SquashFS HOWTO

Artemiy I. Pavlov

Sinevibes (http://www.sinevibes.com)

Marco Cecchetti

mrc (dot) ildp (at) gmail (dot) com

Diario delle Revisioni

Revisione 1.8 06/01/2008 Revisionato da:
Cambiamenti per la versione 3.3 di squashfs. Aggiunta di alcune parti.
Revisione 1.7 25/03/2005 Revisionato da:
Cambiamenti per la versione 2.1 di squashfs.
Revisione 1.6 10/11/2004 Revisionato da:
Cambiamenti per la versione 2.0 di squashfs. Correzioni nel testo.
Revisione 1.5 07/06/2004 Revisionato da:
Cambiamenti per la versione 2.0 alpha di squashfs. Chiarimenti e molte descrizioni migliorate. Instruzioni se
Revisione 1.1 22/05/2004 Revisionato da:
Cambiamenti per la versione 1.3r3 di squashfs.
Revisione 1.0 19/02/2004 Revisionato da:
Rilascio iniziale, revisionato da LDP.
Revisione 0.2 08/12/2003 Revisionato da:
Correzioni nel testo. Aggiunta la Licenza
Revisione 0.1 24/11/2003 Revisionato da:
Versione iniziale. Instruzioni per la versione 1.3r2 di squashfs.

Abstract

Questo HOWTO descrive l'uso di SquashFS, un file system ad alta compressione in sola lettura per Linux, utile per sistemi operativi minimimali ed integrati e in ogni situazione in cui si voglia utilizzare un file system compresso. Con questo documento si apprenderà come preparare un kernel Linux che gestisca SquashFS, creare un file system di tipo SquashFS ed utilizzarlo senza intoppi.

Home page di questo HOWTO

L'howto [in inglese] di SquashFS è disponibile all'indirizzo: http://artemio.net/projects/linuxdoc/squashfs. dove è possibile trovare l'ultima versione del documento ed inviare il proprio feedback.

Traduzione a cura di Marco Cecchetti. Revisione di Antonio Colombo.

1. Cos'è SquashFS

1.1. Introduzione

Nella preparazione di sistemi operativi Linux minimali ed integrati ogni byte del dispositivo di memorizzazione (floppy, flash disk, ecc.) è molto importante, quindi la compressione viene usata ovunque possibile. Inoltre, file system compressi sono di solito necessari ai fini dell'archiviazione. Ciò è essenziale per i grandi archivi pubblici o anche per archivi personali.

Il file system SquashFS porta tutto ciò ad un nuovo livello. Si tratta di un file system in sola lettura che consente la compressione di interi file system o singole directory, la loro scrittura su altri dispositivi/partizioni o su normali file ed il mount diretto (se si tratta di una partizione) o mediante l'utilizzo del dispositivo di loopback (se si tratta di un file). La struttura modulare e compatta di SquashFS è davvero brillante. Ai fini dell'archiviazione, SquashFS è molto più flessibile e veloce di un archivio tarball.

Il software di SquashFS distribuito comprende una patch per i sorgenti del kernel Linux (che abilita nel kernel il supporto in lettura del file system SquashFS), lo strumento **mksquashfs**, che serve per creare file system compressi e lo strumento **unsquashfs**, che permette di estrarre file multipli da un file system SquashFS esistente.

L'ultima versione rilasciata di SquashFS è la 3.x, la precedente era la versione 2.x. Questo documento descrive entrambe le versioni, con relative note. Ad esempio, se qualche caratteristica o parametro è diversa per queste versioni, verrà indicato come segue: *nuovo valore (3.x) oppure vecchio valore (2.x).*

1.2. Panoramica su SquashFS

- Dati, inode e directory vengono compressi
- SquashFS memorizza per intero uid/gid (32 bit) e la data di creazione dei file
- Sono supportati file di dimensioni fino a 2^64 byte; i file system possono essere estesi fino a 2^64 byte
- Inode e directory sono altamente compressi e memorizzati con la granularità di un byte. Ogni inode compresso ha una dimensione media di 8 byte (la dimensione esatta varia in funzione del tipo di file; cioè inode di file semplice, directory, collegamento simbolico, dispostivo a blocchi ed a caratteri hanno dimensioni differenti).
- Il file system SquashFS supporta dimensioni di blocco fino a 64 kb (2.x) o 1MB (3.x). La dimensione di default dei blocchi è 128 kb (3.x), che consente di ottenere rapporti di compressione maggiori della normale dimensione di blocco di 4kb.

- Dalla versione 2.x è stato introdotto il concetto di *frammento di blocco*, che implementa la possibilità di unire più file di dimensioni inferiori alla dimensione di blocco in un singolo blocco, ottenendo rapporti di compressione più alti.
- File duplicati sono individuati e rimossi
- Entrambe le architetture big e little endian sono supportate; SquashFS può montare file system creati su macchine con diverso ordine di byte.

1.3. Per fare chiarezza

Ora ci si assicurerà che ogni ulteriore discussione risulti più chiara da comprendere. La procedura per rendere disponibile SquashFS fondamentalmente consiste nei seguenti passaggi:

1. Applicare la patch e ricompilare il kernel per attivare il supporto di squashfs
2. Compilare lo strumento **mksquashfs** e **unsquashfs**
3. Creare un file system compresso con **mksquashfs**
4. Test: montare il file system squashfs sotto un punto di mount temporaneo.
5. Modificare il file `/etc/fstab` o gli script di avvio del sistema Linux per montare il nuovo file system SquashFS quando necessario.

2. Preparativi per SquashFS

2.1. Ottenere SquashFS

La pagina web di SquashFS è all'indirizzo http://squashfs.sourceforge.net/ - Si tratta di un changelog, che contiene le informazioni sull'ultimo rilascio del software ed informazioni generali su SquashFS. E' possibile ottenere l'ultima versione di SquashFS alla pagina web del progetto (http://sourceforge.net/project/showfiles.php?group_id=63835) SquashFS su SourceForge.

SquashFS è anche disponibile con l'algoritmo di compressione *Lempel-Ziv-Markov* (LZMA) all'indirizzo http://www.squashfs-lzma.org/

2.2. Preparazione di un kernel che supporta SquashFS

Per la lettura del file system SquashFS, è necessario che esso sia supportato dal kernel - proprio come si trattasse di un file system `reiserfs` o `ext3`. Bisogna assicurarsi che vi sia una patch appropriata per la versione del kernel che si intende utilizzare nella sotto-directory `kernel-patches/linux-2.x.y`, all'interno della directory dei sorgenti di SquashFS. Inoltre, si ricordi che, nella maggior parte dei casi, saranno necessari i sorgenti del kernel Linux *originale*, a cui non siano state applicate patch, rilasciato da kernel.org (http://kernel.org/). Se il codice sorgente del proprio kernel proviene da una distribuzione, potrebbe essere già stato modificato con le patch proprie della distribuzione e l'applicazione della patch di SquashFS quasi certamente non potrà funzionare, dato che le patch di SquashFS sono fatte per i kernel Linux *originali*. Tuttavia alcune distribuzioni rendono disponibili i moduli del kernel per SquashFS ed i relativi strumenti sotto forma di pacchetto nei repository. Se si pensa di utilizzare tali pacchetti, non sarà necessario applicare la patch di SquashFS al kernel e neppure compilare gli strumenti a partire dal codice sorgente. In tal caso, è comunque necessario assicurarsi di acquisire dalla distribuzione il modulo del kernel appropriato alla propria architettura. Si noti che, in questo modo, la procedura d'installazione è più semplice, ma non si ha alcun controllo sulla configurazione dei parametri del kernel, nel caso si voglia utilizzare SquashFS per scopi particolari (ad esempio per sistemi integrati).

2.2.1. Applicare la patch ai sorgenti del kernel

Ottenuto il codice sorgente del kernel e la relativa patch per SquashFS, (si assuma che i sorgenti del kernel siano collocati nella directory `/usr/src/linux` e che il codice sorgente di SquashFS sia in `/usr/src/SquashFS`) tutto ciò che si deve fare è:

Portarsi nella directory dei sorgenti di SquashFS e copiare la patch per il kernel (assumendo che il nome del file sia `SquashFS-patch`) in `/usr/src/linux`.

```
bash# cd /usr/src/SquashFS
bash# cp linux-2.x.y/SquashFS-patch /usr/src/linux
```

Portarsi nella directory dei sorgenti del kernel `/usr/src/linux`:

```
bash# cd /usr/src/linux
```

Nota: Le ulteriori procedure relative alla preparazione del kernel saranno tutte svolte all'interno di questa directory. I percorsi saranno quindi espressi relativamente a `/usr/src/linux`.

Ora, applicare la patch di SquashFS al codice sorgente del kernel:

```
bash# patch -p1 < SquashFS-patch
```

2.2.2. Compilazione per un kernel 2.6.x

Pulire e preparare i sorgenti del kernel:

```
bash# make distclean
bash# make mrproper
```

Configurare il kernel utilizzando il proprio metodo preferito (config/menuconfig/xconfig/gconfig):

```
bash# make menuconfig
```

1. Nella sezione“*File systems*”, sotto-sezione “*Miscellaneous file systems*”, abilitare l’opzione “*Squashed file system*”, sotto forma di modulo o compilato nel kernel. E’ obbligatorio compilare il supporto di SquashFS all’interno del kernel solo se si intende utilizzare ramdisk iniziali (**initrd**) compressi con SquashFS.
2. Nella stessa sotto-sezione **NON** si abiliti l’opzione “*Additional option for memory-constrained system*”, a meno che non si stia configurando un kernel per un sistema integrato.
3. Nel caso si voglia utilizzare un ramdisk iniziale compresso con SquashFS, selezionare l’opzione “*Initial RAM disk support*” nella sotto-sezione “*Block devices*” della sezione “*Device drivers*”.
4. Se si vuole eseguire il mount del file system SquashFS attraverso un *dispositivo di loopback*, abilitare il supporto “*Loopback device support*” nella sotto-sezione “*Block devices*” della sezione “*Device drivers*”

Ora si potrà compilare il kernel ed i moduli:

```
bash# make
```

2.2.3. Compilazione per un kernel 2.4.x

Configurare il kernel:

```
bash# make menuconfig
```

1. Nella sezione “*File systems*” abilitare l’opzione “*Squashed filesystem*”, sotto forma di modulo o compilandone il supporto nel kernel. E’ obbligatorio compilare SquashFS all’interno del kernel solo se si intende utilizzare ramdisk inziali (**initrd**).
2. Nel caso si voglia utilizzare un ramdisk inziale, abilitare “*Initial RAM disk support*” nella sezione “*Block devices*”.
3. Se si vuole eseguire il mount del file system SquashFS attraverso un *dispositivo di loopback*, abilitare “ *Loopback device support* ” nella sezione “*Block devices* ”.

Ora è possibile compilare il kernel ed i moduli:

```
bash# make dep
bash# make bzImage
bash# make modules
```

2.2.4. Installazione e test del kernel

E' giunto il momento di installare il nuovo kernel con il supporto di SquashFS. Le istruzioni sotto riportate serviranno ad installare ed avviare il kernel sulla macchina host. Si potrà volerlo installare e testare sul sistema di destinazione.

Si assume che il kernel sia stato compilato per una architettura x86, e che l'immagine compressa del kernel sia collocata nella sotto-directory `arch/i386/boot/` della directory dei sorgenti del kernel. Ora, copiare il kernel nella directory `/boot` (se si vuole, rinominandolo `bzImage-sqsh` per convenienza):

```
bash# cp arch/i386/boot/bzImage /boot/bzImage-sqsh
```

Non dimenticare di installare i moduli del kernel se presenti:

```
bash# make modules_install
```

Modificare il file di configurazione del boot loader per includere il nuovo kernel ed installare (aggiornare) il boot loader. Ora è possibile riavviare con il nuovo kernel. Al riavvio verificare che tutto sia andato bene:

```
bash# cat /proc/filesystems
```

Oppure, se il supporto per SquashFS è stato compilato come modulo del kernel:

```
bash# insmod SquashFS
bash# cat /proc/filesystems
```

Se è visibile la linea di `SquashFS` tra altri file system, significa che SquashFS è supportato correttamente dal proprio kernel.

2.3. Compilare gli strumenti di SquashFS

Ora sarà necessario compilare **mksquashfs** - lo strumento per creare file system compressi e **unsquashfs**, che serve ad estrarre file da un file system SquashFS esistente.

```
bash# cd /usr/src/SquashFS/SquashFS-tools
```

Compilare ed installare gli strumenti:

```
bash# make
bash# cp mkSquashFS /usr/sbin
bash# cp unSquashFS /usr/sbin
```

Se tutto è andato bene, digitando **mksquashfs** oppure **unsquashfs** al prompt della shell, dovrebbe essere mostrato il messaggio di utilizzo.

2.4. Installare SquashFS su Debian

Se si utilizza Debian (o un'altra distribuzione Linux) è possibile ottenere il modulo di SquashFS ed i relativi tool dai repository. Con Debian, si devono installare il modulo appropriato del kernel e gli strumenti di SquashFS con i seguenti comandi:

(Assumendo che la propria architettura sia x86)

```
bash# apt-get install SquashFS-modules-2.6-486

bash# apt-get install SquashFS-tools
```

Ora, caricare il modulo di SquashFS per il kernel Linux e se caricato in modo corretto, si dovrebbe poterlo trovare nel elenco relativo

```
bash# modprobe SquashFS

bash# lsmod|grep squash
squashfs              39620  0
```

Quindi, se è necessario caricare il modulo di SquashFS all'avvio del sistema, aggiungere la relativa voce al file /etc/modules

```
bash# echo squashfs >> /etc/modules
```

Notare che al momento della scrittura di questo HOWTO, i pacchetti Debian (Etch. 4.0 r2) sono relativi al rilascio 3.1 di SquashFS. Alcune opzioni e caratteristiche recenti della versione 3.3 del software possono non essere supporate. Si veda la sezione seguente per approfondimenti.

3. Spiegazione degli strumenti di SquashFS

3.1. Utilizzo di mksquashfs

Il programma **mksquashfs** è lo strumento che serve a creare nuovi file system compressi e ad accodare nuovi dati a quelli esistenti. Il formato generale della riga di comando per **mksquashfs** è:

```
bash# mksquashfs sorgente1 sorgente2 ... destinazione [opzioni]
```

- `sorgente1`, `sorgente2`, ecc.: file o directory da aggiungere al file system risultante, indicati con percorsi assoluti o relativi.
- `destinazione`: un file regolare (file-immagine di un filesystem) o un dispositivo a blocchi (come `/dev/fd0` o `/dev/hda3`) dove si voglia creare un file system SquashFS.

Note sul comportamento predefinito di **mksquashfs**:

- Quando nuovi file sono aggiunti ad un nuovo file system o accodati ad un file system esistente, **mksquashfs** rinomina automaticamente i file che hanno lo stesso nome: se due o più file con nome `text` compaiono nella stessa directory finale, il secondo file verrà rinominato `text_1`, il terzo `text_2` e così via.
- I file duplicati vengono rimossi, così da avere una sola istanza fisica. (Dalla versione SquashFS 2.x, è possibile disabilitare l'individuazione/rimozione dei file duplicati con l'opzione **-no-duplicates**).
- Se la *destinazione* è un file system SquashFS pre-esistente, per default, i nuovi elementi *sorgente* vengono accodati alla directory root esistente. Consultare la tabella delle opzioni che segue per richiedere a **mksquashfs** di sovrascrivere interamente la destinazione o cambiare la modalità in cui elementi nuovi vengono aggiunti.
- Se viene indicata una singola directory o file sorgente, essa diviene la root del nuovo file system creato. Se vengono indicati due o più file o directory, essi divengono sotto-directory della root del nuovo file system.
- Il file system risultante viene arrotondato per eccesso ad un valore multiplo di 4 Kb: ciò è richiesto per file system da utilizzare su dispositivi a blocchi. Nel caso si sia certi di non averne bisogno, usare l'opzione **-nopad** per disabilitare l'operazione.

Si veda la prossima sezione maggiori dettagli su tutte le opzioni possibili.

3.2. Opzioni per la riga di comando

Tutte le opzioni possibili di **mksquashfs** sono mostrate nella tabella sotto.

Tabella 1. Opzioni per la riga di comando dello strumento mksquashfs

Opzioni	Descrizione
-2.0	chiede a **mksquashfs** di creare un file system per la versione 2.0
-all-root o **-root-owned**	rende tutti i file sul file system di destinazione proprietà dell'utente root (UID=0, GID=0)
-always-use-fragments	divide tutti i file più grandi della dimensione di un blocco in frammenti di blocco (dalla versione 2.x). Ciò rende i rapporti di compressione più elevati

Opzioni	Descrizione
-b [block size]	usa un blocco di dimensione [block size] per il file system (il valore predefinito per la versione 2.x è 32Kb, quello della versione 3.x, 128Kb) - può assumere i valori 4096, 8192, 16384, 32768, 65536 or 131072 byte
-be o **-le**	chiede la creazione di un file system rispettivamente big endian o little endian
-check-data	abilita verifiche ulteriori sul file system
-e [file1] ([file2] ...)	specifica quali file e/o directory saranno omesse dal nuovo file system che viene creato
-ef [file]	specifica un `file` che contenga la lista di file/directory da escludere
-force-gid [GID]	imposta l'identificativo di gruppo [GID] per tutti i file del file system di destinazione (si può specificare come nome o come numero)
-force-uid [UID]	imposta l'identificativo utente [UID] per tutti i file del file system di destinazione (si può specificare come nome o come numero)
-info	stampa i nomi dei file, la loro dimensione originale ed il rapporto di compressione mentre vengono aggiunti al file system
-keep-as-directory	se la sorgente è una singola directory, essa sarà una sotto-directory della root nel file system creato
-noappend	se il file o il dispositivo di destinazione contiene già un file system compresso, lo sovrascrive, piuttosto che accodare i nuovi dati al file system esistente
-no-duplicates	non individua/rimuove i file duplicati
-noD o **-noDataCompression**	non comprime i dati
-noF o **-noFragmentCompression**	non comprime i frammenti di blocco (disponibile a partire dalla versione 2.x)
-no-fragments	non genera frammenti di blocco (disponibile a partire dalla versione 2.x; ciò produce un file system simile a quello creato dalla versione 1.x)
-noI o **-noInodeCompression**	non comprime la tavola degli inode
-nopad	non arrotonda la dimensione del file system finale ad un valore multiplo di 4 Kb
-root-becomes [name]	questa opzione può essere utilizzata quando si accodano dati ad un file system compresso pre-esistente; ciò produce una nuova root, e una directory [name] che contiene tutti i file e le directory pre-esistenti.

Opzioni	Descrizione
-version	stampa un messaggio con versione, copyright e licenza
-recover [name]	ripristina i dati del file system utilizzando il file di ripristino [name] (3.3)
-no-recovery	non crea file di ripristino (3.3).
-no-exports	non consente l'esportazione del file system mediante NFS (3.x)
-no-sparse	non esegue il controllo per file sparsi (3.x)
-processors [number]	imposta il numero delle CPU per creare il file system. Il comportamento predefinito è quello di utilizzare tutti i processori disponibili (3.x)
--read-queue [size]	imposta la coda di input a [size] Mb. (Il valore predefinito è 64 Mb)(3.x)
-write-queue [size]	imposta la coda di output a [size] Mb (3.x)
-sort [sort_file]	ordina i file in funzione della priorità nel [sort_file] (3.x)
-wildcards	abilita i metacaratteri estesi della shell per escludere directory/file (da usare con l'opzione -e)
-regex	abilita l'utilizzo delle espressioni regolari POSIX (3.3)

Nella maggior parte dei casi, le opzioni predefinite per compressione e blocchi, permettono a **mksquashfs** di ottenere il miglior rapporto di compressione possibile.

3.3. Utilizzo di unsquashfs

Il programma **unsquashfs** è lo strumento che serve ad estrarre dati dal file system compresso. Il forma generale della riga di comando per **unsquashfs** è:

```
unsquashfs [opzioni] target [file/directory da estrarre]
```

- target è il file system compresso da estrarre.

Note per il comportamento di **unsquashfs**:

- Non specificando alcun *percorso di destinazione*, unsquashfs estrae il file system compresso nella directory *./squashfs-root* .
- Lo strumento non estrae un file system compresso su una directory già esistente a meno che non venga specificata l'opzione **-f**.

- Si può specificare sulla riga di comando, un numero a piacere di file o directory da estrarre e gli oggetti da estrarre possono essere anche indicati in file con l'opzione **-e [file]**.

Le possibili opzioni per **unsquashfs** sono mostrate nella tabella sotto.

Tabella 2. Opzioni per la riga di comando dello strumento unsquashfs

Opzione	Descrizione
-v[ersion]	stampa un messaggio con versione, copyright e licenza.
-i[nfo]	stampa i nomi dei file mentre vengono estratti dal file system
-l[ist]	elenca il contenuto del file system compresso senza estrarre i file
-li	elenca i file con i loro attributi mentre vengono estratti (3.3)
-ll	elenca i file del file system compresso senza eseguire estrazione (3.3)
-d[estination] path	specifica un percorso di destinazione per gli oggetti estratti
-f[orce]	se i file esistono li sovrascrive
-s[tat]	mostra le informazioni sul superblocco del file system (può individuare la versione del file system e le opzioni usate per comprimerlo - 3.3)
-e[f] [extract file]	elenca le directory e i file da estrarre (le voci vanno date una per linea) (3.3)
-r[egex]	utilizza epressioni regolari POSIX per gli oggetti da estrarre (3.3)

Nota che a partire dal rilascio 3.x è possibile estrarre anche file system creati con la versione 1.x o 2.x di mksquashfs.

4. Creare ed utilizzare file system compressi

4.1. Passi base

Per creare un file system compresso a partire da una singola directory (si assuma, `/some/dir`), e scriverlo su un file regolare (producendo cosÃ¬ un file-immagine del file system), basta dire una sola frase magica:

```
bash# mksquashfs /some/dir dir.sqsh
```

Il programma **mksquashfs** eseguirÃ Â la compressione, stamperÃ il numero di inode, le dimensioni dei dati scritti e il rapporto di compressione medio. Ora, si avrÃ la directory `/some/dir` sul file-immagine `dir.sqsh`, e si potrÃ quindi usare il comando **mount** per montarlo utilizzando un dispositivo di loopback:

```
bash# mkdir /mnt/dir
bash# mount dir.sqsh /mnt/dir -t squashfs -o loop
```

Per verificare che si abbia quanto atteso:

```
bash# ls /mnt/dir
```

Se si vuole scrivere il file system direttamente su un dispositivo (ad esempio il proprio floppy su `/dev/fd0`):

```
bash# mksquashfs /some/dir /dev/fd0
```

Eseguire poi il **mount** del dispositivo:

```
bash# mount /dev/fd0 /mnt/floppy -t squashfs
```

Quindi verificare se tutto Ã¨ andato a buon fine:

```
bash# ls /mnt/floppy
```

4.2. Compressione dei file system

Le operazioni qui descritte corrispondono alla maggior parte dei casi di utilizzo di un file system compresso in sola lettura, sia che si voglia utilizzarlo su un dispositivo a blocchi che su un file. Potrebbe trattarsi di qualsiasi cosa, da grandi archivi di server FTP/HTTP, il cui contenuto non viene modificato spesso, all'utilizzo di una partizione `/usr` compressa o altre cose simili.

4.2.1. Esempio 1

Si supponga di avere una directory `/var/arch` con molti file e che si voglia trasformarla in un file system compresso, conservandola nella propria partizione di root come un file (si tratta di un file-immagine del file system da montare mediante un dispositivo di loopback). Le operazioni necessarie sono le seguenti:

Comprimere la directory, quindi montarla mediante loopback per controllo:

```
bash# mksquashfs /var/arch /var/arch.sqsh
bash# mkdir /mnt/tmp
bash# mount /var/arch.sqsh /mnt/tmp -t squashfs -o loop
```

```
bash# ls /mnt/tmp
```

Se tutto Ã¨ come ci si aspetta, montare il file system all'avvio automaticamente aggiungendo questa riga al proprio file `/etc/fstab`:

```
/var/arch.sqsh /var/arch squashfs ro,defaults 0 0
```

Smontare il file system dal punto di mount temporaneo e montarlo utilizzando la relativa voce in fstab :

```
bash# umount /mnt/tmp
bash# mount /var/arch
```

Ora, per assicurarsi che tutto Ã¨ andato bene

```
bash# ls /var/arch
```

4.2.2. Esempio 2

Si assuma di avere due partizioni del disco rigido, `/dev/hda6` (che Ã¨ vuota) e `/dev/hda7` (piÃ¹ grande di `/dev/hda6`, montata su `/var/arch`, che contiene dati ed Ã¨ piena). Ora, si consideri di voler comprimere il file system su `/dev/hda7` e spostarlo su `/dev/hda6`, al fine di utilizzare poi la partizione `/dev/hda7` per qualche altro scopo. Si supponga che sul file `/etc/fstab` (**reiserfs** Ã¨ solo un file system di esempio utilizzato su `/dev/hda7`) si abbia la seguente riga:

```
/dev/hda7 /var/arch reiserfs defaults 0 0
```

Allo stesso modo dell'esempio precedente:

```
bash# mksquashfs /var/arch /var/arch.sqsh
bash# mkdir /mnt/tmp
bash# mount /var/arch.sqsh /mnt/tmp -t squashfs -o loop
bash# ls /mnt/tmp
```

Se tutto Ã¨ andato bene, smontare `/dev/hda7` (se necessario) e usare **dd** per copiare `/var/arch.sqsh` su `/dev/hda6`:

```
bash# umount /dev/hda7
bash# dd if=/var/arch.sqsh of=/dev/hda6
```

Ora cambiare la riga su `/etc/fstab` per la partizione `/dev/hda7` in:

```
/dev/hda6 /var/arch squashfs ro,defaults 0 0
```

Montare il nuovo file system e verificare se tutto Ã¨ andato bene:

```
bash# mount /var/arch
bash# ls /var/arch
```

Non dimenticare di cancellare il file-immagine non piÃ¹ necessario:

```
bash# rm /var/arch.sqsh
```

4.3. Creazione di sistemi mininimali/integrati

Per sistemi "minimali/integrati", si vuole intendere sistemi Linux fatti per essere avviati da floppy disk, flash disk IDE/USB, CD-ROM iso9660, hard drive di piccole dimensioni e simili. Sia che si voglia il proprio file system di root interamente su un singolo media (una singola partizione, o un singolo floppy), oppure un sistema modulare (diversi floppy o partizioni), il procedimento Ã¨ quasi lo stesso. La creazione di tali sistemi Linux Ã¨ al di fuori della finalitÃ di questo HOWTO - per questo esistono HOWTO e guide dedicati (come il *Bootdisk HOWTO* e *Linux From Scratch* - visita www.tldp.org (http://www.tldp.org) per ottenere questi documenti).

4.3.1. File system SquashFS su floppy/flash/hard disk

Al fine di usare SquashFS per la creazione di sistemi Linux su piccoli dischi, si dovranno seguire i soliti passi per la creazione di un sistema minimale, eseguendo le seguenti procedure nei punti indicati:

1. Nella preparazione di un kernel per il proprio sistema, assicurarsi di abilitare il supporto per SquashFS, in modo da poter montare i relativi file system.
2. Utilizzare **mksquashfs** per creare ram disk iniziali in sola lettura o altri file system.
3. Non dimenticare di impostare il tipo di file system `squashfs` nel file `/etc/fstab` e gli script di avvio del proprio sistema per montare i file system compressi.

Esempio per Floppy. Si assuma di avere un punto di mount del floppy su `/home/user/floppylinux` e di voler porre il file system di root su un floppy disk e la directory `/usr` su un altro. Quello che si dovrebbe fare Ã¨:

```
bash# cd /home/user
bash# mksquashfs floppylinux root.sqsh -e usr
bash# mksquashfs floppylinux/usr usr.sqsh
```

Nota 1: si puÃ² vedere qui l'utilizzo dell'opzione **-e** per omettere la directory `/usr` dal file system di root.

Nota 2: non dimenticare di specificare `squashfs` nel file `/etc/fstab` del proprio disco di root o negli script di avvio per montare il file system `/usr`.

Inserire un disco di root nel propria porta floppy da 3.5" (si considera che su questo floppy siano presenti lilo o grub, che vi sia un file system, e che il file system di root SquashFS venga collocato sotto la directory `/boot` di quest'ultimo):

```
bash# mount /mnt/floppy
bash# cp root.sqsh /mnt/floppy/boot
```

Fatto ciÃ², smontare il floppy di root, cambiare floppy disk per `/usr` ed usare **dd** per trasferire il relativo file system:

```
bash# dd if=usr.sqsh of=/dev/fd0
```

4.3.2. File system compressi su CD-ROM

Con SquashFS Ã¨ possibile comprimere file system di grandi dimensioni, per esempio da utilizzare in Live-CD. A questo scopo SquashFS Ã¨ anche utilizzato insieme a UnionFS.

1. Abilitare SquashFS nel kernel Linux del sistema in oggetto
2. Creare un file system di root compresso
3. Modificare il file `/etc/fstab` o gli script di avvio del sistema in oggetto per montare il file system compresso quando necessario.

Se si crea un file system di root a partire da un sistema Linux in esecuzione, utilizzare l'opzione **-e** di **mksquashfs** per omettere tutti gli pseudo-filesystem come `/proc`, `/sys` (su un kernel linux successivo al rilascio 2.5.x) e `/dev` (quando si usa DevFS). Inoltre, non dimenticare di aggiungere il file-immagine, che Ã¨ stato creato con **mksquashfs** (si suppongono note le ragioni di queste esclusioni).

4.4. Rendere il file system scrivibile

Come accennato, un altro utilizzo interessante di **SquashFS** Ã¨ insieme al file system **Unionfs**, che fornisce semantica *copy-on-write* per i file system in sola lettura, migliorando le possibilitÃ Â . (Per Unionfs si veda http://www.filesystems.org/project-unionfs.html, il sito del progetto)

Per fare un esempio, si potrebbe voler comprimere la propria directory utente (home/user1), per creare una copia compressa dei propri file senza perdere la possibilitÃ di eseguire cambiamenti o scrivere nuovi file.

Creare il file system compresso `ro.fs` e la directory `rw.fs`.

```
bash# mksquashfs /home/user1 ro.fs
bash# mkdir /home/rw.fs
```

Montare il file system compresso ro.fs utilizzando un dispositivo di loopback.

```
bash# mount -t squashfs ro.fs /mnt -o loop
```

Montare il file system Unionfs, che rende `/mnt` e `/home/rw.fs` apparentemente fusi sotto il punto di mount `/home/user1`

```
bash# cd /home
bash# mount -t unionfs -o dirs=rw.fs=rw:/mnt=ro unionfs user1
```

Come si vede, ora Ã¨ possibile creare nuovi file in /home/user1.

```
bash# cd /home/user1
bash# touch file1
bash# ls
```

smontare il file system Unionfs e SquashFS ed elencare il contenuto delle directory /home/user1 e /home/rw.fs.

```
bash# cd ..
bash# umount /home/user1
bash# umount /mnt

bash# ls /home/user1
bash# ls /home/rw.fs
```

Come si nota, il nuovo `file1` Ã¨ stato creato in `/home/rw.fs` che Ã¨ il ramo scrivibile del file system unificato.

Quando si voglia aggiungere i nuovi file creati al file system SquashFS *stabile* e *compresso*, si potranno accodare a quello esistente.

```
bash# mksquashfs /home/rw.fs /home/ro.fs
```

Ora, per montare la directory utente compressa all'avvio del sistema, si puÃ² fare come segue:

Far caricare i moduli del kernel squashfs e unionfs all'avvio.

```
bash# echo squashfs >> /etc/modules
bash# echo unionfs >> /etc/modules
```

Cambiare il proprietario del ramo scrivibile, in modo che corrisponda all'utente.

```
chown user1 /home/rw.fs
```

Aggiungere queste righe al file /etc/fstab per montare SquashFS and UnionFS all'avvio.

```
...
/home/ro.fs  /mnt squashfs loop 0 0
unionfs /home/user1 unionfs dirs=/home/rw.fs=rw:/mnt=ro 0 0
```

5. Ringraziamenti

Vorrei esprimere i miei sinceri ringraziamenti ed immenso rispetto a:

- Phillip Lougher - per il suo lavoro brillante con squashfs, per aver creato una patch dedicata per Linux-2.4.18, per il suo aiuto nella revione di questo howto e le risposte alle mie email.
- Tabatha Marshall di TLDP per avermi aiutato a portare questo HOWTO alla versione di rilascio 1.0
- Tutti quelli di The Linux Documentation Project (http://www.tldp.org) per il loro grande lavoro su tutti gli HOWTO e le guide che mi hanno aiutato molto nell'esplorazione e hacking di Linux.
- Tutti gli utenti delle mailing list di TLDP che mi hanno aiutato ad iniziare
- Infiniti rigraziamenti e rispetto per tutti gli sviluppatori di software open-source.

Artemiy I. Pavlov

Voglio ringraziare Artemiy per la sua pazienza nel rispondere alle mie email, che mi hanno consentito di collaborare a questo HOWTO. Vorrei anche esprimere i miei ringraziamenti e rispetto a tutti colori sono impegnati nel sofware libero.

Marco Cecchetti

6. Licenza