

Partclone

開發技術報告

報告組別： 軟體技術組
報告人： 蔡育欽
報告日期： 2007/12/15
2008/11/29
2009/12/25
2010/12/20
2011/12/15

內容目錄

壹、Partclone – 開發源起.....	5
一、備份方式比較.....	5
1.檔案複製：copy, tar, rsync 等工具.....	6
2.區塊複製：partimage, ntfsclone 等專案.....	6
二、相關專案探討.....	6
1.Partimage.....	6
2.LinuxNTFS-ntfsclone.....	7
3.Fsarchiver.....	7
4.Xfsdump 分析.....	7
5.Xfs_copy 分析.....	7
6.Zfsclone.....	8
三、需求分析.....	8
貳、Partclone – 軟體架構.....	8
一、初步開發目標.....	8
二、運作流程.....	10
參、開發架構.....	11
一、使用 GCC 編譯程式.....	12
二、使用 GNU Autotools 工具開發.....	13
三、Deb 檔的包裝.....	15
四、搭配 TRAC 整合目標與除錯.....	16
五、搭配版本控制系統紀錄開發程式.....	17
六、使用 SourceForge.....	18
七、新增檔案系統.....	18
肆、Partclone – 開發進度與成果.....	23
一、2007 開發成果.....	23
1.彈性程式架構.....	23
2.檔案系統 Bitmap 資訊.....	24
3.大檔案支援.....	25
4.i18n 多語系支援.....	25
二、2008 開發成果.....	25
1.執行介面之改善 – 更多友善的資訊.....	25
2.備份壞軌硬碟 – rescue 參數.....	26
3.檢查備份資料 – CRC 參數.....	27
4.實驗與測試方法.....	27

5.達成多語系支援.....	27
三、2009 開發成果.....	28
1.UFS2 檔案支援.....	28
2.EXT4 檔案支援.....	28
3.檔案系統備份修正.....	28
4.Partclone.dd.....	28
5.Partclone.restore.....	29
6.Partclone.chkimg.....	29
7.2009 其他修正.....	29
四、2010 開發成果.....	29
1.Btrfs 檔案系統支援.....	29
2.JFS 檔案系統支援.....	30
3.VMFS 檔案系統支援.....	30
4.XFS 檔案系統 演算法重建.....	31
5.改用 GIT 原始碼管理.....	31
6.增加 French 語系.....	32
7.新增參數.....	32
8.錯誤修正.....	32
9.Partclone 版本分支開發.....	33
五、2011 年開發成果.....	33
1.連續區塊寫入.....	33
2.審查與整合記憶體使用改善程式碼.....	33
3.審查與整合支援 ddrescue 區塊列表程式碼.....	34
4.配合 debian 套件開發.....	34
5.自我測試程式撰寫.....	34
6.檔案系統 API 升級(e2fsprogs, vmfs, ntfs3g).....	34
7.其他修正.....	35
伍、Partclone 操作說明.....	35
一、使用手冊 - man page.....	37
陸、Partclone – 後續改進與心得.....	38
一、2007 後續目標與改善.....	38
1.更多 File System.....	38
2.執行介面之改善 – 更多友善的資訊.....	39
3.備份壞軌硬碟 – rescue 參數.....	39
4.檢查備份資料 – CheckSum 參數.....	39
二、2008 後續目標與改善.....	39
1.Partdiff 與 Partpatch.....	39

2.Partclone.restore.....	40
3.Partclone.dd.....	40
4.Partclone.zfs、Partclone.btrfs.....	40
三、2009 後續目標與改善.....	40
1.Partclone.zfs、Partclone.btrfs.....	40
2.改善記憶體使用.....	40
3.開發 JFS 檔案系統.....	41
4.增加 md5、sha1、adler32 等驗證方式.....	41
5.使用手冊撰寫.....	41
四、2010 後續目標與改善.....	41
1.增益備份.....	41
2.ExFAT 檔案系統支援.....	41
3.ZFS 檔案系統支援.....	42
4.增加 partclone.mount.....	42
5.增加 GUI.....	42
6.多裝置一次寫入.....	42
五、2011 後續目標與改善.....	42
1.調整架構與釋出 Partclone API.....	42
2.架構改寫支援 multi-thread.....	43
3.NILFS 支援.....	43
4.減少 seeking 時間.....	43
六、Partclone – 開發心得.....	43
柒、文獻目錄.....	44

插圖目錄

插圖 1: Partclone 流程圖.....	10
插圖 2: 自由軟體開發模型.....	11
插圖 3: Autotoold 流程圖.....	13
插圖 4: TRAC 功能截圖.....	16
插圖 5: EXECNAME 與 fs_open 程式碼.....	19
插圖 6: fs_close 程式碼.....	19
插圖 7: initial_image_hdr 程式碼.....	20
插圖 8: readbitmap 程式碼.....	20
插圖 9: 程式架構全景圖.....	21
插圖 10: Bitmap 讀取流程圖.....	22

插圖 11: Ncurses 範例.....	24
插圖 12: Dialog 範例.....	24
插圖 13: 中文介面.....	26
插圖 14: 備份截圖.....	33
插圖 15: 標準輸出與標準輸入之運用.....	33
插圖 16: Partclone 的 manpage 說明.....	34

表格目錄

表格 1: 檔案系統之函式庫與備份程式.....	10
表格 2: GCC 常用參數整理.....	12
表格 3: autotools 指令整理.....	14
表格 4: 製作 Deb 檔-常用指令整理.....	14
表格 5: 新增檔案系統異動表.....	17
表格 6: 新檔案系統-函式定義.....	18
表格 7: Partclone 參數說明.....	32

壹、 Partclone – 開發源起

眾所皆知備份是相當重要的事情，不論是企業行號或是個人使用，只要有資料產生、儲存與使用等行為，都需要適時的進行資料備份，已防止任何的意外導致資料遺失。在 Linux 作業系統，常用到的備份方式為 cp (copy) 指令，用複製檔案的方式來做為備份，但並不完整且可能遺忘重要檔案；而整個磁碟的備份包含作業系統與檔案基本上可以使用 dd 指令。而 dd 指令屬於完整備份，亦將不需要的資料也備份起來。著名的專案 Partimage[1] 使用到的概念是判斷需要備份的資料並加以複製。在 Clonezilla[2] 專案中，大部分檔案系統使用 Partimage 來進行備份，除 NTFS 磁碟格式另有 Linux-NTFS[3] 專案的支援而使用 ntfsclone 程式來進行備份。而目前的 Partimage 在判斷檔案系統上時有缺漏，而整體架構亦複雜難以修改，故以成立新專案 Partclone 以簡單、實用為目標進行開發。

一、 備份方式比較

在眾多備份方式中，主要分成檔案複製與區塊複製兩類。檔案複製單純以備份指定檔案或是目錄為目標，相當容易理解與使用。而區塊複製則是備份硬碟區塊。所謂的區塊 (Block) 就是檔案系統的最小單位，通常是由數個 Sector (硬碟最小單位) 組成，檔案系統管理檔案與區塊之關連，將檔案分割成數個區塊至於磁區之中。以下為各程式應用說明：

1. 檔案複製：copy, tar, rsync等工具

利用 copy 指令，可以備份指定檔案如前描述。tar 指令則是將檔案包裝，將數個檔案合併成一個檔案，可搭配壓縮指令加以壓縮，減少檔案佔有空間。而 rsync 是以同步檔案為目標，不會複製相同的檔案以節省時間。

2. 區塊複製：partimage, ntfsclone等專案

著名的 Partimage 專案以備份使用到的區塊為目標，根據檔案系統中的 bitmap 來判斷哪些區塊是被使用到的，如果是使用到的則加以備份。ntfsclone 也是一樣，唯專屬於 NTFS 檔案系統所使用。Clonezilla 專案目前仍是以備份區塊為目標，區塊複製的方式可以進行比較完整的備份，包含檔案、目錄與系統檔案。故 Partclone 同樣會以備份使用到的區塊為目標。

二、 相關專案探討

Linux 底下備份區塊程式著名的有 dd, Partimage, ntfsclone。因為 dd 沒有做檔案系統區塊判斷，備份往往需要耗費很多時間，但其優點就是完整備份，即使不被識別的檔案系統仍然可以備份，目前 Clonezilla 還是有在使用。而目前主要使用的是 Partimage 與 ntfsclone，另外亦有許多檔案系統專案特別開發出來用以備份的工具，各項工具說明如下：

1. Partimage

Partimage 主要是將磁區備份成影像檔，支援的檔案系統有 EXT2/3, Reiser3, FAT16/32, HPFS, JFS, XFS, NTFS 等，而 HFS, UFS 還是 beta 版，尚無法運作。該程式除了備份磁區的同時也增加了許多功能，例如透過網路備份、檔案壓縮、磁區判斷、資料檢驗、分割檔案等。但系統備份需要搭配 LiveCD 例如 SystemRescueCd 的開機光碟進行系統備份與還原，且支援直接燒錄至光碟。且支援跨硬體架構，目前支援在 X86 與 PowerPC 下使用，也因為要跨硬體架構的同時又要做到大量的功能，程式十分龐大，許多程式必須自行撰寫，無法使用系統呼叫程式函式庫(Library)。而且近期之內陸續發現有些檔案系統的 bitmap 判別錯誤、增加壓縮格式困難等問題，使得 Clonezilla 無法充分發揮彈性，至為可惜。

2. LinuxNTFS-ntfsclone

LinuxNTFS 專案以讓 Linux 完全支援 NTFS 檔案系統為目標，ntfsclone 是其中之一，用以備份還原 NTFS 磁區。該程式簡潔方便，完全以有效率且完整的備份 NTFS 磁區資料為目的，不理會壓縮、分割檔案、網路備份等其他功能。可能原因就是這些功能基本上 Linux 都已經有現成專案各自進行，只要透過基本的 Shell Script 語法，將這些工具合併起來使用，完全不需要自己撰寫眾多程式碼。實作中，例如 Clonezilla 運用 Shell Script 將 lzop, ntfsclone 透過 PIPE 合併使用，輕易達到新增壓縮格式之功能。[3]

3. Fsarchiver

Fsarchiver 主要是儲存檔案系統中的檔案內容，最主要的特色是可以還原到不同大小的 partition 或是還原資料到不同的檔案系統。Fsarchiver 是 FILE 層級的備份工具，與 partimage / partclone 等 Block 層級工具不同，因而有不同用途與優勢。使用上，只要是 Kernel 核心有支援的檔案系統，就通通可以備份，備份完整度與限制與 Kernel 一致，也就是 Kernel 不支援的檔案系統，Fsarchiver 也不支援。[4]

4. Xfsdump分析

xfsdump 主要目的在於備份 xfs 檔案系統，主要是備份檔案與其屬性，同時也支援遞增備份等功能。可以輸出備份結果到一般檔案、裝置或是 TAP 中，並且可以支援分割的功能。較特別的式 xfsdump 必須要先 mount，也就是 On-line 備份，不支援 off-line 的方式進行，其行為類似 fsarchiver，但是針對 xfs 有較優秀得演算法語處理原則。[5]

5. Xfs_copy分析

與 xfsdump 目標相同，都是用以備份 xfs 檔案系統。為其屬於 Block 備份程式，透過 bitmap 演算法取得需要備份的 Block，但是產生出來的映像檔較為特殊，檔案與 Partition 大小一樣，但是實際上比較小，因為只有備份 used block，雖然備份的時間減少，但是映像檔大小卻沒有減少。其中，程式支援多裝置一次循序寫入，也就是可以同時還原映像檔到多個磁區之中，此想法應該需要列入 Partclone 未來可以支援的方式。[6]

6. Zfsclone

用來將已經 snapshot 過得 zfs 給 clone 出來，做方式將可作為 partclone.zfs 的參考，透過 Partclone 把檔案系統 snapshot 並 clone 起來。[7]

三、 需求分析

Clonezilla 中大量使用到 Shell Script 來保持應用的彈性，每個程式各自獨立運作，既簡單又有彈性。因此 Partclone 將採用類似 ntfsclone 的方式，產生根據不同檔案系統各自獨立的備份工具，保持簡單有彈性的風格。同時為了避免檔案系統讀取不完整，將儘可能的採用檔案系統所提供的函式庫來讀取檔案系統，如此一來備份的程式將會相依檔案系統函式庫，當檔案系統升級時，程式也隨之更新，不會有太多修改上的困擾。

貳、 Partclone – 軟體架構

Partclone 的軟體架構十分簡單，以下針對開發目標與運作流程做說明。

一、 初步開發目標

根據以上分析，初步定義目標為：

- 磁區備份採區塊複製

這部份與 Partimage 或是 ntfsclone 採相同方式設計，主要是找出檔案系統的 Bitmap 資訊。大部分檔案系統的 Bitmap 是用以紀錄區塊是否被使用，通常以一個 bit 的空間來紀錄狀態。當 Bitmap 資訊顯示“0”表示區塊未被使用，程式將不會備份此區塊；反之當 Bitmap 資訊顯示“1”表示區塊已經被使用，程式將會備份此區塊。而 Bitmap 資訊通常是被檔案系統所決定的，所以正確的讀取 Bitmap 是最重要的部份。

- 相依官方或第三方函式庫

如何讀取正確的 Bitmap 是最重要的部份，如果判斷錯誤，即使只有一個錯誤，很有可能導致整個系統無法還原。根據對 Partimage 原始碼的觀察，其判斷的依據是自行根據檔案系統結構而寫，也就是先了解整個檔案系統架構，在根據這樣的架構來進程式碼的撰寫。但這樣並不十分保險，也就是當檔案系統變更了，程式的部份沒有更新，很有可能會

產生錯誤；或是自行撰寫的程式沒有檔案系統的彈性，也可能失敗。而 ntfsclone 採函式庫的作法，使用 Linux-NTFS 專案所產出的函式庫，所有的動作包涵開啟磁區、讀取資訊、寫入資料、關閉磁區都使用函式庫提供的函式，可以保持整個程式的完整性與一致性。Partclone 也將採用與 ntfsclone 相同的方式來讀取檔案系統，維持檔案系統的完整性。

- 官方文件

讀取 Bitmap 資訊時，不一定都有提供函式庫，有時只提供正式的或非正式的官方文件，而參考這些文件製作出來的程式需要更仔細的校對，除非必要，儘量避免為原則。文件上的內容有時難以理解，或是忽略細節，造成程式開發時不小的障礙。

- 支援檔案系統：

- EXT2/3, Reiserfs3.5/3.6, Reiser4, HFS-plus, XFS, FAT16/32, NTFS

經過初步調查得知以有眾多檔案系統有發展函式庫，這些函式庫可以提供使用者自行撰寫應用程式，例如目前開發的 Partclone。已經調查到的函式庫有：

- EXT2/3，使用由官方支援的 e2fsprogs[8] 專案。
- Reiserfs3，使用第三方支援的 progsreiserfs[9] 專案。
- Reiser4，使用官方支援的 reiser4progs[10] 專案。
- XFS，使用官方支援的 xfsprogs[11] 專案。
- ~~HFS Plus，使用第三方支援的 hfsplus[12] 專案。~~
- HFS Plus 官方文件 Technical Note TN1150[13]。
- NTFS，使用第三方支援的 linux-ntfs[3] 專案。
- FAT12/16/32 WIKI 文件[14]。
- 更多支援檔案系統請參考開發成果 22。

這些函式庫提供許多函式供使用者使用，以自行撰寫出各種應用，同時又不用顧及與檔案系統的一致性，只要正確的使用函式，就可以相容於檔案系統。

- 保持程式彈性運作

要讓 Shell Script 程式可以充分的發揮運作，Partclone 程式本身需要提供相對應的介面，例如標準輸入與標準輸出等。Partclone 只負責正確的讀寫磁碟磁區，之後只需要提供標準輸出或是標準輸入之介面與其他應用

程式溝通，通常 Shell Script 可以透過 PIPE 結合多種應用程式達到不同的目的。

- 多語系支援

計劃也將要支援多種語系，因為程式介面也是十分重要的一環。友善的介面會減少使用者使用上的焦慮，能操作到熟悉的文字介面，在使用上可以更加簡單。因此程式開發支援 i18n 的標準，能給使用者自行增加支援的語系，目前已經支援的有中文與英文，更多支援語系請參考開發成果 22。

二、 運作流程

綜合以上目標，Partclone 的流程圖如下

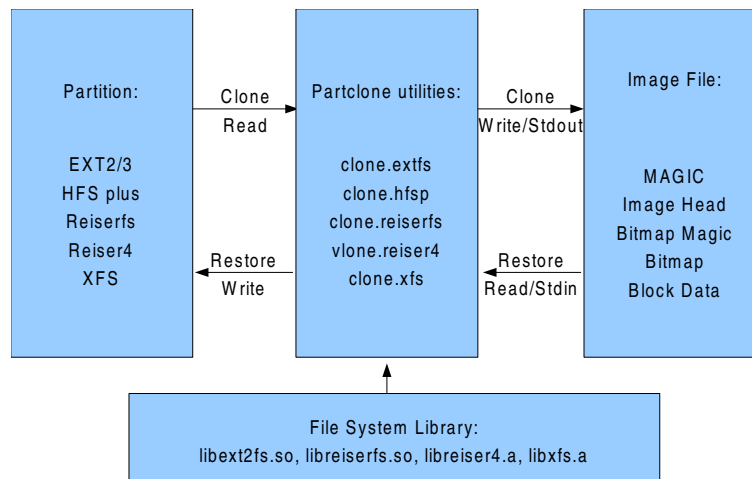


插圖 1: Partclone 流程圖

由[插圖 1]得見，程式將會依據不同的檔案系統而有不同的備份程式，而這些程式都會寫入到特殊格式的 Image 檔案，而各自備份的程式只要讀取到此 Image 檔案也可以還原資料到原本的磁區。程式在與磁區的讀寫上，大多僅使用低階的開啟、關閉、讀檔、寫檔，在第二階段判斷 Bitmap 會使用含是庫提供的函式開啟磁區、讀取 Bitmap、關閉磁區，最後在將資料寫入影像檔時除了提供低階的讀取與寫入外，更提供標準輸入與標準輸出的程式，供 Shell Script 應用。

將檔案系統與備份程式整理成[表格 1]。

檔案系統	專案	相依函式檔案	程式名稱
EXT2/3/4	e2fsprogs	libext2fs.so	partclone.extfs
HFS Plus	hfsplus	無	partclone.hfsp
Reiser3.5/3.6	progsreiserfs	libreiserfs.so	partclone.reiserfs
Reiser4	reiser4progs	libreiser4.a	partclone.reiser4
XFS	xfsprogs	libxfs.a	partclone.xfs
FAT		無	partclone.fat
NTFS	ntfsprogs	libntfs.so	partclone.ntfs

表格 1: 檔案系統之函式庫與備份程式

由[表格 1]得知 Partclone 所開發出的備份程式，都相依特定檔案系統函式庫，如果未來要新增備份程式，只要找到該檔案系統的函式庫，很容易開發並使用。Partclone 也不需要對每個檔案系統而寫特定的程式來維持相容性，程式將隨著檔案系統更新而更新，程式容易維護。

參、開發架構

Partclone 的開發遵循大部分自由軟體開發流程，自由軟體通常是由使用者和開發者社群中針對特定的主題有共同的興趣，由使用者提出各種需求與回饋，讓開發者了解並進行開發如[插圖 2]所示。

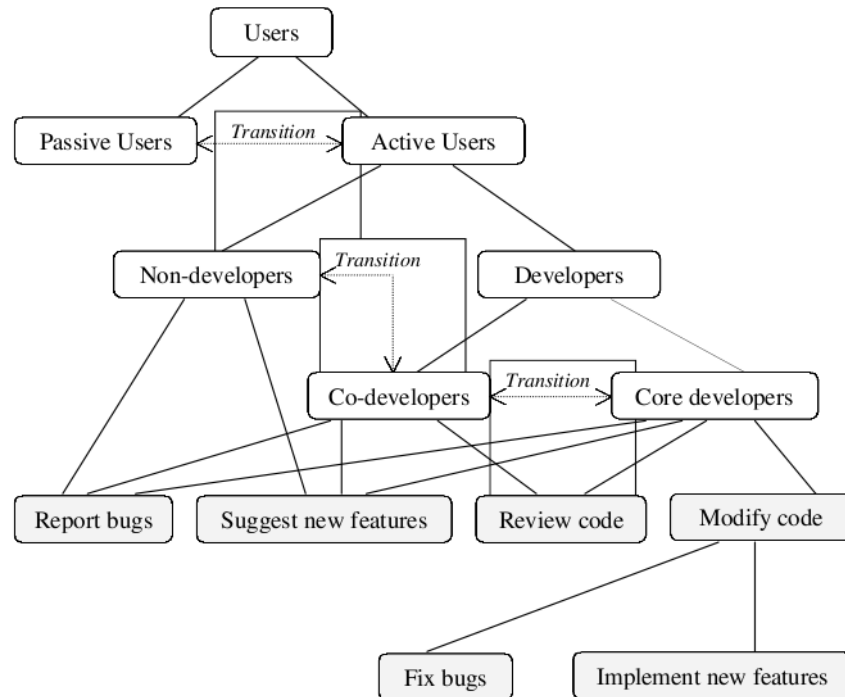


插圖 2: 自由軟體開發模型

與商業軟體相比的話，自由軟體有以下優勢：

- 有能力開發的使用者可以自行開發
- 修改後的版本可獨立發佈
- 使用者的需求更容易被接受
- 更快速的新功能與錯誤修正
- 符合使用者需求與期待

在開發過程還可以使用大量優質的開發工具輔助，減少開發的困擾。同時這些開發工具也都是自由軟體，不僅免費且好用。基於這些開發軟體的貢獻，確實讓所有程式設計有了好的環境，更能開發出優質的自由軟體。

一、 使用 *GCC* 編譯程式

GCC (GNU Compiler Collection, GNU 編譯器套裝)[15]，是一套由 GNU 開發的編程語言編譯器。它是一套以 GPL 及 LGPL 許可證所發行的自由軟體，也是 GNU 計畫的關鍵部分，亦是自由的類 Unix 及蘋果電腦 Mac OS X 操作系統的標準編譯器。

GCC 原名為 GNU C 語言編譯器，因為它原本只能處理 C 語言。*GCC* 很快地

擴展，變得可處理 C++。之後也變得可處理 Fortran, Pascal, Objective-C, Java, 以及 Ada 與其他語言。GCC 目前由世界各地不同的數個程式設計師小組維護。它是移植到中央處理器架構以及作業系統最多的編譯器。由於 GCC 已成為 GNU 系統的官方編譯器(包括 GNU/Linux 家族)，它也成為編譯與建立其他作業系統的主要編譯器，包括 BSD 家族、Mac OS X、NeXTSTEP 與 BeOS。

GCC 通常是跨平臺軟體的編譯器首選。有別於一般侷限於特定系統與執行環境的編譯器，GCC 在所有平臺上都使用同一個前端處理程式，產生一樣的中介碼，因此此中介碼在各個其他平臺上使用 GCC 編譯，有很大的機會可得到正確無誤的輸出程式。常用的 GCC 編譯指令與參數說明整理如[表格 2]。

-c	只做編譯(不做連結)
-S	輸出組譯碼
-o filename	指定輸出檔名
-ansi	程式要求依據 ansi c 標準
-Dmarco	使定義巨集(marco)為有效
-Dmarco=defn	使定義巨集(marco)為 defn
-I	追加 include 檔案的搜尋路徑
-L	追加 library 檔案的搜尋路徑
-l	指定連結的函式庫
-Wall	顯示所有的警告訊息
-g	編入除錯資訊(要使用 GDB 除錯一定要加)
-O2	做最佳化

表格 2: GCC 常用參數整理

二、 使用 *GNU Autotools* 工具開發

GNU build system[16] 是 GNU project 做出來的一套工具，可以幫助軟體在多個 UNIX-like 的系統上執行，而成為跨平台的軟體。它是 GNU 工具鏈的一部分，由 GNU Autoconf[17], GNU Automake[18] 和 GNU Libtool 組合而成。其他相關的工具包

括 GNU make、GNU gettext、pkg-config 工具和 GNU Compiler Collection。GNU build system 受到許多自由軟體和開放源碼套件廣泛的使用，雖然組成 GNU build system 的工具是以 GPL 釋出的自由軟體，但並不限制用它們來做出跨平台的非自由軟體。

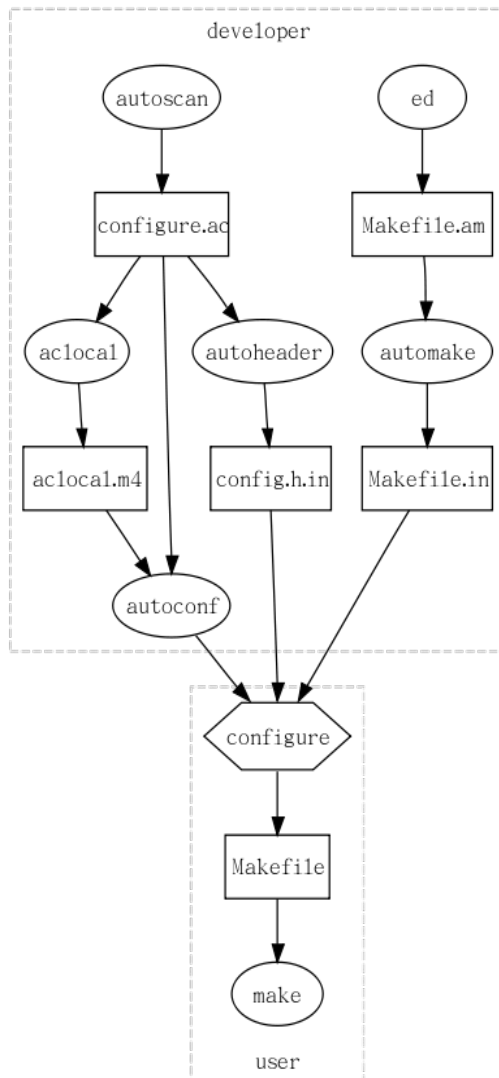


插圖 3: Autotools 流程圖

在 Partclone 中主要使用到 GCC, GNU Autoconf 和 GNU Automake，尤其是 Autoconf 與 Automake 兩自動化工具非常實用。當程式原始碼撰寫完成之後，需使用 GCC 編譯程式進行編譯，很多程式需要編譯又要搭配許多參數因應不同狀況，處理起來步驟多且麻煩，因此大多會撰寫 Makefile 腳本，使用 make 指令進行編譯。而 Automake 正是協助程式設計師快速產生正確的 makefile 腳本之工具之一。程式編譯前通常需要檢查系統是否具有編譯程式的資源與能力，通常會使用 configure 程式來進行，而 Autoconf 則是幫助程式設計師產生 configure 腳本的工具。其使用流程如[插圖 4]，程式設計師只要撰寫 configure.ac 與 makefile.am 透過 Autoconf 與 Automake 工具產生 configure 腳本，執行完 configure 之後就會產生 makefile 供 make 進行編譯。常用的 autoconf 與 automake 指令說明整理如[表格 3]：

指令	動作
aclocal	產生 aclocal.m4
autoconf	產生 configure
autoheader	產生 config.h.in
automake	產生 makefile.in
autoreconf	將所有程序重新執行

表格 3: autotools 指令整理

三、 Deb 檔的包裝

Linux 上主要有 deb 與 rpm 兩種格式。Debian 這個 distribution 製作優良，結構嚴謹早為社群內人士所肯定，其實覺得他的 package 工具與 upgrade 的方法也是比較好的。Deb 是 Debian 軟體包格式的副檔名，Deb 是 unixar 的標準歸檔，將包文件信息以及包內容，經過 gzip 和 tar 打包而成。處理這些包的經典程序是 dpkg，經常是通過 apt 來運作。通過 Alien 工具，可以將 deb 包轉換成 rpm，tar.gz 格式。

指令	動作
dpkg	管理 Debain 套件系統所需程式
dpkg-dev	製作 Package 所需套件
dh-make	製做必要資訊檔的 sample 檔案
debhelper	幫助製作 debian/rules 的工具
devscripts	製作 Debian Package 所需的 scripts
fakeroot	扮演為 root 的工具
lintian	測試與檢查 package 檔的工具
dch	修改 ChangeLog 的工具

表格 4: 製作 Deb 檔-常用指令整理

製作 deb 包裝檔至少需要[表格 4]所需之工具。基本上執行 dh-make 就可以產生 debian 目錄用以包裝 deb 檔。debian 目錄中有許多檔案需要是專案的性質、用途、財產權等加以修改，請參考 Debian New Maintainers' Guide [19]，根據其說明至少需要修改的檔案有：

- debian/control：修改相依性等內容
- debian/copyright：描述財產權等內容
- debian/changelog：紀錄修改紀錄
- debian/rules：製作 deb 檔主要程序

修改完 debian 目錄下的檔案之後，只要下 "dpkg-buildpackage -rfakeroot" 會模擬 root 製作 deb 檔於專案目錄之上。

四、 搭配 *TRAC* 整合目標與除錯

TRAC[20] 是網頁式專案管理與錯誤追蹤的工具，同時也是開放原始碼軟體。整套工具是以 Python 語言開發，2005 年以前都是採用 GPL 授權，直到版本 0.9 以後才改以 BSD 授權，大抵都是自由軟體的方式授權。TRAC 包含有錯誤追蹤、時間軸、瀏覽原始碼、新增檢視代辦事項、搜尋、版本控制、Wiki 文件等功能，版本控制支援 Subversion, Git, Mercurial, 與 Bazaar。

通常在某個軟體專案的開發過程中，可能不會耗時費力地為專案建立專屬的網站或網頁，一般只會把原始碼集中在 CVS 或 Subversion 上，問題或待辦事項以 Issue/Bug Tracking System 來追蹤，而重要的關於專案的細節與訊息溝通，大都透過當面對話、即時訊息系統(ICQ、MSN Messenger)與電子郵件來達成。在這樣的開發過程中，除了資訊四處散落之外，專案發展過程的重要管理項目與經驗等都無法被有效彙總與整理，導致專案的細節只有深入專案的參與人員才能有所體會，但隨著時間流逝，這些重要的體驗也將逐漸被遺忘。為了強化專案資訊的有效流通與專案狀態的傳承，Trac 系統是個不錯的選擇。Trac 的核心是所謂的 Wiki，經由 Wiki 我們可以迅速建立需要的網頁與眾多連結。相對於 HTML、CSS 等的使用，Wiki 是相對簡單且易學的。

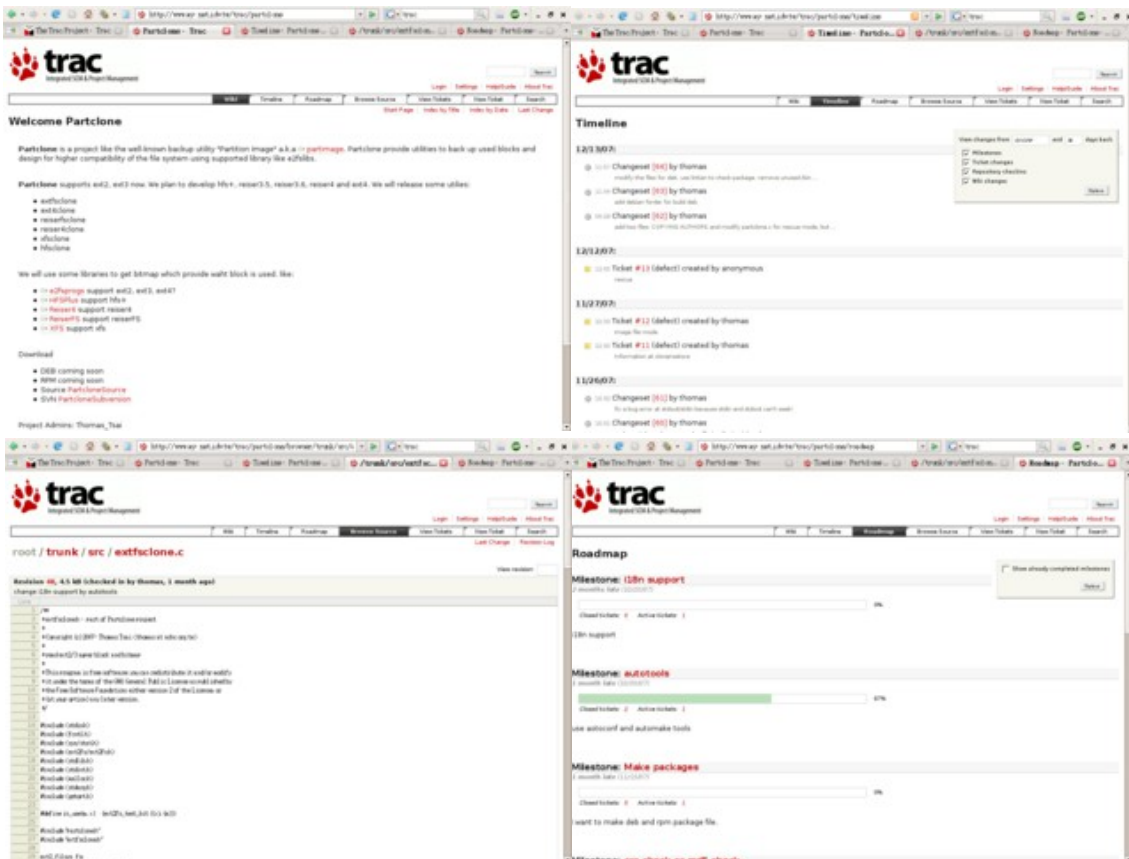


插圖 4: TRAC 功能截圖

實際使用 TRAC 來管理 Partclone 專案時，可以很輕易的使用 Wiki(圖 3 左上) 來新增說明文件，完全不需要修改太多 HTML，好維護且好使用。其中 Timeline(圖 3 右上) 的功能使管理者掌握最新的進度，不論是 wiki 頁面修改、程式碼修改、新增事件等都會在此留下紀錄供管理者處理。當需要討論或是瀏覽程式碼時可以透過 Browse Source(圖 3 左下) 達成，不僅可以看程式碼還可以直接在網頁上比較不同版本之差異。而 Roadmap(圖 3 右下) 可以掌握各模組之進度。

五、 搭配版本控制系統紀錄開發程式

自由軟體開發中，版本控制的觀念是很重要的。不論是早期的 CVS 或是現在常用到的 SVN，都是版本控制系統之一。

CVS 是 Concurrent Versions System 的簡稱[21]。它是現今 Open Source 成功發展的幕後功臣之一。CVS 解決多人合作開發時程式版本控管的問題，通常會再搭配郵件列表(Mailing List)做為開發團隊溝通的管道。這種組合，使開發團隊不受時間地域限制，合作伙伴分散全世界，且團隊大小沒有上限，因此 Open Source 才能集合世界各地的程式設計師，持續不斷地推出高品質的自由軟體。

SVN 是 Subversion version control 的簡稱[22]，雖然在 2006 年時 Subversion 的

使用族群仍然遠少於傳統的 CVS，但已經有許多開放原碼團體決定將 CVS 轉換為 Subversion。有許多的團隊換用 Subversion 是因為 Trac 所提供的專案管理環境。除此之外，SourceForge 除了提供 CVS 外，現在也提供專案開發者使用 Subversion 作為原碼管理系統，而 Google Code 以及 BountySource 則是只提供 Subversion 作為原碼管理系統。Partclone 就是 TRAC 搭配 SVN 一起使用。

本專案於 2010 年以開始改用 Git 作為版本控制工具，並結合 Trac 中 git 的擴充模組整合使用。Git 中設定有 Public 與 Developer 兩角色，一般使用者如有需求可自行 clone 原始碼，符合 GPL 條件下使用，但無法上傳到資料庫中；開發者需要有 ssh 權限，開發完成後可直接寫回，透過此工具大幅增加開發效益，尤其是針對離線操作的開發行為有非常大的幫助。

六、 使用 *SourceForge*

SourceForge 可說是最大的自由軟體專案管理集散地，有許多著名專案都註冊在 SourceForge，該平台提供了許多工具給使用者/開發者使用，為使用者與開發者建立直接面對面的溝通平台。該平台提供了 Forum、Mailing System、TRAC、bug report system、Web Page、wiki、git/svn 等專案管理工具，對使用者查找軟體、下載、提供建議、共筆都非常方便。本專案也註冊於 partclone.sf.net。

七、 新增檔案系統

為了增加新增檔案系統時，必須要補足的項目主要有新檔案系統程式碼、編譯方式、函式庫檢查及文件撰寫，如下表[表格 5]。

修改	檢查函式庫與標頭檔	configure.ac
新增	檔案系統程式碼	src/newfsclone.c
		src/newfsclone.h
修改	新檔案系統註冊	src/partclone.h
修改	程式碼編譯	src/Makefile.am
新增	man 文件原始碼	docs/partclone.newfs.8
		docs/partclone.newfs.8.in
修改	文件編譯	docs/Makefile.am

表格 5: 新增檔案系統異動表

以上 Autotool 的部份建議直接參考手冊

新檔案系統註冊需要增加變數於 src/partclone.h、src/main.c：

```
src/partclone.h
#define newfs_MAGIC "NEWFS"
```

```
src/main.c
#ifdef NEWFS
#include "newfsclone.h"
#define FS "NEWFS"
```

檔案系統的部份，為了與主程式銜接，需要完成函式部份有：

函式	傳入值	用途
fs_open	char* device	開啟磁區
fs_close		關閉磁區
readbitmap	char* device image_head image_hdr char* bitmap	讀取檔案系統 BITMAP 資訊
initial_image_hdr	char* device image_head* image_hdr	讀取檔案基本資訊並初始化 Partclone image 格式

表格 6: 新檔案系統-函式定義

newfsclone.c 中必須定義的 global 變數

EXECNAME //程式執行名稱，如：partclone.newfs

舉例如下：插圖 5、插圖 6、插圖 7、插圖 8

```
aal_device_t      *fs_device;
reiser4_fs_t      *fs = NULL;
reiser4_format_t  *format;
char *EXECNAME = "clone.reiser4";

/// open device
static void fs_open(char* device){
    int debug = 2;

    if (libreiser4_init()) {
        log_mesg(0, 1, 1, debug, "Can't initialize libreiser4.\n");
    }

    if (!(fs_device = aal_device_open(&file_ops, device, 512, 0_RDONLY)))
    {
        log_mesg(0, 1, 1, debug, "Cannot open the partition (%s).\n", device);
    }

    if (!(fs = reiser4_fs_open(fs_device, 0))) {
        log_mesg(0, 1, 1, debug, "Can't open reiser4 on %s", device);
    }

    //reiser4_opset_profile(fs->tree->ent.opset);

    if (!(fs->journal = reiser4_journal_open(fs, fs_device))) {
        log_mesg(0, 1, 1, debug, "Can't open journal on %s", device);
    }

    //reiser4_opset_profile(fs->tree->ent.opset);
    fs->format = reiser4_format_open(fs);
}
```

35,0-1

37

插圖 5: EXECNAME 與 fs_open 程式碼

定義 EXECNAME 與 fs_open

定義 fs_close

```
/// close device
static void fs_close(){
    reiser4 fs_close(fs);
}
```

插圖 6: fs_close 程式碼

定義 initial_image_hdr

```

/// read super block and write to image head
extern void initial_image_hdr(char* device, image_head* image_hdr)
{
    int debug=1;
    reiser4_bitmap_t *fs_bitmap;
    unsigned long long free_blocks=0;

    fs_open(device);
    fs_bitmap = reiser4_bitmap_create(reiser4_format_get_len(fs->format));
    reiser4_alloc_extract(fs->alloc, fs_bitmap);
    free_blocks = reiser4_format_get_free(fs->format);
    memcpy(image_hdr->magic, IMAGE_MAGIC, IMAGE_MAGIC_SIZE);
    memcpy(image_hdr->fs, reiser4_MAGIC, FS_MAGIC_SIZE);
    image_hdr->block_size = (int)get_ms_blksize(SUPER(fs->master));
    image_hdr->totalblock = (unsigned long long)reiser4_format_get_len(fs->format);
    image_hdr->usedblocks = (unsigned long long)(reiser4_format_get_len(fs->format) - free_blocks);
    image_hdr->device_size = (unsigned long long)(image_hdr->block_size * image_hdr->totalblock);
    fs_close();
}

```

插圖 7: initial_image_hdr 程式碼

定義 readbitmap

```

/// readbitmap - read bitmap
extern void readbitmap(char* device, image_head image_hdr, char*bitmap)
{
    reiser4_bitmap_t *fs_bitmap;
    unsigned long long bit, block, bused = 0, bfree = 0;
    int debug = 2;

    fs_open(device);
    fs_bitmap = reiser4_bitmap_create(reiser4_format_get_len(fs->format));
    reiser4_alloc_extract(fs->alloc, fs_bitmap);

    for(bit = 0; bit < reiser4_format_get_len(fs->format); bit++){
        block = bit ;
        if(reiser4_bitmap_test(fs_bitmap, bit)){
            bused++;
            bitmap[block] = 1;
            log_mesg(3, 0, 0, debug, "bitmap is used %lli", block);
        } else {
            bitmap[block] = 0;
            bfree++;
            log_mesg(3, 0, 0, debug, "bitmap is free %lli", block);
        }
    }

    if(bfree != reiser4_format_get_free(fs->format))
        log_mesg(0, 1, 1, debug, "bitmap free count err, bfree:%lli, sfree:%lli\n", bfree, reiser4_format_get_free(fs->format));

    fs_close();
}

```

103,0-1 78

插圖 8: readbitmap 程式碼

肆、 Partclone – 開發進度與成果

目前開發中的 Partclone 已經可以備份 EXT2/3, Reiserfs3, Reiser4, XFS, HFS plus 等檔案系統。在語系支援上，也已經符合 i18n 規範，可以翻譯成各種語言。同時也完成基本的使用說明。

一、 2007 開發成果

1. 彈性程式架構

程式架構如前所述，主要是判斷 Bitmap 資訊來決定要不要備份該磁碟區塊。參考[插圖 9]，程式基本上運作順序為如下所述。

開始時先會取得參數決定是備份模式或是還原模式，如果是備份模式就從磁區讀取 Bitmap，如果是還原模式就從 Image 印像檔讀取 Bitmap。

讀取完 Bitmap 資訊之後，就開始判斷 Bitmap 中的 Bit，每一個 Bit 代表一個 Block，如果 Bit 是 "0" 就表示該區塊沒有被使用到；反之，Bit 非 "0" 就是該區塊是被使用到的，需要備份與還原。

備份與還原的動作多使用 System Call 來進行，大部分使用底層 I/O 的 Function，例如 "open", "close", "read", "write" 等。備份時，根據 Bit 判斷需不需要從磁區讀取資料，如果需要就讀取並寫入印象檔；還原時根據 Bit 判斷是否該從印象檔讀取資料寫入磁區。

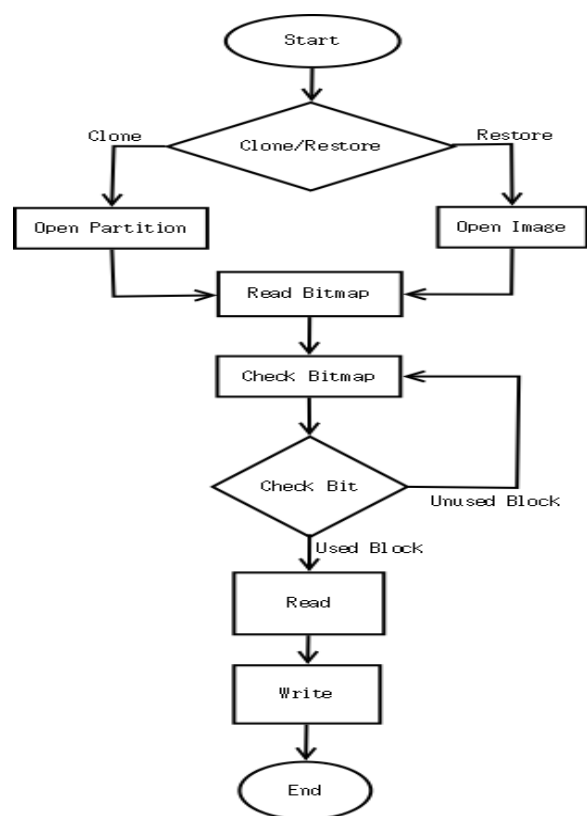


插圖 9: 程式架構全景圖

2. 檔案系統Bitmap資訊

其中比較核心的是如何正確的從磁區讀取判斷 Bitmap。為了正確讀取 Bitmap 與根據磁區資訊，這邊將大量使用函式庫所提供的函式，而根據不同的檔案系統與函式庫，其提供的函式名稱與用法有所差異，因此只能大略描述流程於[插圖 10]。

其流程大致上為是當收到要開始讀取 Bitmap 資訊時，先開啟磁區，通常由與多函式組合而成，並非只是 "open" 而已。接著讀取 Super Block，其中定義許多磁區的 Metadata，包括有磁碟空間、區塊大小、區塊使用量、區塊總數等。接著開始讀取 Bitmap，將其資訊讀取至記憶體，並開始判斷。讀取與判斷都有函式可以使用，程式設計師只要了解用法，而不必觸及檔案系統底層設計，就可以輕易完成，而且也確保資料的正確性與完整性。讀取出來的 Bitmap 將以陣列儲存於記憶體，並將寫入印象檔。完成之後還需正確的釋放記憶體與關閉磁區，使後續動作得以順利進行。

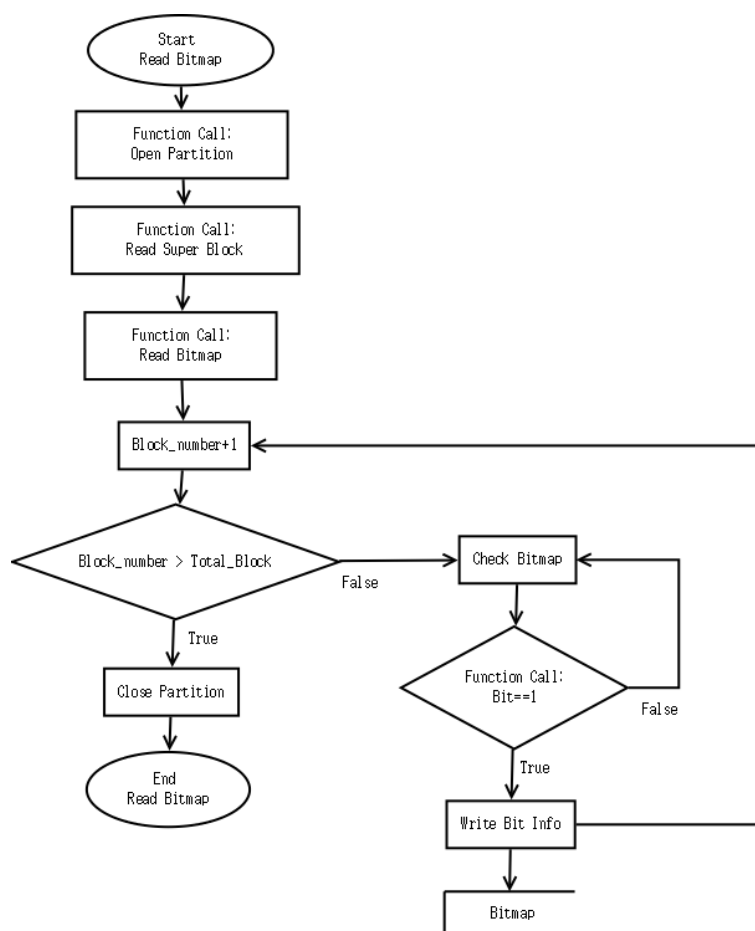


插圖 10: Bitmap 讀取流程圖

3. 大檔案支援

備份過程中曾經遇到的問題是如何備份 2G 以上的印象檔。目前檔案系統都有支援大檔案，但程式設計方面還需要針對大檔案做特殊處理。在 Linux 底下有不少關於 LFS(Large File Support)[23]的資料，經過研究得知在 32 位元的系統上，例如 X86, PowerPC, MIPS 等，都會有單一檔案 2G 限制。在 Glibc 2.2 之後的版本已經解決，作法就是：

1. 編譯時加上 "-D_FILE_OFFSET_BITS=64" 參數，可以讓 "off_t" 的空間變大，等同 "off_t64"。
2. 編譯時加上 _LARGEFILE_SOURCE 與 _LARGEFILE64_SOURCE，讓編譯器知道要使用 LFS 支援。
3. 當使用底層 I/O 的 "open" 時，增加參數 "O_LARGEFILE"。

如此一來就可以順利解決 2G 大小的限制，而使用 Autoconf 語法更是方便，只要在 "configure.ac" 加入 "AC_SYS_LARGEFILE" 這一行描述就完成了。

4. i18n多語系支援

要支援多國語系主要分成兩部份，第一修改程式，第二是製作翻譯檔了。在程式支援的部份主要是：

1. 定義 setlocale，如增加程式碼 "setlocale(LC_ALL, "")"
2. 使用 gettext(String)，用以取代 "printf"。

在來就是翻譯檔了，主要流程是 source code --> .pot --> .pox --> .gmo --> .mo --> 安裝至系統。使用 xgettext 產生 .pot 檔，將 .pot 複製成 .pox，使用編輯器修改與翻譯 .pox 檔之內容，使用 msgfmt 指令將 prog.pox 編譯成 prog.gmo，最後就是我們要的訊息檔，等到我們把它安裝到 "/usr/share/locale/.../LC_MESSAGES/" 就完成多語系的支援了。

二、 2008 開發成果

1. 執行介面之改善 – 更多友善的資訊

要能讓使用者在運作過程中能提供明確的資訊細節，在基本的文字介面

下新增了 TUI(Text User Interface)的實做，透過文字與符號表達簡易圖形介面，分成 ncurses[插圖 11]與 dialog[插圖 12]。

```
lease wait...
artclone v0.0.8 (Rev:164:165M) www.partclone.org, partclone.nchc.org.tw
Starting clone device (ntfs-raw) to image (ntfs.img)
File system: NTFS
Device size: 269 MB
Space in use: 2 MB
Block size: 4096 Byte
Used block count: 441

Elapsed: 00:00:01
Remaining: 00:00:03
Rate: 24.58MB/min
```

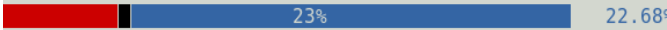


插圖 11: Ncurses 範例

```
Partclone v0.0.8 (Rev:164:165M) www.partclone.org,
partclone.nchc.org.tw
Starting clone device (ntfs-raw) to image (-)
File system: NTFS
Device size: 269 MB
Space in use: 2 MB
Block size: 4096 Byte
Used block count: 441
Syncing... OK!
Partclone successfully cloned the device (ntfs-raw) to the image
(-)
```

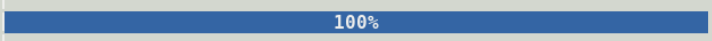


插圖 12: Dialog 範例

2. 備份壞軌硬碟 – rescue 參數

進行備份還原的硬碟，其狀態並不一定是 100%正常的，為了讓壞軌的硬碟也能儘其可能的將資料取出，需要有 rescue 參數，如同 ntfsclone 一般。根據了解 ntfsclone 的作法是將讀取失敗的區塊以"?"符號取代，如果檔案系統有修復能力時，有可能可以還原；或是當壞軌的資料不重要，透過這樣的方式，還可以把重要的資料還原的可能性大為提高。

Partclone 只要程式運作時增加參數 -R 來啟動 rescue 模式，當讀取錯誤時人嘗試讀取多次，如果仍舊失敗，則以"?"代替，並顯示警告訊息。

3. 檢查備份資料 – CRC 參數

防止記憶體或系統不穩定，導致備份資料毀損。類似 Partimage 中自行撰寫的 CheckSum 演算法，用來確認任何 I/O 的資料在一定的信心水準之下無誤，通常在記憶體毀損的情形下非常有用。

Partclone 只要程式運作時，透過 CRC 演算法[24]，將每一個 I/O 的 Block 將以 CRC 參數確認資料無誤。

4. 實驗與測試方法

由於備份與還原需要磁碟配合，如果使用操作中的磁碟，往往會造成資料毀損因此需要建立實驗磁碟環境。由於準備硬碟與磁區，往往花過多時間準備，尤其是需要測試不同的檔案系統，要花相當多的時間進行測試。因此開發中建立簡單的環境做測試是十分重要的。

在測試 Partclone 的程式時，會使用虛擬環境來進行碟備份與還原，過程如下：

- 先以"dd"指令建立磁區。以 "dd if=/dev/zero of=partition.raw bs=1M count=512" 進行，可以產生 512M 大小的虛擬磁區。
- 再將檔案轉成裝置。以 "losetup /dev/loop0 partition.raw"，將 partition.raw 視為裝置 "/dev/loop0"。
- 最後進行格式化。以 EXT2 而言，執行 "mkfs.ext2 /dev/loop0" 就產生了一個 EXT2 檔案系統磁區。

當要測時程式時，直接以 "/dev/loop0" 為一磁區進行備份與還原，不僅不會傷害本機磁區，也不會造成資料流失。即使不同的檔案系統也可以輕易建立實驗環境。

5. 達成多語系支援

目前 Partclone 也已經有中英文兩種介面，當環境變數 "LANG" 為 "zh_TW.UTF8" 時，就會直接以中文輸出。例如下例，先將環境變數轉為中文，再進行還原的動作，參考[插圖 13]。

```
root@thomas-laptop:/home/thomas/tmp# export LANG=zh_TW.UTF8
root@thomas-laptop:/home/thomas/tmp# clone.extfs -d -r -s image.e2fs.img -o /dev
/loop1
開始還原影像檔(image.e2fs.img)到裝置(/dev/loop1)
裝置大小 268435456
裝置使用到的空間 21386240
裝置區塊大小 1024
裝置使用區塊數量 20885
31.12 已經完成
```

插圖 13: 中文介面

三、 2009 開發成果

1. UFS2 檔案支援

參加 2009 LSM 研討會時，有使用者表示 FreeBSD 無法備份，主要在於檔案系統為 UFS2，經過研究與測試發現已經有比較穩定的 libufs 可以使用。唯該函式庫並沒有獨立的 package。經過開發與調整，目前已經有 package，且 partclone 也可以支援 UFS2 的檔案系統。

2. EXT4 檔案支援

隨著 e2fsprog 的更新，目前已經可以支援 EXT4 檔案系統，由於 API 使用的方式略有不同，故參考相關程式之後略做修改，即可支援 EXT4。

3. 檔案系統備份修正

隨著使用者回報，目前已經陸續修正許多檔案系統的錯誤，例如 FAT、XFS、HFS+，即便如此，未來仍需要不斷維護這些備份程式，方能確保穩定性。

4. Partclone.dd

今年首度開發類似 dd 的 Partclone。在不支援的檔案系統上，還是可以用 dd 備份。傳統的 dd 缺乏 UI 介面、CRC 檢查、預估時間統計等，因此嘗試修改 partclone 架構，製作出類似 dd 功能的 partclone。然，基本上 dd 所支援的參數如 bs、count、seek 等，partclone 還不支援。此為未來需改善之項目。

5. Partclone.restore

Partclone 還原時不需要檔案系統之參數，Clonezilla 基於此特性，整合出 on-the-fly 的方式，點對點的複製，但需要另外的程式來整合，Partclone.restore 將可以不分檔案系統，而完全還原到磁區之內。

6. Partclone.chkimg

新增檢查影像檔的功能，主要是用以檢查 CRC 與 block，而不直接寫入磁碟，這一工具將可以驗證 image 的完整性，也適合快速摹擬實際還原時可能會遇到的問題。

7. 2009 其他修正

2009 年 Clonezilla 開始預設使用 Partclone 專案備份，

1. add progress for bitmap，增加 bitmap 狀態列
2. ntfsreloc 修正，將上游程式整合到 partclone 之中
3. check filesystem before clone partition 增加備份前檢查

四、 2010 開發成果

1. Btrfs 檔案系統支援

Btrfs（通常念成 Butter FS），是由 Oracle 於 2007 年宣布並進行中的 copy-on-write 文件系統。目標是取代 Linux 目前的 ext3 文件系統，改善 ext3 的限制，特別是單個文件的大小，總文件系統大小或文件檢查和加入目前 ext3 未支持的功能，像是 writable snapshots、snapshots of snapshots、內建磁碟陣列（RAID）支持，以及 subvolumes。Btrfs 也宣稱專注在「容錯、修復及易於管理」。[25]

Partclone 根據 Btrfs 檔案系統結構與原始碼，開發了針對 Btrfs 備份的功能。唯 Btrfs 與傳檔案系統差異多了更多功能，故 Partclone 開發初期不針對 Multi device 做處理，也無法備份 Raid Btrfs，只針對單一 Partition 做 Btrfs 檔案系統備份。

Btrfs 發展至今尚無公佈可用函式庫 Library，Partclone 針對此檔案系統改以收納 Btrfs 原始碼方式，重新編譯 Partclone.btrfs。待 Btrfs 官方釋出 Public Library API 供外部程式使用，將獲得更完善的支援。

2. JFS 檔案系統支援

JFS(Journaled File System)最早是由 IBM 為 AIX 作業系統(Unix 的一支)所打造，也就是 AIX 5L 上的 JFS2(Enhanced Journaled File System)。JFS 有幾項特色：Journaling、B+ Tree 的目錄結構、壓縮以及平行的 I/O 動作。JFS 的 Journal 比較像 XFS 檔案系統，只記錄檔案的 metadata，所以並不保證資料本身。JFS 也採用 B+ Tree 來儲存目錄結構，以加速查詢的速度。在 JFS 中，檔案存放在一串可變長度的 block 中，稱之為 Extents，JFS 也將 extents 的 index 放在 B+ Tree 裡加快搜尋的速度。[26]

Partclone 也就是根據 Extents 判斷哪些 Blocks 需要備份，Extents 中的資訊與大部分 Bitmap 判斷方式一致，因此可以支援其資料備份；為其中比較需要注意者如 log 資訊的備份不在 Extents 之中，需要根據其結構另外處理，否則將成備份失敗。

3. VMFS 檔案系統支援

VMFS (Virtual Machine File System) 為 VMware ESX 的檔案系統，許多後續的進階應用例如 VMotion、HA、DRS 皆由此高效能的檔案系統做為基礎來延伸，通常儲存設備為 SAN、DAS 時便會將其格式化為 VMFS，此檔案系統最大二項特色如下：

- Cluster File System：在同一個 LUN (VMFS Volume) 上允許 多台 Host 同時 進行存取。
- On-Disk Locking：VM 只能 同時被一台 Host 控制

同時，VMFS 也支援空間擴充，稱為 Growing/Extents VMFS，也就是 LUN 容量切多大透過 Growing 機制可使 VMFS 最大容量至 2TB，舉例來說切一個 2TB 的 LUN 但一開始只使用 1TB 並格式化為 VMFS 檔案系統，隨著時間過去後空間不夠此時即可透過 Growing 機制使 VMFS 空間擴充至 2TB。

何謂 Extents VMFS？即 LUN 合併例如切了 2 個 2TB 的 LUN1 及 LUN2 後可以透過 Extents 機制將 LUN1 及 LUN2 合併成為 4TB 的 VMFS Volume，最多可以合併 32 個 LUN (每個 LUN 2TB) 也就是最大 64TB 的容量，

另外必須注意的是當您將 LUN 進行合併後之後無法再分割。整理後 VMFS (Virtual Machine File System) 對於容量的支援度如下：

- VMFS Volume 容量 最小 可支援至 1.2GB (小於此數值容量將無法建立 VMFS)
- VMFS Volume 容量 最大 可支援至 64TB (將 32 個 LUNs 每個 2TB 進行合併)[27]

Vmware 公司並沒有正式文件詳細的介紹 vmfs 檔案架構，而經查詢之後，已經有社群開發出 JAVA 版應用程式可以存取 vmfs，Christophe Fillot 與 Mike Hommey 則將其成果 porting 到 Linux 環境，並改以 C 語言重現，並加強了更多功能。Vmfs-tools[28] 正是其開發成果，經過研究認為可用來作為備份的底層函式，故重新編譯出 Library 版本，結合 Partclone 使用，開發出 partclone.vmfs。

4. XFS 檔案系統 演算法重建

XFS 檔案系統的備份還原採用的演算法與 Partimage 相同，但隨著 xfs 結構與 Library 的變更，以逐漸不適用。今年度則重新研究了 xfs 相關備份程式，主要有 xfs_copy 與 xfsdump 兩者，觀察其行為與程式原始碼，xfsdump 不使用 xfs library，且不屬於 Block 備份，雖然功能完整，但 Partclone 難以為用。反觀 xfs_copy 雖然功能簡單，但是是以 Block 為備份基礎，但是其產生的映像檔較類似 dd 的成果但是大小只有實際的使用大小。經過研究，決定採用 xfs_copy 的判斷 block 演算法，找出需要備份的 Block，並成功重寫 partclone.xfs。目前為止，xfs 依舊不打算釋出內部 Library，需要重新編譯 xfsprogs。

5. 改用GIT原始碼管理

今年轉換用 Git 為 SCM (Source Code Management) 工具。協同合作是軟體專案開發的要素，所以我們有 SCM (Source Control Management)，但要如何有效協助開發者，仍是頗大的挑戰。在分散式版本控制系統中，branch/merge 變成理所當然的行為 (核心想法)，並到處都有 repository 搭配有完整的歷史紀錄 + 本地端更動，Git 甚至指出，SCM 應該是「檔案工具」，而非限制開發者的「制度」。Git 令人驚艷的高效能，完整的工具組合也值得採用，這幾年快速發展後，也逐漸被許多世界級專案採用，Git 也是用於 Linux 核心開發的版本控制工具。與常用的版本控制工具 CVS, Subversion 等不同，它採用了分布式版本庫的方式，不必伺服器端軟體支持，使原始碼的發布和交流極其方便。

Git 的速度很快，這對於諸如 Linux kernel 這樣的大項目來說自然很重要。Git 最為出色的是它的合併跟蹤（merge tracing）能力。[29]

本專案改用 Git 同時亦獲益良多，轉換之後的優勢如下：

- 本地分支開發
- 所有開發紀錄都在本地端
- 速度快
- 體積小
- 分散式開發

也就是採用之後，Partclone 可以直接在連線的狀態下建立分支進行分支開發，新的能完成後在合併回主幹即可。Git 維護的程式碼不是只有本次原始碼本身，而是整個開發過程，讓使用者得以於離線狀態下取得所以開發與修改紀錄。Git 底層採用了壓縮的方式管理原始碼，減少程式原始碼資料庫傳送的時間與體積，如此的架構都在於優秀的主幹分支管理架構，因此相當適合分散式開發。

6. 增加French語系

今年增加了 French 語系。跨語系的支援對 Partclone 容易，本專案採用標準的 i18n 架構，只需要翻譯數行文句即可達成。因此，有賴於社群的努力，感謝翻譯者 Cedric OLLIVIER 對 Partclone 的付出，同時這也是自由軟體的精神象徵最佳寫真。

7. 新增參數

Partclone 今年新增了兩個參數，`--restore_row_file` 與 `--quiet`。`restore_row_file` 參數讓使用者可以還原映像檔到一般檔案，而不用限定在磁區裝置上。某些進階使用者，只需要處理檔案，而不需要透過裝置，某些應用上例如虛擬化也會適用，考量相關應用與彈性，加上此一參數，並於還原時補上最後一個 block 避免檔案系統錯誤。`quiet` 參數，則不會輸出 Progressbar，以無聲方式進行還原，可提高部份效率。

8. 錯誤修正

本專案今年修正了幾項較大的 Bug，列舉說明如下：

1. 單一磁區大於 8.7T 備份失敗，只要是因為記憶體不足導致，已經修正。
2. ntfs 相容性加強，由社群提供 patch，檢視後加入。
3. hfsplus 相容性加強，實做時發現 bitmap 可能之錯誤，已經修正。
4. progress bar 效率提昇。

9. Partclone 版本分支開發

本專案今年將大幅修改程式與映像檔格式，為避免錯誤，改用分支的方式開發。目前於分支中開發了：

1. 記憶體使用率提昇
2. MD5、SHA1、SHA128、SHA256、SHA512 等驗證碼
3. 調整程式架構，減少程式碼重複率約 25%，程式相對獲得穩定的效益，管理上也比較容易，避免過多重複的程式碼管理。並開發 libpartclone 內部使用。
4. 透過驗證碼實做增益笨份目前尚在開發中，為需要較多時間落實。

五、 2011年開發成果

1. 連續區塊寫入

進行 Block 讀取與寫入儲存空間時，以連續區段暫存，充份利用 cache，速度會比較快。改善之前的作法為一個一個判斷是否需要寫入，但是遇到 Block size 比較小的情形時，時常會發現效能大幅降低。經過研究調查後發現，可以利用 cache 機制提升效能。因此設定 cache 最大值，於程式中增加預先判斷機制，如果是連續驅段則加入 cache，直到 cache 空間用完或是遇到非連續驅塊就會開始將 cache 一次寫入儲存空間。

2. 審查與整合記憶體使用改善程式碼

審查由 Partclone 使用者(Andre Przywara)所開發的程式，其提供的 patch 可以減少記憶體使用的空間。Partclone 所儲存 bitmap 的資訊使用的單位

是 char, 基本上是 8bit, Andre Przywara 則將其轉換為 bit, 所以記憶體使用上減少了 7/8, 且在所儲存的硬像檔中還是維持原先格式(char), 所以可以與之前的硬像檔相容。

3. 審查與整合支援 ddrescue 區塊列表程式碼

增加選項 "--domain-log=FILE" 以支援 ddrescue 快速備份。這份 patch 是由 Ian Abbott 所提供, 他實作了讓 partclone 可以輸出一份列表, 紀錄使用到的 block 位置, 讓 ddrescue 可以採用這分清單, 進行 ddrescue 的備份工作。

4. 配合 debian 套件開發

今年與 debian package maintainer (Georges Khaznadar) 合作, 請他幫忙將 partclone 導入 debian 儲藏庫。根據 Debian Package Policy 有許多地方需要注意並且修改以符合其規範, 因此接納其建議修改了文件的格式、部份的 makefile, usage 等, 目前已經成功被收納到 Debain Repository Testing(sweezy)中。

5. 自我測試程式撰寫

debian package release 之前, 套件維護者會建議撰寫 test script, 一種類似 QA 的方式, 進行自我檢測, 確保套件維護者可以試出正確的套件。這部份目前撰寫了兩個部份, 一個是簡易自我測試, 另一個則是上游開發者測試工具。簡易自我測試能確保系統不會再測試過程中毀損, 但只能做少部份測試; 進階測試工具, 主要提供給開發者測試, 可以測試較多項目, 但也有可能不小心毀損系統。

6. 檔案系統API升級(e2fsprogs, vmfs, ntfs3g)

今年陸續調整了 3 個主要檔案系統相關的函式庫。隨著檔案系統 API 的升級與調整, partclone 也必須稍做修改, 以達到相容於新版的功能。今年主要是修改了

- e2fsprogs 升級到 1.42-WIP
- vmfs-tools 支援 vmfs5

- ntfsprog 合併到 ntfs3g

以上升級動作，其中很重要的一是還必須相容舊版，所以在程式與 makefile 上做了許多相容性調整。

7. 其他修正

本年除上述主要變化之外，也增加了許多錯誤修正、效能加強等更新，並 release 了 3 個穩定版(0.2.16, 0.2.26, 0.2.38)，與 0.2.17-0.2.39 共 13 個測試版。

伍、 Partclone 操作說明

Partclone 目前已經支援 EXT2/3, Reiserfs3, Reiser4, XFS, HFS plus, FAT16/32, NTFS 等檔案系統，且持續增加之中。檔案命名為了方便記憶採用前綴單字"partclone"，在連接要操作的檔案系統名稱，例如要備份或是還原 EXT2 檔案系同，程式名稱為 partclone.extfs，以此類推其他檔案系統，所有程式請參考[表格 1]。如此命名純粹方便記憶，系統中類似的命名有"mkfs"與"fsck"等前綴單字。

目前各程式操作選項整理如下[表格 7]，一定要給的參數有來源、目的與模式，除錯與說明則是選用參數。當來源或是目的檔案以"-"表示時，即代表標準輸入與標準輸出。

短選項	長選項	作用
-s FILE	--source FILE	來源檔案或磁碟
-o FILE	--output FILE	目的檔案或磁碟
-O FILE	--overwrite FILE	覆寫目的檔案
-D	--domain	產生 ddrescue domain file
	--offset_domain=X	設定 offset 的值
-c	--clone	備份模式
-r	--restore	還原模式
-b	--dev-to-dev	即時複製模式
-dx	--debug=x	除錯資訊(X=1, 2, 3)
-R	--rescue	嘗試備份壞軌硬碟
-C	--no-check	略過檢查磁區與剩餘空間
-N	--ncurses	文字式圖形介面
-X	--dialog	文字式圖形介面
-h	--help	說明
-L	--logfile	紀錄檔
-d	--dev-to-dev	即時複製模式
-C	--no_check	強制不檢查 size
-I	--ignore_fschk	略過檔案系統檢查
	--ignore_crc	略過 CRC 錯誤
-F	--force	錯誤不中斷
-m	--max_block_cache	block cache size
-f	--UI-fresh	更新 progress 頻率

表格 7: Partclone 參數說明

以下為一操作範例，將裝置"/dev/loop0"磁碟備份到影像檔"image.etfs.img"，指令與運作過程參考[插圖 14]。

```

thomas@thomas-laptop:~/tmp$ sudo clone.extfs -d -c -s /dev/loop0 -o image.e2fs.i
mg
Starting clone device(/dev/loop0) to image(image.e2fs.img)
The device size is 268435456
The used size is 21386240
The block size is 1024
The used block count is 20885
13.89 percent completed

```

插圖 14: 備份截圖

再提供另一個範例，以下為使用 partclone.extfs 將裝置"/dev/loop0"一邊備份同時一邊還原到裝置"/dev/loop1"，就是將備份結果輸出到標準輸出，透過 PIPE 將資料轉向標準輸入供 partclone.extfs 當作讀取來源指令與運作過程參考[插圖 15]。

```

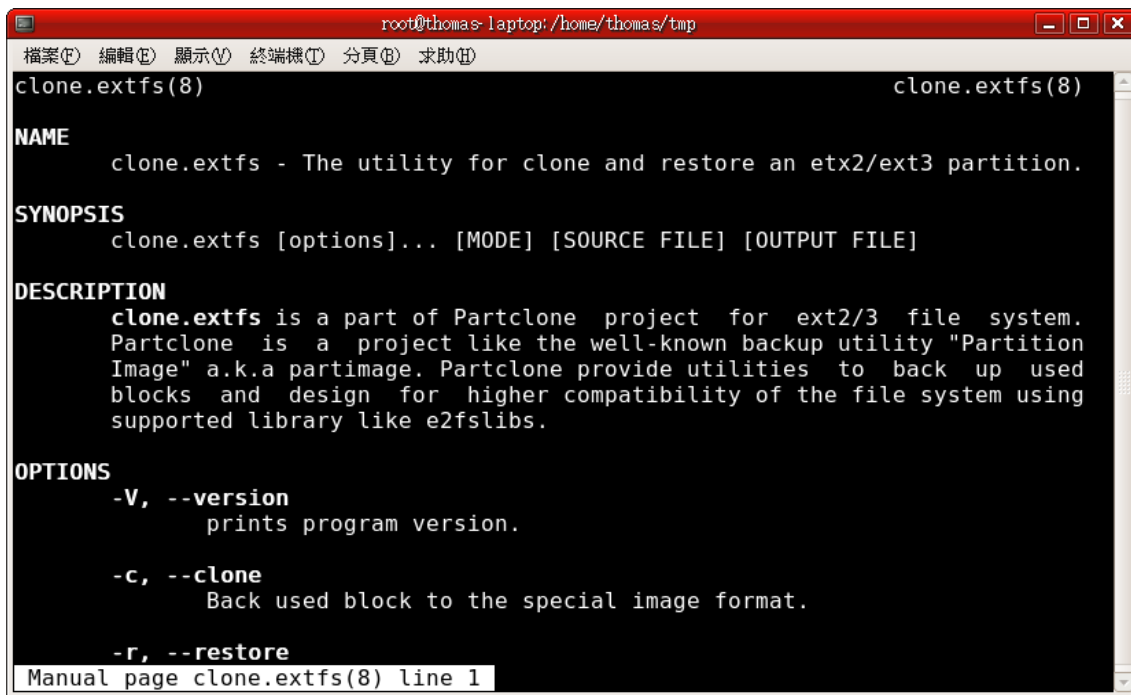
root@thomas-laptop:/home/thomas/tmp# clone.extfs -d -c -s /dev/loop0 -o - |clone
.extfs -d -r -s - -o /dev/loop1
Starting restore image(-) to device(/dev/loop1)
The device size is 268435456
The used size is 21386240
The block size is 1024
The used block count is 20885
Starting clone device(/dev/loop0) to image(-)
The device size is 268435456
The used size is 21386240
The block size is 1024
The used block count is 20885
23.94 percent completed

```

插圖 15: 標準輸出與標準輸入之運用

一、 使用手冊 - *man page*

Linux 作業系統中，大部分指令都會有說明，以 man 指令來取得該程式的資訊。Partclone 也為各個備份程式撰寫了簡單易懂的操作資訊。直接執行 "man partclone.extfs" 就可以看到相關說明，如[插圖 16]。



```
root@thomas-laptop: /home/thomas/tmp
clone.extfs(8) clone.extfs(8)
NAME
    clone.extfs - The utility for clone and restore an etx2/ext3 partition.
SYNOPSIS
    clone.extfs [options]... [MODE] [SOURCE FILE] [OUTPUT FILE]
DESCRIPTION
    clone.extfs is a part of Partclone project for ext2/3 file system.
    Partclone is a project like the well-known backup utility "Partition
    Image" a.k.a partimage. Partclone provide utilities to back up used
    blocks and design for higher compatibility of the file system using
    supported library like e2fslibs.
OPTIONS
    -V, --version
        prints program version.

    -c, --clone
        Back used block to the special image format.

    -r, --restore
Manual page clone.extfs(8) line 1
```

插圖 16: Partclone 的 manpage 說明

陸、 Partclone – 後續改進與心得

一、 2007 後續目標與改善

Partclone 目前還在開發中，雖然可以備份常用的檔案系統，但還有許多問題有待克服，描述如下：

1. 更多 File System

檔案系統的種類十分眾多，因此不斷的增加對檔案系統的支援勢在必行，尤其是要找有提供函式庫的十分不容易。

目前正開始規畫增加以下檔案系統：

UFS2：多在 FreeBSD 5-CURRENT 與 NetBSD 使用此檔案系統。

ZFS：多在 Open Solaris 所使用。

JFS：多在 HP-UX 使用此檔案系統。

程式設計之初已經將檔案系統與主程式徹底分離，新增檔案系統十分容

易。但是要找到穩定的檔案系統函式庫才是困難之所在。

2. 執行介面之改善 – 更多友善的資訊

既有介面中的資訊仍是不足，備份中的資訊應該比照 Partimage 提供更多細節與進度給使用者，數字的表現應該更加容易讀取。

此外，目前只有純文字介面，將來可望發展好用的圖形介面。但圖形介面需求其實不高，Partclone 的目的比較單純且容易使用，所以指令參數很少，切要搭配 Shell Script 更能發揮其效能。而圖形介面受限於操作環境，因此若不需要可以不做。

不論是圖形與文字介面，只要能讓使用者方便操作、過程中能提供明確的資訊細節，還有更多可以發揮之處。

3. 備份壞軌硬碟 – rescue 參數

進行備份還原的硬碟，其狀態並不一定是 100% 正常的，為了讓壞軌的硬碟也能儘其可能的將資料取出，需要有 rescue 參數，如同 ntfsclone 一般。根據了解 ntfsclone 的作法是將讀取失敗的區塊以 "?" 符號取代，如果檔案系統有修復能力時，有可能可以還原；或是當壞軌的資料不重要，透過這樣的方式，還可以把重要的資料還原的可能性大為提高。

4. 檢查備份資料 – CheckSum 參數

防止記憶體或系統不穩定，導致備份資料毀損。類似 Partimage 中自行撰寫的 CheckSum 演算法，用來確認任何 I/O 的資料在一定的信心水準之下無誤，通常在記憶體毀損的情形下非常有用。

二、 2008 後續目標與改善

1. Partdiff 與 Partpatch

最常被使用者問道的問題就是遞增備份。但由於 Partclone 所備份的對象是磁塊(Block)，而不是檔案，故無法象 rsync 等方式作遞增備份。但由於 Partclone 運作過程中強制對每個 Block 進行 CRC 演算[24]，同時將其結果保留起來，將來可用以判斷該 Block 是否異動，若有異動則備份；否則就跳過等方

式來嘗試實現遞增備份的功能。

2. Partclone.restore

計畫撰寫新功能可以不考慮 Image 是哪種檔案系統，通通可以直接還原。目前 Partclone 所使用的方式都是預設知道備份或還原的對象的檔案系統，而實際使用時卻往往不見得知道要還原的 Image 是哪個檔案系統，尤其以 pipe 方式運作時尤造成不小的影響。故計畫開發新程式用以還原 Image 但是玩全不需要知道檔案系統。

3. Partclone.dd

嘗試將 "dd" 加上簡單的 TUI 介面。預計會使用並修改程式 "dd" 成為 partclone.dd，讓使用者在使用 dd 的同時，可以有更多有用的資訊可以參考。

4. Partclone.zfs、Partclone.btrfs

檔案系統不斷的更新與進步，Partclone 也必須跟上腳步，只要有人使用的檔案系統，應該都要有方便的程式可以協助備份與還原。ZFS 與 BTRFS 都是 Partclone 主要支援的對象。

三、 2009 後續目標與改善

1. Partclone.zfs、Partclone.btrfs

檔案系統不斷的更新與進步，Partclone 也必須跟上腳步，只要有人使用的檔案系統，應該都要有方便的程式可以協助備份與還原。ZFS 與 BTRFS 都是 Partclone 主要支援的對象。

今年 BTRFS 的格式比較穩定，kernel 也正式支援，應可順利完成於 2009 年中。

2. 改善記憶體使用

隨著硬碟容量的增大，發現偶而會有記憶體不足之現象，為了增加記憶體的使用效益，計劃調整記憶體使用之架構。除了系統架構需要調整也將改善印象檔之格式，方可有效改善記憶體取用之效益。

3. 開發 JFS 檔案系統

明年規劃開始開發 JFS 檔案系統，將以類似 XFS 處理的方式直接從 partimage 移轉。

4. 增加 md5、sha1、adler32 等驗證方式

嘗試增加更多驗證方式，讓使用者自行選擇。

5. 使用手冊撰寫

重新設計與撰寫使用手冊，結合 WIKI 與 manual 讓操作方式更簡結易懂。

四、 2010 後續目標與改善

1. 增益備份

今年以取得 MD5、SHA128、SHA256、SHA512 關鍵技術，可透過 Library 達成，將夠過此技術進一步開發為增益備份。程式將比對應像檔中的 Block 驗證碼，找出相異的部份並且備份。

2. ExFAT 檔案系統支援

exFAT (Extended File Allocation Table, 又名 FAT64) 是一種特別適合於閃存驅動器的檔案系統，最先從微軟的 Windows Embedded CE 6.0 導入這種檔案系統，再延伸到後來的 Windows Vista Service Pack 1 桌面作業系統中。由於 NTFS 檔案系統的一些數據格式規定所限，對閃存驅動器而言 exFAT 檔案系統顯得更見優勢。

Partclone 目前不支援 ExFAT，而已經由 Clonezilla Source Forge 獲得使用者這方面的需求，
<https://sourceforge.net/projects/clonezilla/forums/forum/663168/topic/3860495>。經過研究後，了解 Exfat 實做上需要的資源 Free exFAT file system implementation，來自 <http://code.google.com/p/exfat/source/checkout>，研究後認為可行，並於明年開始實做。

3. ZFS檔案系統支援

ZFS 源自於 Sun Microsystems 為 Solaris 作業系統開發的文件系統。ZFS 是一個具有高存儲容量、文件系統與卷管理概念整合、嶄新的磁碟邏輯結構的輕量級文件系統，同時也是一個便捷的存儲池管理系統。ZFS 是一個使用 CDDL 協議條款授權的開源項目。不同於傳統文件系統需要駐留於單獨設備或者需要一個卷管理系統去使用一個以上的設備，ZFS 建立在虛擬的，被稱為「zpool」的存儲池之上。每個存儲池由若干虛擬設備（*virtual devices*，*vdevs*）組成。這些虛擬設備可以是原始磁碟，也可能是一個 RAID1 鏡像設備，或是非標準 RAID 等級的多磁碟組。於是 zpool 上的文件系統可以使用這些虛擬設備的總存儲容量。可以使用磁碟限額以及設置磁碟預留空間來限制存儲池中單個文件系統所佔用的空間。

Partclone 將此取類似 zfsclone 的相同演算法實做 partclone.zfs。

4. 增加partclone.mount

Partclone 所產生的映像檔無法直接存取其中的內容，網路上已有人實做出工具可以做到。明年將整合該程式於主體中，並修正相關 bug。

5. 增加GUI

目前許多程式大多透過 Clonezilla 這類程式整合，難以接近一般使用者，為了突破此一限制，將開始進行 GUI 設計與開發。目前尚在研究 Gtk、Qt、Wx 等工具，期未來能有人性化的使用者介面讓一般使用者使用。

6. 多裝置一次寫入

取法 xfs_copy 的作法，利用 Multi thread 將要被還原的資料，同時寫到多個裝置。今年有許多需求式與此類似的，例如 usb 隨身碟得大量複製等。

五、 2011 後續目標與改善

1. 調整架構與釋出Partclone API

將計劃重新規劃 partclone 內部程式碼，朝向容易管理、模組化、分工等細節進行調整。同時以開發 API 為目標，將 Partclone 寫成 library，內部可以以呼叫 library 方式運作，同時也分階段 release 出 API 給社群使用。

2. 架構改寫支援 multi-thread

觀察許多相似功能的程式，大多包含有 multithread 的功能，Partclone 為了提升效能，也將嘗試改寫加入 multithread 功能，將 crc 計算、input、output、multioutput 等各自獨立，以達到效能最佳化。

3. NILFS 支援

NILFS 是相當有趣的檔案系統，其最大的特色是可管理檔案鏡攝系統。檔案系統運作時會儘量保留大部份有資料的 block，除非空間不夠才會覆寫資料到舊的 block。同時也保留了這些就有資料的相依性，使得該檔案系統可以輕易的製作檢查點(check point)與鏡攝點(snapshot)。其中最特殊之處在於，檔案系統上線時，可以即時另外重複呼叫舊的檢查點(check point)與鏡攝點(snapshot)來取得舊的資料。Partclone 針對此部份將嘗試透過其內部的 API 進行備份，此功能之實作也出現再 btrfs, zfs 等檔案系統。

4. 減少 seeking 時間

目前 partclone seeking 就是 上一個 block 寫入完成了，下一個是不需要，續的下下一個不需要一直又遇到真正要寫入的 block，才會真正的 seek 硬碟指標到該位置寫入資料！同時 seeking 會不知道什麼時候結束部份檔案系統，時常會寫少數資料放在磁區最後，所以會看到前面完成了但只差一兩個 block，所以需要 seeking 一段時間。日後規劃把 bitmap 從 array 改成 struct array, 像 tree 一樣，設計為直接知道下一個 block 的方式進行，可減少 seeking 的時間。

六、 Partclone – 開發心得

在 Partclone 的開發過程中，陸續累積不少開發上的經驗。對 C 語言的熟悉度、SVN 版本控制的使用、利用 TRAC 系統的管理功能、deb 製作套件與 Autotools 的使

用，都有不少學習上的價值。更重要的是，不僅只是學習還有實際操作的機會，使學習結果更加印象深刻。

除了開發上的技巧外，對磁區的結構也有所認識。以 HFS plus 而言，因為函式庫的問題，需要了解磁區結構之外，還需要自行撰寫程式來驗證所知，這樣的經驗使自己對磁區結構的分析更有經驗，未來在磁區相關除錯的技巧上與新增磁區的應用等皆會有所幫助。

往後的開發中，在增加檔案系統支援時，會對檔案系統結構加以研究，不僅僅只是使用檔案系統函式，而更能了解函式背後的意義將是下一階段十分重要的學習歷程。自由軟體的開發成果，不僅只有產品本身的價值。開發過程中所接觸到的自由軟體開發工具、團隊合作態度的培養、溝通技巧、對程式設計的訓練與要求與公開程式碼的挑戰等，都是對程式設計師的成長非常有幫助的。

柒、 文獻目錄

- 1: Francois Dupoux, <http://www.partimage.org>。
- 2: Steven Hsiau, <http://clonezilla.nchc.org.tw>。
- 3: Anton Altaparmakov, <http://www.linux-ntfs.org>。
- 4: Francois Dupoux, <http://www.fsarchiver.org>。
- 5: Silicon Graphics Inc., http://xfs.org/docs/xfsdocs-xml-dev/XFS_User_Guide/tmp/en-US/html/index.html。
- 6: xfs, http://linux.die.net/man/8/xfs_copy。
- 7: Sun, <http://dlc.sun.com/pdf/819-5461/819-5461.pdf>。
- 8: Tytso, <http://e2fsprogs.sourceforge.net/>。
- 9: Yury Umanets, <http://reiserfs.linux.kiev.ua/>。
- 10: Hans Reiser, <http://www.namesys.com/>。
- 11: Ty Coon, <http://oss.sgi.com/projects/xfs/>。
- 12: Klaus Halfmann, <http://www.penguinppc.org/historical/hfsplus/>。
- 13: Apple Inc, <http://developer.apple.com/mac/library/technotes/tn/tn1150.html#VolumeHeader>。
- 14: wikipwdia, http://en.wikipedia.org/wiki/File_Allocation_Table。
- 15: Richard Stallman, <http://gcc.gnu.org/>。
- 16: GNU, <http://developer.gnome.org/tools/build.html>。
- 17: GNU, <http://www.gnu.org/software/autoconf/>。
- 18: GNU, <http://www.gnu.org/software/automake/>。
- 19: Osamu Aoki, <http://www.debian.org/doc/manuals/maint-guide/index.en.html>。

- 20: Edgewal.Inc, <http://trac.edgewall.org/>。
- 21: CVS, http://en.wikipedia.org/wiki/Concurrent_Versions_System。
- 22: SVN, http://en.wikipedia.org/wiki/Subversion_%28software%29。
- 23: Andreas Jaeger, http://www.suse.de/~aj/linux_lfs.html。
- 24: Lammert Bies, http://www.lammertbies.nl/comm/info/nl_crc-calculation.html。
- 25: btrfs wiki, https://btrfs.wiki.kernel.org/index.php/Main_Page。
- 26: Steve Best, Dave Kleikamp, How the Journaled File System handles the on-disk layout, 2000
- 27: Wikipedia, http://en.wikipedia.org/wiki/VMware_VMFS。
- 28: Christophe Fillot, Mike Hommey, <http://glandium.org/projects/vmfs-tools/>。
- 29: Junio C Hamano, <http://git-scm.com/>。