從源代碼打造一個最小化的Linux系統實作篇

Greg O'Keefe, gcokeefe@postoffice.utas.edu.au

2000年09月第0.8版

以下就是從源代碼中打造一個最小化的Linux系統的操作說明．它曾經是從加電啓動到*Bash*提示符*(From PowerUp to Bash Prompt)* <http://www.linuxdoc.org/HOWTO/From-PowerUp-To-Bash-Prompt-HOWTO.html>的一部分．但是我將它們分離開來，以便使得它們更簡短而更為集中化．我們在此所要打造的系統是非常小的，而且并不准備作為工作產品來使用．如果您想從頭開始打造一個有實際用途的系統，請參閱Gerard Beekmans所撰寫的*Linux*空手道實作指南篇 *(Linux From Scratch HOWTO)* <http://www.linuxfromscratch.org>．

目錄

1 您所需要具備的條件

我們首先要安裝一個Linux發行套件比如小紅帽(RedHat)到一個分區上，然后 使用它來在另一個分區上打造一個新的Linux系統．我將我們所要打造的系統 稱為目標(target)而把我們所使用來打造新系統的系統稱為源頭(source)． 可別把這個源頭(source)同我們同時使用的源碼*(source code)*相 混淆了哦．:)

因此，您得需要一台具有兩個獨立分區的機器．如果可能，請盡量使用一台 沒有重要資料在里頭的機器，以免數據受損．您可以使用一個已經存在的 Linux系統作為源頭系統，但是我并不推荐這種方式．如果您不慎遺漏了我們 打造的指令的某些參數，您有可能會意外地在這個系統上安裝了一些沒有必要 的東西，有可能會導致不兼容和沖突．

舊型的PC機硬件，大部分的486機器或者更早的機型，其BIOS都有一些極其 煩人的限制．它們沒有辦法讀取硬盤超過前512兆之后的空間．當然，這個 對于Linux來説并不是什么大問題，因為只要Linux能夠引導啓動了，將使用 Linux自己的磁盤IO，略過BIOS的調用．但是為了能夠讓這些舊型機器能夠 引導Linux，那么Linux內核必須存放在硬盤的前512兆之前的某個位置．如果您 正好有這么個舊型機器，您得准備好一個獨立的且完全在前512兆范圍內的 硬盤分區，并將其挂載為`/boot`．其它的分區就可以在任何位置， 可以任意處理而不必擔心是在硬盤的什么位置了．

上一次我打造這個系統時，所使用的源頭系統是小紅帽6.1(RedHat 6.1)， 我安裝了基本系統，附加有以下軟件包：

- cpp (C++編譯器)
- egcs (增强型C編譯器)
- egcs-c++ (增强型C++語言編譯器)
- patch (打補丁程序)
- make (編譯批處理解釋器)
- dev86 (設備文件包)
- ncurses-devel (ncurses庫開發包)
- glibc-devel (glibc庫開發包)
- kernel-headers(內核源碼頭文件包)

我還安裝了X Window視窗系統和Mozilla網絡瀏覽器以便更輕松地閱讀文檔， 而實際上這兩個東東并不是必要的·在我竣工之時，這個源頭系統大概使用 了350兆的磁盤空間(看起來是多了一些，可是我還在納悶為什么呢)·

工之時的目標系統占用了650兆磁盤空間，但是這個數值包含了所有的源碼 以及中途打造出來的文件·如果空間比較緊湊，您應該在每個軟件包都打造 完畢之后執行一下`make clean`來清除臨時文件·當然了，我對這個 也是有點吃驚的·

最后，您的准備好我們所要用來打造系統的源碼包·這些就是我在本文所討論 的軟件包·這些軟件包都可以從源碼盤里面找到，或者從國際互聯網上找到· 我會給出美國的站點和位于澳大利亞的鏡像站點的地址·

- MAKEDEV (設備生成器包) 美國 `<ftp://tsx-11.mit.edu/pub/linux/sources/sbin>` 另一個是美國 `<ftp://sunsite.unc.edu/pub/Linux/system/admin>`
- Lilo (Linux引導器包) 美國 `<ftp://lrcftp.epfl.ch/pub/linux/local/lilo/>`, 澳大利亞 `<ftp://mirror.aarnet.edu.au/pub/linux/metalab/system/boot/lilo/>`.
- Linux內核包(Kernel) 使用主頁 `<http://www.kernel.org>`上所列舉的鏡像站點 而最好不要使用美國 `<ftp://ftp.kernel.org/pub/linux/kernel>` 站點下載，因為這些地方通常是超負荷運轉的 · 澳大利亞 `<ftp://kernel.mirror.aarnet.edu.au/pub/linux/kernel/>`
- GNU libc庫包 其本身，以及liuxthreads線程附加庫可在以下地址下載到： 美國 `<ftp://ftp.gnu.org/pub/gnu/glibc>`和 澳大利亞 `<ftp://mirror.aarnet.edu.au/pub/gnu/glibc>`
- GNU libc附加庫包 您可能還會需要linuxthreads線程附加庫和libcrypt加密附加庫· 如果libcrypt沒在那個地方找到，那就是因為美國出口法律限制的原因， 那么您就可以從這里弄到 *libcrypt*加密附加庫 `<ftp://ftp.gwdg.de/pub/linux/glibc>`· 通常linuxthreads線程附加庫跟libc庫是放在同一個地方的·
- GNU ncurses 美國 `<ftp://ftp.gnu.org/gnu/ncurses>` 澳大利亞 `<ftp://mirror.aarnet.edu.au/pub/gnu/ncurses>`
- SysVinit (初始化腳本包) 美國 `<ftp://sunsite.unc.edu/pub/Linux/system/daemons/init>` 澳大利亞 `<ftp://mirror.aarnet.edu.au/pub/linux/metalab/system/daemons/init>`
- GNU Bash (命令解釋器包) 美國 `<ftp://ftp.gnu.org/gnu/bash>`澳大利亞 `<ftp://mirror.aarnet.edu.au/pub/gnu/bash>`
- GNU sh-utils (命令解釋器工具包) 美國 `<ftp://ftp.gnu.org/gnu/sh-utils>` 澳大利亞 `<ftp://mirror.aarnet.edu.au/pub/gnu/sh-utils>`
- util-linux (Linux常用工具包) 另外某個地方 `<ftp://ftp.win.tue.nl/pub/linux/utils/util-linux/>` 澳大利亞 `<ftp://mirror.aarnet.edu.au/pub/linux/metalab/system/misc>`本軟件包包含有`agetty`和`login`·

總結一下，您所需要的就是：

- 一台具有兩個分別是400兆和700兆獨立分區的機器，或許您可能會需要少一些·
- 一個Linux發行套件(譬如一個Red Hat光盤)和安裝方式(譬如一個光驅)
- 以上所列舉的源碼的tar包·

我假定您可以自己安裝源頭系統，而用不著我來幫忙．從這里開始，我假定源頭 系統已經安裝好了．

本小項目的第一個里程碑就是使得內核啓動起來然后死翹翹，因為它沒找到 init初始化程序．也就是說我們得安裝一個內核和安裝lilo．為了 順利安裝lilo，我們要用上在目標系統上/dev目錄下的設備文件． lilo需要它們來實現底層必需的磁盤存取來寫入引導扇區．MAKEDEV正是用來 創建這些設備文件的腳本程序(您當然可以只需要從源頭系統當中復制出來， 不過這可是作弊不勞而獲哦)．但是最重要的事情就是，我們需要一個文件 系統來放置所有的這些東西．

2 文件系統

我們的新系統是要安裝在文件系統上的．因此首先我們得使用命令mke2fs來 創建文件系統，然后將其挂載到某個地方．我建議是挂載到/mnt/target這個 目錄上．接下來的操作中，我假定就用這個目錄了．為了節省您的寶貴時間，您可以 在/etc/fstab文件里面添加上這一項，以便每次源頭系統啓動的時候就能夠 自動將這個目錄挂載上．

當我們啓動了目標系統，放置在/mnt/target上的所有東西就會被當成了 放置在/根目錄上．

我們需要在目標系統上建立固定的目錄結構．請參閱"文件層次結構標准(簡稱FHS， File Heirarchy Standard)"，見于13.2 (文件系統)一節來了解 詳情，或者只需要cd切換目錄到目標系統所挂載的地方然后盡管執行以 下命令：

```
mkdir bin boot dev etc home lib mnt root sbin tmp usr var
cd var; mkdir lock log run spool
cd ../usr; mkdir bin include lib local sbin share src
cd share/; mkdir man; cd man
mkdir man1 man2 man3 ... man9
```

因為FHS標准和大部分的軟件包在手冊頁(man page)放置位置處理上并不一致， 因此我們需要做一個符號連接：

```
cd ..; ln -s share/man man
```

3 MAKEDEV(設備生成器)

我們要把源代碼放置到目標系統的/usr/src目錄下面．因此，舉個例 子吧，如果您的目標系統是挂載在/mnt/target這個地方，且您的tar 包是放在/root里面，那么您要做的就是：

```
cd /mnt/target/usr/src
tar -xzvf /root/MAKEDEV-2.5.tar.gz
```

然后就把這些tar包復制到您要解開它們的地方就行了．千萬別迷糊了哦．;->

當您安裝軟件的時候，通常情況下您會把它們安裝在正在使用的系統上．但是我們 并不想這么做，因為我們是要把/mnt/target當做根文件系統(root filesystem)，就是要把這些軟件安裝到這個地方．不同的軟件包有不同的處理 方式．比如說MAKEDEV設備生成器包，您要做的是：

```
ROOT=/mnt/target make install
```

您得先在這個包當中的README說明文件和INSTALL安裝說明文件當中查出這些選項， 或者執行命令./configure -help查看幫助說明·

查看一下MAKEDEV包當中的Makefile文件，看看它是怎樣處理我們在命令 行當中設置的ROOT變量的·接著通過執行man ./MAKEDEV.man來 查看一下它的手冊頁，看看它是怎么起到作用的·您會發現生成我們自己的設備的 方式就是執行cd /mnt/target/dev然后./MAKEDEV generic· 請使用ls命令來看看它都為我們生成了哪些設備文件吧·

4 內核

下一步就是生成內核了·我假設您以前是做過編譯內核這種事的，所以我就長話短說 了·如果要啓動的內核已經准備好的話，那么要安裝lilo就會更容易·請返回到 目標系統的usr/src目錄，然后在那兒解開Linux內核源碼·進入Linux 源碼樹(cd linux)然后使用您最喜歡的方式配置內核，比如make menuconfig·如果您想讓自己的輕松一些，那么您可以為自己配置一個沒有 模塊的內核·如果您已經配置了模塊，那么您就得編輯Makefile文件， 找出INSTALL_MOD_PATH并將其設置為/mnt/target·

現在您可以執行make dep，make bzImage了·如果您設置了模塊 項，可以再執行make modules，make modules_install·把內核 映象文件arch/i386/boot/bzImage和系統函數映象文件System.map 復制到目標系統的boot啓動目錄/mnt/target/boot下面，然后准備安裝 系統引導器lilo了·

5 Lilo系統引導器

Lilo包里面帶有一個很小巧的腳本，名叫QuickInst·請把lilo源碼包 解壓到目標系統的源代碼目錄/mnt/target/usr/src下面，然后執行該 腳本，方法是：ROOT=/mnt/target ./QuickInst·它會詢問您一些關于 您想怎樣安裝lilo的問題·

切記：因為我們已經設置ROOT根系統為目標系統分區了，所以您回答 提問時所給出的文件名同它是密切相關的·比如當它詢問您默認想啓動哪個內核 的時候，您的回答應該是/boot/bzImage，而并不是/mnt/target/boot/bzimage哦·我發現這個腳本里面有個小錯誤，它 會提示說：

```
./QuickInst: /boot/bzImage: no such file
```

但是您甭理這個提示就是了，不會有事的·

我們該讓QuickInst把引導扇區(boot sector)放在何處為妥呢？ 當我們重啓時，我們希望可以選擇引導進入源頭系統或者目標系統或者 其它共存于同一台機器的其它系統·而且我們還希望我們要使用所編譯 的lilo來引導我們新系統的內核·我們怎么把這兩件事情合而為一呢？ 讓我們先跑一小會兒題，看看lilo在一個雙重啓動的Linux系統上是怎 樣引導DOS的·在這樣的一個系統上的lilo.conf文件的內容看 起來可能會跟下面的差不多：

```
prompt
timeout = 50
default = linux
```

```
image = /boot/bzImage
        label  = linux
        root   = /dev/hda1
        read-only

other = /dev/hda2
        label = dos
```

如果機器是這么安裝起來的，那么主引導記錄(MBR，master boot record)就可以 被BIOS讀取并加載，然后MBR加載lilo啓動引導器，而后者則給出一個提示．如果 您在提示后面輸入dos，lilo就會從hda2加載引導記錄，就加載了DOS．

我們所要做的事情跟上頭是一樣的，除了在hda2的引導記錄應該是另外一個lilo 引導記錄之外，也就是在QuickInst所詢問要安裝的那個．因此來自Linux 發行套件的lilo會加載我們所編譯安裝的lilo，然后我們所編譯安裝的lilo就會 加載我們所編譯安裝的內核．當您重啓后，您會看到兩次lilo的提示．

長話短說，當QuickInst詢問您該把引導扇區(boot sector)放到什么地方 時，您就回答目標系統所在的分區，比如說是：/dev/hda2．

現在來修改您的源頭系統上的lilo.conf配置文件，那么看起來會有點像 這個樣子：

```
other = /dev/hda2
        label = target
```

修改完畢，接著執行lilo安裝LILO．我們應該可以第一個引導進入目標系統了．

6 Glibc庫

下一步我們要安裝init，但是同運行在Linux上几乎全部的程序一樣， init使用了GNU C語言庫glibc所提供庫函數，因此我們先得把這個東東 安裝上．

Glibc庫是一個很大而且很復雜的軟件包．在我那個舊型的帶8兆內存的386sx/16機器 上，得花掉我90個小時來完成編譯工作．但是在我那帶有64兆內存的賽楊(Celeron) 433上只花掉了33分鐘．如果您只有8兆內存(或者少得讓人打顫的容量)的話，那就 做好苦熬的准備吧．

glibc的安裝文檔建議在不同的獨立分離目錄里面編譯．這樣做就能夠讓您很輕松 地再次編譯，因為您可以該目錄下面接著編譯．您可能也會想這么做，因為可以為 您節省大約265兆的磁盤空間哦!

跟平常一樣，把glibc-2.1.3.tar.gz(或者其它版本)這個tar包解壓到 /mnt/target/usr/src這個目錄下面．接下來，我們得把附加庫也解壓 到glibc庫目錄下面．所以先cd glibc-2.1.3，然后接著在這個目錄下面 把glibc-crypt-2.1.3.tar.gz和glibc-linuxthreads-2.1.3.tar.gz 這兩個tar包解開．

現在我們就可以生成編譯目錄，設置選項，執行make編譯和安裝glibc庫了．這些 都是我所使用過的命令，但是最好您自己閱讀一下文檔，確認最適合您的狀況的 做法．然而在您開始前，您可能需要執行df命令來查看一下還有多少 剩余空間．您還可以在編譯并安裝完畢glibc庫之后再執行一次看看這玩意兒到底 得占多大地兒．

```
        cd ..
```

```
mkdir glibc-build
../glibc-2.1.3/configure --enable-add-ons --prefix=/usr
make
make install_root=/mnt/target install
```

注意了，我們還有別的方法來告知一個軟件包該裝到什么地方·

7 SysVinit初始化腳本包

編譯并安裝SysVinit可執行代碼是非常之簡潔明了的·我偷懶一次，就給您 操作命令吧·假定您已經解壓并且進入SysVinit源碼目錄了：

```
cd src
make
ROOT=/mnt/target make install
```

另外還有很多與`init`相關的腳本·在SysVinit包里面有一些工作 正常的范例腳本，但是您得自個兒手工安裝了·它們在SysVinit源碼樹中 是有層次地布置在`debian/etc`下面的·您只需要執行類似這樣的命令：`cd ../debian/etc; cp -r * /mnt/target/etc`，直接把 它們復制到目標系統的`etc`目錄下面就行了·當然了，您最好是 在復制之前查看一下·

當重啓之后，目標系統的內核就會加載`init`，一切都該各就其位了· 此時的問題可能是腳本不能正常運行，因為沒有命令解釋器`bash`來 解釋執行這些腳本·而且`init`還會嘗試執行`getty`，但是 根本就沒有`getty`可供運行·請重新啓動并確認沒有其它的錯誤·

8 Ncurses庫

我們所需要的下一個東東是命令解釋器Bash，而bash需要ncurses庫，所以 我們得先安裝這玩意兒·ncurses庫可以代替termcap處理文本屏幕的活計， 同時還通過支持termcap調用提供了向后兼容性·為了擁有一個簡潔新潮的 系統，我覺得最好是禁止舊式的termcap方法·如果您后頭要編譯使用了termcap的較老的應用程序，您可能會不斷地與麻煩為伴了·但是您至少會 知道什么東東使用了什么東東·如果您必須要用，那么您可以重新編譯 ncurses庫，使其帶有termcap支持·

我所使用的命令是：

```
./configure --prefix=/usr --with-install-prefix=/mnt/target --with-shared --disable-termcap
make
make install
```

9 Bash命令解釋器

為了把bash安裝到我原以為它該呆的地方，我花費了很多時間做了大量閱讀和 思考以及不斷地嘗試和出錯，可謂是歷盡千辛萬苦啊·我說使用的配置選項是：

```
./configure --prefix=/mnt/target/usr/local --exec-prefix=/mnt/target --with-curses
```

一旦您已經編譯并安裝了bash之后，您需要生成一個符號連接，就象這樣： `cd /mnt/target/bin; ln -s bash sh`．這是因為腳本通常頭一句 是這么寫著的：

```
#!/bin/sh
```

如果您沒有這么一個符號連接，那么您的腳本就不能運行，因為它們會去尋找 `/bin/sh`而非`/bin/bash`．

如果您愿意，您也可以到這里時重新啓動一次．您會注意到腳本這一次確實運行 了．雖然您還是沒能登錄(login)，這是因為還沒有安裝`getty`或者 `login`這些程序．

10 Util-linux (getty 和login)

軟件包util-linux包含有`agetty`和`login`．我們需要這兩個 程序才能登錄系統(log in)和得到命令行提示符(bash prompt)．在安裝之后， 請在目標系統的`/sbin`目錄下為`agetty`建立一個符號連接到 `getty`．`getty`是所有Unix類系統當中被認為應該呆在那個 地方的程序之一，所以生成連接的主意要强于改動`inittab`來運行 `agetty`．

對于util-linux這個包，我剩下的一個問題就是該包的編譯．這個包還包含 有`more`這個程序，而我沒法讓`make`進程給`more` 在目標系統上做一個指向ncurses 5庫的連接，而不是在源頭系統上指向 ncurses 4庫的連接．我會努力克服這個困難的．

您還得在目標系統上准備一個密碼文件`/etc/passwd`．`login` 登錄程序正是通過查詢該文件來確認您是否允許登錄的．因為此次我們只是 打造一個玩具系統，所以我們可以只設置根系統用戶就夠了，而且不需要任何 密碼!! 只需要在目標系統的密碼文件`/etc/passwd`加上如下一行即可：

```
root::0:0:root:/root:/bin/bash
```

所有的域是通過冒號(:)分隔開的，自左向右分別代表：用戶名稱(user id)， 密碼密串(password)，用戶號碼(user number)，用戶群組號碼(group number)， 用戶真實姓名(user's name)，用戶主目錄(home directory)和缺省命令解釋器 (default shell)．

11 Sh-utils

我們所必須的最后一個軟件包就是GNU sh-utils包．我們此時所需要從這個包里面 得到的唯一的程序就是`stty`，它會在`/etc/init.d/rc`里面用到． 而后者是用于改變運行級別和進入初始化運行級別的腳本．實際上我有一個而且 用過僅包含有`stty`的軟件包，但是卻忘了是從何處得到的了．使用GNU的軟件包是個好主意，因為在里頭還有其它您需要的東西，增加了這些東東會使得 系統可用性更好．

好了，打造完畢．您現在應該擁有一個可以啓動并且能夠提示您登錄的系統了． 輸入"root"，您就會進入命令解釋器了．但是您做不了很多事情，甚至沒有 一個`ls`命令給您看看您的作品里面都有些什么東西．請連續按兩次 TAB鍵，您就會看到可用的命令了．這大概是這個系統中，我所發現的最令我滿意的事情．

12 可用性商榷

看起來好像我們打造的是一個毫無用處的系統．說真的，要讓它能夠有實用 價值也并不是什么難事．首先要做的事情之一就是您應該使得根文件系統(root filesystem)以可讀寫方式挂載起來．SysVinit軟

件包里面有干這活兒的腳本， 就在`/etc/init.d/mountall.sh`里面．還執行了一次`mount -a` 把所有在`/etc/fstab`當中的條目以您所指定的方式挂載起來．請在 目標系統的`etc/rc2.d`目錄下生成一個類似`S05mountall` 的符號連接．

您可能會看到這個腳本會用到您尚未安裝的命令．如果真是這樣，找到包含該 命令的軟件包并安裝之．請參看 13.1 (隨機小技巧(Random Tips))這一小節，了解如何 查找軟件包．

看看在`/etc/init.d`里面的其它腳本．它們大部分都應該包含在任何 正經的系統里面．一次添加一個，別忘了要確定添加下一個之前個個都運行 無誤．

請對照文件層次結構標准(File Heirarchy Standard)，請參看 13.2 (文件系統(Filesystem))一節． 那里有一個命令列表，都是該在`/bin`和`/sbin`的命令．請 確定您已經把那里列舉的所有命令都安裝在系統上了．最好就是再找找相關 這類問題的POSIX文檔來看看．

從此，在這個系統里面添加更多必要的軟件包就真是個事兒了．越是早些把編譯 工具，比如說`gcc`和`make`這些添加進去就越好．一旦這些都 完工了，您就可以利用目標系統來自我生息，就會越來越簡單了．

13 更多信息

13.1 隨機小技巧

如果您的Linux系統上曾經使用RPM安裝過有一個叫做`thingy`命令，而您 想獲知這個命令的源碼來源，那么您就使用如下命令：

```
rpm -qif `which thingy`
```

如果您有小紅帽RedHat的源碼光盤，那么您就可以使用下列命令安裝源碼包了：

```
rpm -i /mnt/cdrom/SRPMS/what.it.just.said-1.2.srpm
```

這個命令會把tar包以及任何RedHat補丁包放到`/usr/src/redhat/SOURCES` 目錄下面．

13.2 資源鏈接

- 有一個關于從源代碼編譯軟件的小型實作指南(mini-howto)，就是 軟件打造小小實作指南*(Software Building mini-HOWTO)* `<http://www.linuxdoc.org/HOWTO/Software-Building.html>`.
- 另外還有一個關于從一窮二白空手起家打造一個Linux系統的實作指南． 該文更為集中于打造一個有實際應用價值的系統，而不僅僅是一個實習． 請看： *Linux*系統空手道實作指南篇*(The Linux From Scratch HOWTO)* `<http://www.linuxfromscratch.org>`．
- *Unix*文件系統標准 *(Unix File System Standard)* `<ftp://tsx-11.mit.edu/pub/linux/docs/linux-standards/fsstnd/>` 還有一個關于Unix文件系統標准的鏈接 `<http://www.pathname.com/fhs/>`．這個標准描述了在一個Unix系統中什么東東該呆在什么位置 以及原因．它還描述了在`/bin`，`/sbin`等等目錄中最小化 的要求．如果您的目標是要打造一個小而全的系統，那么這個標准正是一個 好的參考．

14 Administrivia

14.1 版權聲明(Copyright)

14.2 主頁

本文最新的版本可在此找到： *From Powerup To Bash Prompt* <http://learning.taslug.org.au/power2bash>

本文的中譯版可以找 *Linuxrat* <mailto:linuxrat@gnuchina.org>索取．

14.3 您的反饋意見

我很樂意從讀者您那兒得知任何評論，改進意見和建議．請寫信給我： *Greg O'Keefe* <mailto:gcokeefe@postoffice.utas.edu.au>

14.4 鳴謝錄

本文所提及的產品名稱是相應持有者的商標，在此我一并致謝．

我想對以下人員致謝，因為他們的幫助才有了這篇實作指南．

Michael Emery
因其提醒我注意到Unios．

Tim Little
因其提供了關于`/etc/passwd`的一些線索．

sPaKr on #linux in efnet
因其發現syslogd需要`/etc/services`的支持以及介紹給我 使用短語"rolling your own"來表述從源碼打造系統．

Alex Aitkin
因其引起了我對Vico以及他的"verum ipsum factum"(對編譯進一步的理解) 的注意．

Dennis Scott
因其糾正了我的十六進制計算錯誤．

jdd
因其指出一些拼寫錯誤．

14.5 修訂歷史記錄

14.5.1 0.8

- 最初版本．自"From PowerUp to Bash Prompt(從加電啓動到Bash提示符)"實作 篇分離獨立出來．

14.6 未來計划(TODO)

- 轉換為docbook格式．