 12.13: 除錯軟體包列表

```
$ gdb program
(gdb) b 1               # set break point at line 1
(gdb) run args          # run program with args
(gdb) next              # next line
...
(gdb) step              # step forward
...
(gdb) p parm            # print parm
...
(gdb) p parm=12         # set value to 12
...
(gdb) quit
```

提示
許多 gdb(1) 指令都能被縮寫。Tab 擴展跟在 shell 一樣都能工作。

12.5.2 除錯 Debian 軟體包

Debian 系統在預設情況下，所有安裝的二進位制程式會被 stripped，因此大部分除錯符號（debugging symbols）在通常的軟體包裡面會被移除。為了使用 gdb(1) 除錯 Debian 軟體包，*-dbgsym 軟體包需要被安裝。（例如，安裝 coreutils-dbgsym，用於除錯 coreutils）原始碼軟體包和普通的二進位制軟體包一起自動生成 *-dbgsym 軟體包。那些除錯軟體包將被獨立放在 debian-debug 檔案庫。更多資訊請參閱 Debian Wiki 文件。

如果一個需要被除錯的軟體包沒有提供其 *-dbgsym 軟體包，你需要按如下所示的從原始碼中重構並且安裝它。

```
$ mkdir /path/new ; cd /path/new
$ sudo apt-get update
$ sudo apt-get dist-upgrade
$ sudo apt-get install fakeroot devscripts build-essential
$ apt-get source package_name
$ cd package_name*
$ sudo apt-get build-dep ./
```

按需修改 bug。

軟體包除錯版本跟它的官方 Debian 版本不衝突，例如當重新編譯已存在的軟體包版本產生的”+debug1” 字尾，如下所示是編譯未發行的軟體包版本產生的”~pre1” 字尾。

```
$ dch -i
```

如下所示編譯並安裝帶有除錯符號的軟體包。

```
$ export DEB_BUILD_OPTIONS=”nostrip noopt”
$ debuild
$ cd ..
$ sudo debi package_name*.changes
```

你需要檢查軟體包的構建指令碼並確保編譯二進位制的時候使用了”CFLAGS=-g -Wall” 選項。

12.5.3 獲得棧幀

當你碰到程式崩潰的時候，報告 bug 時附上棧幀資訊是個不錯的注意。

使用如下方案之一，可以透過 gdb(1) 取得棧幀資訊：

- 在 GDB 中崩潰的方案：

  - 從 GDB 執行程式。
  - 崩潰程式。
  - 在 GDB 提示符輸入”bt”。

- 先奔潰的方案：
  - 更新“**/etc/security/limits.conf**”檔案，包括下面內容：

```
* soft core unlimited
```

  - shell 提示符下輸入”`ulimit -c unlimited`”。
  - 從這個 shell 提示符執行程式。
  - 崩潰的程式產生一個 core dump 檔案。
  - 載入 core dump 檔案到 GDB，用”`gdb gdb ./program_binary core`”。
  - 在 GDB 提示符輸入”bt”。

對於無限迴圈或者鍵盤凍結的情況，你可以透過按 `Ctrl-\` 或 `Ctrl-C` 或者執行“`kill -ABRT` *`PID`*”強制奔潰程式。(參見節 9.4.12)

提示
通常，你會看到堆疊頂部有一行或者多行有”`malloc()`” 或”`g_malloc()`”. 當這個出現的時候，你的堆疊不是非常有用的。找到一些有用資訊的一個簡單方法是設定環境變數”`$MALLOC_CHECK_`” 的值為 2 (`malloc`(3)). 你可以通過下面的方式在執行 `gdb` 時設定。

```
$ MALLOC_CHECK_=2 gdb hello
```

12.5.4 高階 gdb 指令

指令	指令用途的描述
`(gdb) thread apply all bt`	得到多執行緒程式的所有執行緒棧幀
`(gdb) bt full`	檢視函式呼叫棧中的參數資訊
`(gdb) thread apply all bt full`	和前面的選項一起得到堆疊和參數
`(gdb) thread apply all bt full 10`	得到前 10 個呼叫的棧幀和參數資訊，以此來去除不相關的輸出
`(gdb) set logging on`	把 gdb 的日誌輸出到檔案 (預設的是”`gdb.txt`”)

Table 12.14: 高階 gdb 指令列表

12.5.5 檢查庫依賴性

按如下所示使用 `ldd`(1) 來找出程式的庫依賴性。

```
$ ldd /usr/bin/ls
        librt.so.1 => /lib/librt.so.1 (0x4001e000)
        libc.so.6 => /lib/libc.so.6 (0x40030000)
        libpthread.so.0 => /lib/libpthread.so.0 (0x40153000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

因為 `ls`(1) 執行在 `chroot`ed 環境，以上的庫在 `chroot`ed 環境也必須是可用的。

參見節 9.4.6。

12.5.6 動態呼叫跟蹤工具

在 Debian 中，有幾個動態呼叫跟蹤工具存在。參見節 9.4。

12.5.7 除錯與 X 相關的錯誤

如果一個 GNOME 程式 `preview1` 收到了一個 X 錯誤，您應當看見一條下面這樣的資訊。

```
The program 'preview1' received an X Window System error.
```

如果就是這種情況，你可以嘗試在執行程式的時候加上"`--sync`" 選項，並且在"`gdk_x_error`" 函式處設定中斷來獲得棧幀資訊。

12.5.8 記憶體洩漏檢測工具

Debian 上有一些可用的記憶體洩漏檢測工具。

軟體包	流行度	大小	說明
libc6-dev	V:259, I:569	12051	`mtrace`(1): 除錯 glibc 中的 malloc
valgrind	V:5, I:36	78191	記憶體偵錯程式和分析器
electric-fence	V:0, I:3	73	`malloc`(e) 偵錯程式
libdmalloc5	V:0, I:2	390	記憶體分配庫除錯
duma	V:0, I:0	296	在 C 和 C++ 程式中檢測快取溢位和快取欠載（buffer under-runs）的庫
leaktracer	V:0, I:1	56	C++ 程式記憶體洩露跟蹤器

Table 12.15: 記憶體洩漏檢測工具的列表

12.5.9 反彙編二進位制程式

你可以使用下面的方式通過 `objdump`(1) 反編譯二進位制程式碼。

```
$  objdump -m i386 -b binary -D /usr/lib/grub/x86_64-pc/stage1
```

注
`gdb`(1) 可以用來互動式反彙編程式碼。

12.6 編譯工具

12.6.1 make

Make 是一個維護程式組的工具。一旦執行 `make`(1)，`make` 會讀取規則檔案 `Makefile`，自從上次目標檔案被修改後，如果目標檔案依賴的相關檔案發生了改變，那麼就會更新目標檔案，或者目標檔案不存在，那麼這些檔案更新可能會同時發生。

規則檔案的語法如下所示。

軟體包	流行度	大小	包
make	V:152, I:556	1592	通過 make-doc 包提供的“info make”
autoconf	V:31, I:232	2025	由 autoconf-doc 包提供的“info autoconf”
automake	V:31, I:231	1837	由 automake1.10-doc 包提供的“info automake”
libtool	V:26, I:215	1213	由 libtool-doc 包提供”info libtool”
cmake	V:16, I:115	36695	cmake(1) 跨平臺、開源的編譯系統
ninja-build	V:6, I:40	428	ninja(1) 接近 Make 精髓的小編譯系統
meson	V:3, I:22	3741	meson(1) 在 ninja 之上的高生產力的構建系統
xutils-dev	V:0, I:9	1484	imake(1)，xmkmf(1) 等。

Table 12.16: 編譯工具軟體包列表

```
target: [ prerequisites ... ]
 [TAB]  command1
 [TAB]  -command2 # ignore errors
 [TAB]  @command3 # suppress echoing
```

這裡面的”[TAB]” 是一個 TAB 程式碼。每一行在進行變數替換以後會被 shell 解釋。在行末使用”\” 來繼續此指令碼。使用”$$” 輸入”$” 來獲得 shell 指令碼中的環境變數值。

目標跟相關檔案也可以通過隱式規則給出，例如，如下所示。

```
%.o: %.c header.h
```

在這裡，目標包含了”%” 字元 (只是它們中確切的某一個)。”%” 字元能夠匹配實際的目標檔案中任意一個非空的子串。相關檔案同樣使用”%” 來表明它們是怎樣與目標檔案建立聯絡的。

自動變數	值
$@	當前目標
$<	首個相關檔案
$?	所有較新的相關檔案
$^	所有相關檔案
$*	目標模式中，$* 指代匹配符”%” 匹配的部分

Table 12.17: 自動變數的列表

變數擴展	說明
foo1 := bar	一次性擴展
foo2 = bar	遞迴擴展
foo3 += bar	增加

Table 12.18: 變數擴展的列表

執行”make -p -f/dev/null” 指令來檢視內部自動化的規則。

12.6.2 Autotools（自動化工具）

Autotools 是一套程式設計工具，被設計作為協助將原始碼軟體包移植到許多 類 Unix 系統。

- Autoconf 是一個從”configure.ac” 生成 shell 指令碼”configure” 的工具。
  - ”configure” 隨後用於從”Makefile.in” 模板生成”Makefile”。
- Automake 是一個從”Makefile.am” 生成”Makefile.in” 的工具。
- Libtool 是一個 shell 指令碼，當從原始碼編譯共享庫時，用來定位軟體的移植性問題。

12.6.2.1 編譯並安裝程式

警告
當你安裝編譯好的程式的時候，注意不要覆蓋系統檔案。

Debian 不會在"/usr/local" 或"/opt" 目錄下建立檔案。如果你想要原始碼編譯程式，把它安裝到"/usr/local/" 目錄下，因為這並不會影響到 Debian。

```
$ cd src
$ ./configure --prefix=/usr/local
$ make # this compiles program
$ sudo make install # this installs the files in the system
```

12.6.2.2 解除安裝程式

如果你有原始碼並且它使用 autoconf(1)/automake(1)，如果你能記得你是怎樣調配它的話，執行如下的指令來解除安裝程式。

```
$ ./configure all-of-the-options-you-gave-it
$ sudo make uninstall
```

或者，如果你十分確信安裝程序把檔案都放在了"/usr/local/" 下並且這裡沒什麼重要的東西，你可以通過如下的指令來清除它所有的內容。

```
# find /usr/local -type f -print0 | xargs -0 rm -f
```

如果你不確定檔案被安裝到了哪裡，你可以考慮使用 checkinstall 軟體包中的 checkinstall(8)，它將會提供一個清晰的解除安裝路徑。現在，它支援建立帶有“-D”選項的 Debian 軟體包。

12.6.3 Meson

軟體構建系統也在演進：

- Autotools 位於 Make 之上，從 90 年代開始，便是可移植構建架構的事實標準。它是相當慢的。
- CMake 在 2000 年初始釋出，顯著地改善了速度，但是它源於建立在本質上慢的 Make 之上。(目前 Ninja 能夠作為它的後端。)
- Ninja 在 2012 年初始釋出，是為了取代 Make，進一步改善構建速度，在設計上，它的輸入檔案由上層的構建系統來生成。
- Meson 在 2013 年初始釋出，是新的和流行的，並且是快速的和上層的構建系統，它使用 Ninja 作為它的後端。

參見在"The Meson Build system" 和"The Ninja build system" 裡發現的文件。

12.7 Web

基本的動態互動網頁可由如下方法制作。

- 呈現給瀏覽器使用者的是 HTML 形式。
- 填充並點選表單條目將會從瀏覽器向 web 伺服器傳送帶有編碼參數的下列 URL 字串之一。

  - "https://www.foo.dom/cgi-bin/program.pl?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3"
  - "https://www.foo.dom/cgi-bin/program.py?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3"
  - "https://www.foo.dom/program.php?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3"
- 在 URL 裡面"%nn" 是使用一個 16 進位制字元的 nn 值代替。
- 環境變數設定為: "QUERY_STRING="VAR1=VAL1 VAR2=VAL2 VAR3=VAL3"".
- Web 伺服器上的CGI程式 (任何一個"program.*") 在執行時，都會使用"$QUERY_STRING" 環境變數.
- CGI 程式的 stdout 傳送到瀏覽器，作為互動式的動態 web 頁面展示。

出於安全考慮，最好不要自己從頭編寫解析 CGI 參數的手藝. 在 Perl 和 Python 中有現有的模組可以使用. PHP 中包含這些功能. 當需要客戶端資料儲存時, 可使用HTTP cookies . 當需要處理客戶端資料時, 通常使用Javascript.

更多資訊，參見 通用閘道器介面, Apache 軟體基金會, 和 JavaScript.

直接在瀏覽器地址中輸入 https://www.google.com/search?hl=en&ie=UTF-8&q=CGI+tutorial 就可以在 Google 上搜索"CGI tutorial"。這是在 Google 伺服器上檢視 CGI 指令碼執行的好方法。

12.8 原始碼轉換

原始碼轉換程式。

軟體包	流行度	大小	關鍵詞	說明
perl	V:702, I:989	673	AWK→PERL	把原始碼從 AWK 轉換為 PERL: a2p(1)
f2c	V:0, I:3	442	FORTRAN→C	把原始碼從 FORTRAN 77 轉換成 C/C++: f2c(1)
intel2gas	V:0, I:0	178	intel→gas	從 NASM (Intel 格式) 轉換成 GNU 彙編程式 (GAS)

Table 12.19: 原始碼轉換工具列表

12.9 製作 Debian 包

如果你想製作一個 Debian 包，閱讀下面內容。

- 章 2 理解基本的包管理系統
- 節 2.7.13 理解基本的移植過程
- 節 9.11.4 理解基本的 chroot 技術
- debuild(1) 和 sbuild(1)
- 節 12.5.2 編譯和除錯
- Debian 維護者指引 (debmake-doc 包)
- Debian 開發者參考手冊 (developers-reference 包)
- Debian 策略手冊 (debian-policy 包)

debmake, dh-make, dh-make-perl 等軟體包，對軟體包打包過程，也有幫助。

Appendix A

附錄

本文檔背景。

A.1 Debian 迷宮

Linux 系統是一個面向網絡計算機的功能強大的計算平臺。然而，學習使用它的全部功能並非易事。使用非 PostScript 的打字機調配 LPR 打字機隊列，就是個好例子。（自從使用新 CUPS 系統的新的安裝系統出現後，就不再會有問題。）

有一張完整而詳盡的地圖叫做“SOURCE CODE”，它非常準確但極難理解。還有一些參考書叫 HOWTO 和 mini-HOWTO，它們易於理解，但給出了太多細節反而讓人忘記了大方向。爲了使用某個指令，我有時得在長長的 HOWTO 中找上半天。

我希望這個“Debian 參考手冊（版本 2.113）”（2024-02-02 13:34:43 UTC）能幫助在 Debian 迷宮裡面徘徊的人們，為他們提供一個好的出發方向。

A.2 版權歷史

Debian 參考手冊由我（Osamu Aoki<osamu at debian dot org>）發起，將其作爲一個個人系統管理筆記。其中的許多內容都是我從Debian 使用者郵件列表和其他 Debian 相關資源獲得的積累。

在採納了來自 Josip Rodin 的建議之後（Josip Rodin 在Debian 文檔項目（DDP）中非常活躍），“Debian 參考手冊（第一版，2001-2007）”成爲了 DDP 文檔的一員。

6 年後，我意識到原來的“Debian 參考手冊（第一版）”內容陳舊，便開始重新很多內容。新的“Debian 參考手冊（第二版）”在 2008 年發佈。

我已經更新了”Debian 參考手冊 (版本 2)” 來處理新的話題 (Systemd, Wayland, IMAP, PipeWire, Linux 核心 5.10)，移除過期話題 (SysV init, CVS, Subversion, SSH 1 協議, 2.5 版本之前的 Linux 核心). Jessie 8 (2015-2020) 版本的情況，或者更老的內容也被大部分移除。

這個”Debian 參考手冊 (version 2.113)” (2024-02-02 13:34:43 UTC) 覆蓋了大部分 Bookworm (=`stable`) 和 Trixie (=`testing`) Debian 版本。

教學的起源和靈感，可以通過下面的內容來追溯。

- ”Linux 使用者手冊” Larry Greenfield (1996 年 12 月)
  - 該文件被後來的《Debian 教學》取代
- ”Debian 教程” Havoc Pennington (1998 年 12 月 11 日)
  - 部分由 Oliver Elphick, Ole Tetlie, James Treacy, Craig Sawyer 和 Ivan E. Moore II 書寫

  - 該文件被後來的《Debian GNU/Linux: 安裝和使用手冊》取代
- ”Debian GNU/Linux: 安裝和使用手冊” John Goerzen 和 Ossama Othman (1999)
  - 該文件被《Debian 參考手冊 (第一版)》取代

軟體包和文件描述的一些起源和靈感，能夠通過下面的內容來追溯。

- ”Debian FAQ” (March 2002 年 3 月版本，當時是由 Josip Rodin 維護)

其它內容的一些起源和靈感，能夠通過下面的內容來追溯。

- ”Debian 參考手冊 (第一版)” Osamu Aoki (2001–2007)
  - 於 2008 年被這個新的”Debian 參考手冊 (第二版)” 取代。

先前的“Debian 參考手冊 (第一版)”由許多貢獻者創建。

- Thomas Hood 是網絡調配主題的主要內容貢獻者
- Brian Nelson 突出貢獻了關於 X 和 VCS 的相關主題
- Jens Seidel 對構建腳本和許多內容的更正提供了幫助
- David Sewell 進行了大量的校對
- 來自翻譯者、貢獻者和 bug 報告者的許多貢獻

Debian 系統中許多的手冊頁面和 info 資訊頁面，和上游網站頁面，Wikipedia 維基百科文件，被用來作為這個文件的主要參考。在一定範圍內，青木修 (Osamu Aoki) 也考慮了公平使用，它們中的許多地方，尤其是命令的定義，在細心的編輯以適應樣式和本文件的目標後，作為了本文件的短語部分。

gdb 調試器的描述使用了擴展Debian 維基內容的回溯系統, 這是被 Ari Pollak, Loïc Minier, 和 Dafydd Harries 同意的.

除了上面提到的部分之外，“Debian 參考手冊 (版本 2.113) (2024-02-02 13:34:43 UTC)”的大部分內容是我自己的工作。一些貢獻者也會對內容進行更新。

作者 Osamu Aoki 在此感謝所有在文檔寫作過程中曾給予幫助的人。

A.3 繁體中文翻譯

該文檔的中文翻譯，通過 Debian 中文郵件列表召集討論，具體翻譯工作可以通過 salsa git 或 weblate 進行。歡迎大家繼續通過 weblate 參加翻譯: `https://hosted.weblate.org/projects/debian-reference/translations/-zh_Hans/`

該項目繼續徵集校對人員, 歡迎大家在 salsa git 或 weblate 參與, 有翻譯得不好的地方, 可以直接修改。

Debian 官方網站也及時同步了我們的最新翻譯成果: `https://www.debian.org/doc/manuals/debian-reference/index.zh-cn.html` `https://www.debian.org/doc/manuals/debian-reference/index.zh-tw.html`

該手冊軟件包名字為: debian-reference-zh-tw

官方網頁為: `https://packages.debian.org/testing/doc/debian-reference-zh-cn` `https://packages.debian.org/testing/doc/debian-reference-zh-tw`

提示
在 testing 版裡面, 軟件包更新比較及時，大家如果在 apt 來源裡面設定了 testing 來源, 則可以直接用 apt-get install debian-reference-zh-tw 命令安裝該軟體包。安裝軟體包後，就可以在本機看 pdf 格式 (/usr/share/debian-reference/debian-reference.zh-tw.pdf) 的文件。

對該手冊翻譯的任何問題或建議, 歡迎大家在 Debian 中文郵件列表討論:

- debian-chinese-gb dot lists.debian.org
- debian-l10n-chinese dot lists.debian.org

《Debian 參考手冊》(第二版) 翻譯情況如下:

(一) 參與情況

從 weblate 和 git 日誌統計，先後有 1 位翻譯貢獻者。

貢獻者的名字是:

Chih-Hsuan Yang (楊志璿)

A.4 文檔格式

目前，英文原始文件使用 DocBook XML 檔案寫作。此原始檔可被轉換成 HTML、純文字、PostScript 和 PDF。(釋出時會省略部分格式。)