`foremost` 823 `foremost.conf` 823 `sfill` 830
`$LD_PRELOAD` 826

21.1 Recupero dei dati cancellati

Ogni file system utilizza una propria procedura per amministrare i file all'interno dell'unità di memorizzazione. In generale vale il principio che la cancellazione di un file implichi semplicemente la rimozione dei riferimenti ai dati relativi, mentre la cancellazione effettiva dei contenuti avviene soltanto quando dei dati nuovi vanno a occupare la stessa area.

Con i file system più complessi, quali i tipi Second extended Ext3/Ext4 di Linux, è diventato impossibile risalire a un file cancellato, se non procedendo alla scansione completa dell'unità di memorizzazione. Quindi, tale tipo di ricerca si presta solo a un'analisi forense, alla ricerca di qualunque tipo di informazione disponibile.

21.1.1 Recupero da un file system generico con Foremost

Partendo dal presupposto che, nella maggior parte dei casi, i dati memorizzati in un file system utilizzano blocchi adiacenti, è possibile usare strumenti che scandiscono l'unità di memorizzazione alla ricerca di impronte di riconoscimento nei file. In generale, il programma più citato per questo tipo di scansione è Foremost.[1]

```
foremost [opzioni]
```

L'utilizzo più semplice che se ne può fare consiste nell'indicare un file-immagine da scandire e una directory vuota in cui vanno depositate le copie dei file rinvenuti:

```
# foremost -i immagine -o /tmp/recupero [Invio]
```

In questo esempio, il file-immagine è precisamente '`immagine`' e la directory in cui il programma deve copiare i file è '`/tmp/recupero/`'.

Nella directory usata per copiare i file rinvenuti, vengono create delle sottodirectory per tipologia di file individuato e un file con il riassunto delle operazioni svolte: '`audit.txt`'. La lettura di questo file consente di sapere da quale posizione del file-immagine sono stati estrapolati i file recuperati.

Come accennato, il riconoscimento dei file avviene attraverso delle impronte e il programma dispone già di un discreto numero di impronte incorporate nel codice, mentre è possibile aggiungerne altre nel file di configurazione: '`/etc/foremost.conf`' o quanto indicato con l'opzione '**`-c`** ***file***'. Il file di configurazione predefinito, normalmente è vuoto, oppure contiene la spiegazione di come va compilato attraverso dei commenti. A titolo di esempio si può vedere come potrebbe essere definito un tipo di file contenente codice HTML:

```
…
    htm         n       50000   <html   </html>
…
```

Il primo campo che appare in queste direttive indica il tipo di file che può essere usato nell'opzione '**`-t`** ***tipo***' e serve per attribuire un'estensione ai file individuati. In pratica, in questo caso, nella directory usata per depositare i file rinvenuti, questi file hanno una propria sottodirectory '`htm/`' e i nomi dei file terminano per '`.htm`'. Inoltre, volendo limitare la ricerca ai soli file di questo tipo, andrebbe usata l'opzione '**`-t htm`**'.

Il secondo campo indica se l'impronta debba essere considerata letteralmente o se sia indifferente l'uso di maiuscole e minuscole. In questo caso, la lettera '**`n`**' indica che non fa differenza se l'impronta appare con lettere maiuscole o minuscole; diversamente si userebbe la lettera '**`y`**'.

Il terzo campo indica la dimensione massima in byte del file da estrapolare. In questo caso si tratta di file che al massimo occupano 50 000 byte.

Il quarto campo indica l'impronta iniziale: da quel punto inizia il file. Il quinto campo, facoltativo, indica l'impronta che indica la fine del file.

L'esempio successivo mostra una direttiva del file di configurazione per individuare i file eseguibili in formato ELF. Purtroppo, in questo caso non è possibile indicare un'impronta che permetta di individuare correttamente la fine di tali file. Si osservi che l'impronta è scritta in forma esadecimale:

```
…
    elf         y       1000000     \x7F\x45\x4C\x46
…
```

Quando viene riconosciuto un file per il quale è prevista solo l'impronta iniziale di riconoscimento, diventa cruciale la dimensione massima (predefinita o fissata nel file di configurazione). Infatti, il file che viene individuato viene copiato fino alla dimensione massima, non essendoci altro modo di capire dove può terminare, salvo quando ci si trova alla fine dell'unità di memorizzazione. Ma anche se un file ottenuto dalla scansione ne contiene altri, se questi altri file sono individuabili, il loro contenuto viene comunque copiato in altrettanti file separati. In pratica, la somma delle dimensioni dei file copiati può superare di gran lunga la dimensione complessiva dell'unità di memorizzazione originale.

Tabella 21.3. Alcune opzioni per l'uso di Foremost.

Opzione	Descrizione
`-d`	Attiva il riconoscimento dei blocchi indiretti, utile quando si scandisce un file system Unix (tra quelli conosciuti da Foremost).
`-w`	Si limita a scrivere il file '`audit.txt`' nella directory di destinazione, senza copiare effettivamente i file individuati.
`-i` ***immagine***	Specifica il nome del file-immagine (o del file di dispositivo) da scandire alla ricerca dei file.
`-o` ***directory***	Specifica la directory di destinazione dei file da estrapolare e del file '`audit.txt`'. Se non viene specificata, si intende la directory '`output/`'.
`-c` ***configurazione***	Specifica il file di configurazione da usare. Generalmente, se non viene specificata questa opzione, dovrebbe trattarsi di '`/etc/foremost.conf`'.
`-t` ***tipo***	Se viene usata, questa opzione serve a limitare la ricerca a un certo tipo di file. Il nome che individua il tipo di file può essere trovato nella pagina di manuale *foremost(1)* e nel file di configurazione.

In generale, per una scansione all'interno di un file system Ext2 o Ext3, è conveniente l'uso dell'opzione '**`-d`**'; pertanto il comando tipo dovrebbe essere:

```
# foremost [-t tipo] [-w] -d -i immagine -o directory [Invio]
```

È evidente che la scansione di Foremost non tiene conto delle informazioni contenute nel file system, a parte il caso dell'opzione '**`-d`**', pertanto i file che si ottengono possono essere indifferentemente dati cancellati o dati che hanno una propria identificazione ancora valida.

Altri programmi affini, ma relativi specificatamente a file system Ext2.

Debugfs[2]	Programma di servizio utile per analizzare ed eventualmente modificare il contenuto di un file system di tipo Ext2.
The_Sleuth_Kit[3]	Si tratta di un raccolta di programmi di servizio utili per l'analisi di un file system Ext2.
Autopsy[4]	Si tratta di un programma frontale per l'uso mediato dei programmi che costituiscono il pacchetto The Sleuth Kit.
E2undel[5]	È un pacchetto composto da un programma e da una libreria, per il recupero dei file cancellati da un file system Ext2.

21.2 Libreria per la conservazione dei dati cancellati

I sistemi operativi commerciali, rivolti al pubblico generale, offrono solitamente la possibilità di ripescare i dati cancellati, in quanto questi finiscono di norma all'interno di un «cestino». Nei sistemi Unix sono i programmi usati nel sistema grafico che, in qualche modo, forniscono questo tipo di funzionalità, ma sempre in modo scoordinato, nel senso che programmi diversi potrebbero usare cestini differenti.

Volendo realizzare un cestino generalizzato dei dati cancellati in un sistema Unix, utilizzato da tutti i programmi nello stesso modo, occorre che le librerie standard siano ritoccate appositamente per tale scopo, in modo che le funzioni volte alla cancellazione di un file, in-

vece di farlo realmente, accantonino il file in una directory convenuta per questo scopo.

Nei sistemi GNU/Linux sono esistiti ed esistono lavori di questo tipo, utilizzando un meccanismo di sovrapposizione alle librerie standard, attraverso l'uso della variabile di ambiente ***LD_PRELOAD***, con la quale si informano i programmi di utilizzare prima una certa libreria e poi quelle comuni.

In questa sezione viene descritto un procedimento per utilizzare la libreria Libdelsafe, del pacchetto Delsafe. Va osservato però che **una qualunque libreria del genere, intervenendo sul controllo di funzioni delicate del sistema operativo, deve fare preoccupare un po' di più l'amministratore del sistema**.

21.2.1 Variabile di ambiente «LD_PRELOAD»

Un programma che fa uso di librerie dinamiche, quando all'interno della variabile di ambiente ***LD_PRELOAD*** trova l'indicazione di un percorso, tenta di caricare il file indicato in qualità di libreria, usando le sue funzioni con precedenza rispetto alle altre librerie per le quali è predisposto. Per esempio, se la variabile di ambiente ***LD_PRELOAD*** contiene la stringa '**`/lib/lib-mia.so`**', la libreria costituita dal file '`/lib/lib-mia.so`' viene caricata e usata prima di altre librerie eventuali.

La variabile di ambiente ***LD_PRELOAD*** può contenere un elenco di librerie che, in tal caso, sono separate dai due punti verticali «:», come per esempio: '**`/lib/lib-mia.so:/lib/lib-tua.so:/lib/lib-sua.so`**'.

Per fare in modo che gli utenti si trovino a operare con un certo insieme di variabili di ambiente, di norma si impostano tali valori utilizzando il file '`/etc/profile`' o qualunque altra cosa sia riconosciuta dalla shell, testuale o grafica che sia.

In presenza di una certa configurazione delle variabili di ambiente, si può avviare un programma con una configurazione differente, anteponendo al comando di avvio la nuova configurazione per le variabili. Per esempio, per far sì che un certo programma veda la variabile ***LD_PRELOAD*** vuota, basta un comando come quello seguente:

```
$ LD_PRELOAD="" comando [Invio]
```

21.2.2 Delsafe

Il pacchetto Delsafe,[6] composto dalla libreria Libdelsafe e da alcuni programmi di contorno, consente di organizzare in un sistema GNU/Linux dei «cestini», uno per ogni unità di memorizzazione per la quale ci si voglia proteggere dalle cancellazioni accidentali di file.

In pratica, la libreria Libdelsafe viene fatta caricare preventivamente ai programmi attraverso la variabile di ambiente ***LD_PRELOAD***. Quando un programma tenta di cancellare un file, facendo uso delle funzioni comuni destinate a tale scopo, la libreria verifica se per quell'unità di memorizzazione è presente la directory destinata a fare da cestino e se al suo interno c'è una sottodirectory con il nome dell'utente per conto del quale quel programma sta operando. Se è così, invece di cancellare il file, lo rinomina, trasferendolo all'interno di tale gerarchia, aggiungendo un'estensione che consente di risalire al momento della cancellazione formale.

I programmi di contorno che fanno parte del pacchetto Delsafe, servono principalmente a predisporre le directory da usare come cestini e a ripulire queste dai dati più vecchi, quando la mancanza di spazio lo richiede.

Il pacchetto Delsafe prevede l'uso di un file di configurazione, costituito normalmente da '`/etc/delsafe.conf`', il quale ha lo scopo principale di informare i programmi di contorno sulla collocazione delle directory usate come cestini e sulle modalità di ripulitura delle stesse. La libreria, invece, non interpella il file di configurazione e, per adattarla alle esigenze del proprio contesto, va compilata fornendo le opzioni appropriate.

Va osservato che del pacchetto Delsafe, solo la libreria Libdelsafe è indispensabile, perché gli altri programmi non svolgono compiti tanto sofisticati e un amministratore farebbe bene ad avere un controllo maggiore di tale gestione.

L'interposizione della libreria Libdelsafe a mediare le operazioni di cancellazione dei file, costituisce un elemento delicato nell'amministrazione del sistema, essendo giustificabile solo in un contesto dove la possibilità di recupero dei dati cancellati è molto importante. Tuttavia, per questa gestione deve essere coinvolto un amministratore competente e consapevole, il quale deve avere il massimo controllo su tale questione. Pertanto, di fronte al rischio che un uso errato dei programmi di contorno di Delsafe possa procurare dei danni inaspettati, è molto meglio predisporre manualmente le directory dei cestini e i propri script ritenuti appropriati alle necessità.

21.2.3 Preparazione di Libdelsafe

Sulla base delle considerazioni fatte nella sezione precedente, qui viene proposta la procedura per compilare e utilizzare la libreria Libdelsafe da sola. Si fa riferimento all'edizione 0.4.0 del pacchetto Delsafe che la contiene, distribuito in un solo file denominato '`delsafe-0.4.0.bin`', il quale è uno script che incorpora un archivio compresso da estrarre. Una volta scaricato questo file, avendolo collocato in una directory appropriata, si procede all'estrazione del suo contenuto:

```
$ chmod +x delsafe-0.4.0.bin [Invio]
```

```
$ ./delsafe-0.4.0.bin [Invio]
```

Lo script che avviluppa il pacchetto estrae i dati nella directory '`delsafe-0.4.0/`' e poi mostra la documentazione del pacchetto che viene fatta scorrere con l'ausilio di un programma come '**`more`**' o '**`less`**'. Concludendo il funzionamento del programma usato per scorrere la documentazione, lo script si appresta a fare delle domande per configurare la libreria e i programmi prima della compilazione. Qui si suggerisce di interrompere lo script e di procedere manualmente:

```
q
```

```
Trash name - .Trash? [Ctrl c]
```

Nella directory '`delsafe-0.4.0/`' si trova il file '`libdelsafe.c`' che può andare bene così, oppure si potrebbe ritoccare per le proprie esigenze.

```
$ cd delsafe-0.4.0 [Invio]
```

Ciò che potrebbe essere modificato, eventualmente dopo che si ha guadagnato un po' di esperienza nell'uso di tale libreria, è quanto appare nel listato seguente, dove le voci modificabili sono evidenziate:

```
…
  /*** BEGIN OF DEFAULT STATIC OPTIONS (from config at install time) */
  char *d_mtab="/etc/mtab";
#if defined DEBUG
  char *d_trash_rootname=".Lixo";
#else
  char *d_trash_rootname=".Trash";
#endif
  char *d_exclude_heads[]={
    "/tmp/",
    "/proc/",
    "/dev/",
    "/sys/",
    "/var/",
    "/root/.xauth",
    "/root/.Xauthority",
    "/root/.DCOPserver",
    "/root/.ICEauthority",
    "/root/.xauth",
    ""
  };
  char *d_exclude_tails[]={
    "~",
```

```
    ".o",
    ""
  };
  char *d_exclude_files[]={
    "/etc/mtab",
    ""
  };
  char *d_exclude_patts[]={
    "^/home/.+/\\.kde",
    "^/home/.+/\\.Xauthority.*",
    "^/home/.+/\\.DCOPserver.+",
    "^/home/.+/\\.ICEauthority.*",
    "^/home/.+/\\.fonts\\.cache.*",
    "^/home/.+/\\.serverauth.*",
    "^/home/.+/\\.beagle/.*",
    ""
  };
  int   d_error_action=LEAVE;
  int   d_xdev_action=FORCE;
  /*** END OF DEFAULT STATIC OPTIONS (from config at install time) */
…
```

Secondo le intenzioni dell'autore, le voci che qui sono state evidenziate, sono da definire attraverso il file di configurazione, il quale viene usato per modificare automaticamente il file '`libdelsafe.c`', prima della compilazione. Nel metodo proposto per procedere alla compilazione, questa modifica automatica non avviene.

Una volta modificato il file '`libdelsafe.c`', ammesso che lo si voglia modificare effettivamente, per la compilazione si deve entrare nella directory '`Relise/`' e da lì eseguire i comandi necessari.

```
$ pwd [ Invio ]

  …/delsafe-0-4-0

$ cd Relise [ Invio ]

$ make -f makefile clean [ Invio ]

$ make -f makefile [ Invio ]
```

A questo punto, nella stessa directory '`Relise/`' si deve trovare il file '`libdelsafe-0.4.0.so`' che costituisce la libreria compilata.

Dopo la compilazione, la libreria va collocata in una posizione appropriata e, di conseguenza, bisogna provvedere a configurare la variabile di ambiente ***LD_PRELOAD*** in modo che sia utilizzata. Si può supporre che il file '`libdelsafe-0.4.0.so`' sia collocato nella directory '`/lib/`' e, di conseguenza, nel file '`/etc/profile`' potrebbe apparire una coppia di istruzioni come quella seguente:

```
…
LD_PRELOAD="/lib/libdelsafe-0.4.0.so"
export LD_PRELOAD
…
```

A questo punto, i programmi che incontrano la variabile di ambiente ***LD_PRELOAD*** con il contenuto ipotizzato sopra, utilizzano sempre la libreria Libdelsafe prima di altre, ma la libreria non è efficace se non trova le directory da usare come cestino.

21.2.4 Predisposizione dei cestini

La libreria Libdelsafe è sostanzialmente inerte se non trova le directory destinate alla conservazione dei file che formalmente vengono rimossi. In condizioni normali, si tratta delle directory '`.Trash/`' collocate all'inizio delle gerarchie delle unità di memorizzazione in cui si vuole prevenire la cancellazione. Per esempio, se ci si vuole salvaguardare dalla cancellazione dei dati contenuti nel file system innestato a partire da '`/home/`', occorre ci sia la directory '`/home/.Trash/`'; nello stesso modo, se ci si vuole salvaguardare per il file system principale, occorre la directory '`/.Trash/`'.

Il nome di questa directory è definito nella libreria e può essere modificato. Per la precisione si tratta dell'istruzione seguente, già vista in precedenza:

```
…
  char *d_trash_rootname=".Trash";
…
```


Tuttavia, le directory '`.Trash/`', da sole, non bastano: occorre che ogni utente, per il quale si vuole garantire la possibilità di recuperare i file cancellati, sia presente una sottodirectory con il nome dell'utenza stessa. Per esempio, se si vuole che l'utente '**`tizio`**' possa recuperare i propri file cancellati, occorre che sia presente la directory '`.Trash/tizio/`'. Naturalmente, le sottodirectory degli utenti devono appartenere agli utenti relativi e devono avere i permessi 0700_8, in modo da impedire a tutti gli altri di accedervi.

Se si vuole che anche l'utente '**`root`**' possa recuperare i dati che cancella, occorre che sia predisposta la directory '`.Trash/root/`', secondo le stesse modalità già descritte. Tuttavia, sarebbe auspicabile che per tale utente questa misura non fosse necessaria.

La libreria Libdelsafe, quando trova la directory corretta per la conservazione dei file cancellati, ricrea, a partire dalla stessa, il percorso del file cancellato, usando per il file un nome con un'estensione costituita dalla data e dall'ora della cancellazione. Nel caso un utente fosse interessato al recupero di un certo file, non dovrebbe fare altro che spostarlo, cambiandogli nome nel modo più appropriato.

21.2.5 Script ausiliari a Libdelsafe

Il pacchetto Delsafe, oltre alla libreria Libdelsafe, fornisce dei programmi che possono facilitare il recupero dei file cancellati e consentono di tenere sotto controllo lo spazio utilizzato, eliminando periodicamente ciò che diventa troppo vecchio. Questi programmi non sono indispensabili e si può provvedere autonomamente alla realizzazione dei propri script.

Per prima cosa è necessario fornire un modo agli utenti per eseguire dei programmi senza l'ausilio della libreria Libdelsafe. Per ottenere tale risultato è sufficiente fornire a questi programmi la variabile di ambiente vuota, tuttavia si può predisporre un piccolo script:

```
#!/bin/sh

TRASH=OFF LD_PRELOAD="" "$@"
```

Per fare in modo che le directory contenenti i cestini vengano ripulite periodicamente, si può predisporre uno script come quello seguente, da avviare giornalmente attraverso Cron (di norma è sufficiente collocarlo materialmente nella directory '`/etc/cron.daily/`'). In questo caso i file più vecchi di una settimana vengono eliminati, assieme a eventuali directory vuote:

```
#!/bin/sh
# Modificare /home/.Trash/* in base alla collocazione
# effettiva:
for TRASH in /home/.Trash/*
do
    find $TRASH -mtime +7 -and -ctime +7 \
         -and \( -not -type d \) -exec rm \{\} \;
    find $TRASH -type d \
         -exec rmdir --ignore-fail-on-non-empty \{\} \;
done
```

21.3 Controllo dello spazio rimasto libero

Se da un lato ci può essere l'esigenza di ripescare i dati cancellati all'interno di un'unità di memorizzazione, dall'altro c'è sicuramente quella di poter eliminare definitivamente ciò che si intende cancellare. Il modo più semplice e intuitivo per premunirsi contro la ricerca di dati all'intero dello spazio inutilizzato di un'unità di memorizzazione, consiste nel creare una serie di file temporanei, il cui contenuto sia innocuo, cancellandoli successivamente.

Questo problema può essere considerato con maggiore o minore serietà, a seconda del contesto: se si tratta solo di un piccolo scrupolo, una sovrascrittura singola dei dati può essere sufficiente; se invece il problema della riservatezza è grave, sono necessarie più sovrascritture con dati diversi e casuali.

L'efficacia del procedimento di sovrascrittura dei dati cancellati dipende anche dalle caratteristiche del file system e dal modo in cui

si comunica effettivamente con l'unità di memorizzazione. Per avere un risultato accettabile occorre che l'unità di memorizzazione sia collegata localmente, perché un file system di rete potrebbe essere gestito tramite una memoria tampone significativa, dove molti file creati e poi cancellati non vengono nemmeno memorizzati effettivamente, o comunque dove non tutte le sovrascritture ripetute vengono eseguite realmente. Inoltre, il file system non deve conservare lo storico (ovvero il diario) delle modifiche ai file, altrimenti il procedimento di sovrascrittura diventa perfettamente inutile e dannoso per altri versi.

Eventualmente va considerato anche il problema della cancellazione della memoria di scambio (quella che estende la memoria virtuale in un'unità di memorizzazione di massa), ma in questo capitolo la questione non viene trattata.

21.3.1 Secure-delete

Secure-delete[7] è un piccolo insieme di programmi destinati alla cancellazione definitiva dei dati. In particolare è importante il programma '**sfill**', con il quale si va a sovrascrivere più volte lo spazio inutilizzato all'interno di un file system:

```
sfill [opzioni] directory
```

La sintassi del programma richiede almeno l'indicazione di una directory, all'interno della quale creare i file temporanei per la sovrascrittura. Per esempio, supponendo che nella directory '`/home/`' sia innestato il file system all'interno del quale si vuole fare un po' di pulizia, sarebbe sufficiente il comando seguente:

```
# sfill /home [Invio]
```

In generale, il programma va usato dall'amministratore, ma non viene escluso dagli utenti comuni. Tuttavia, se un utente comune utilizza '**sfill**', è molto probabile che non possa sovrascrivere tutto lo spazio libero, in base alla politica del sistema.

Tabella 21.11. Alcune opzioni per l'uso di '**sfill**'.

Opzione	Descrizione
`-f`	Richiede una modalità di funzionamento rapida (*fast*), ma logicamente poco efficace sul piano della sicurezza della cancellazione.
`-l`	Richiede solo due sovrascritture (*less*), una delle quali con dati casuali.
`-z`	Richiede che l'ultimo passaggio di sovrascrittura avvenga utilizzando il valore zero.

Va letta la documentazione, costituita dalla pagina di manuale *sfill(1)*, se il problema della cancellazione dei dati è delicato nel proprio contesto. Si vedano eventualmente anche gli altri programmi del pacchetto Secure-delete.

21.3.2 Esempio di programma per la sovrascrittura dello spazio libero

Viene proposto un programma molto semplice, scritto in C, con il quale si crea una serie di file nella directory corrente, contenenti la ripetizione di una stringa fornita attraverso la riga di comando. La scrittura avviene una sola volta e i file temporanei vanno cancellati manualmente, dopo l'uso del programma. Il sorgente dovrebbe essere disponibile presso *allegati/c/empty-space-filler.c* .

```
empty-space-filler [blocco [stringa]]
```

Ammesso che il programma venga chiamato come si vede nel modello sintattico, il primo argomento opzionale è costituito dalla dimensione del blocco, ovvero la dimensione di ogni file temporaneo creato; il secondo argomento, se fornito, deve essere la stringa da usare ripetutamente per riempire lo spazio dei file temporanei. Per esempio, il programma potrebbe essere usato così:


```
# cd /tmp [Invio]

# empty-space-filler 131072 "Ciao a tutti! :-) :-) :-)" [Invio]
```

Naturalmente, se non si forniscono opzioni vengono usati i valori predefiniti nel programma.

Per compilare il programma, ammesso che il file originale sia '`empty-space-filler.c`', si può procedere nel modo seguente:

```
$ cc -o empty-space-filler empty-space-filler.c [Invio]
```

21.4 Copie di sicurezza

L'amministrazione di un sistema Unix è da sempre un grosso problema sotto tutti i punti di vista. Il primo dovere tra tutti è quello della salvaguardia dei dati, ma al rischio della loro perdita si pone rimedio solo attraverso una gestione corretta delle copie di sicurezza. Gli strumenti a disposizione per eseguire copie di sicurezza sono molti e si possono distinguere due estremi possibili:

- copiare i file e le directory;
- copiare una partizione o un'unità intera.

La copia di una partizione o di un'unità intera può avere il vantaggio di permettere l'archiviazione della situazione esatta in cui si trova, problemi inclusi. Inoltre, non avendo un processo di lettura sui file, la data di lettura di questi non viene modificata. Lo svantaggio fondamentale di tale tipo di copia è che questa è riferita a un'unità particolare (o a una partizione) di una macchina particolare: è poco probabile che si possano recuperare dati archiviati in questo modo in un'unità differente che non abbia caratteristiche equivalenti.

La copia di file e directory non tiene conto del supporto fisico in cui si trovano e nemmeno del tipo di file system utilizzato. Questo comporta una serie di conseguenze:

- i file e le directory vengono scanditi in lettura, alterando quindi le date di lettura;
- i collegamenti simbolici vanno copiati come tali e non devono essere copiati gli oggetti a cui questi puntano;
- i collegamenti fisici potrebbero non essere distinti come tali.

In generale, dal momento che una copia di file e directory è portabile, mentre una copia di un dispositivo intero non lo è (quando non si tratta di un dispositivo standard), dovrebbe essere preferibile la prima di queste due soluzioni.

21.4.1 Archiviazione

La copia pura e semplice dei file e delle directory è una tecnica possibile, ma richiede condizioni particolari:

- l'unità di destinazione deve essere in grado di accogliere i dati come sono all'origine, in pratica dovrebbe trattarsi di un'unità provvista di file system;
- deve trattarsi di unità rimovibili;
- la capacità di queste unità deve essere maggiore di quella del file più grande che si deve archiviare.

Di solito si preferisce la tecnica dell'archiviazione dei dati in un file unico (che rappresenta l'archivio), assieme a tutte le informazioni necessarie per riprodurre i file e le directory originali. In questo modo si possono utilizzare unità di memorizzazione di qualunque tipo, eventualmente suddividendo l'archivio in pezzi più piccoli contenibili al loro interno.

Gli oggetti contenibili in un file system possono essere di vario tipo (file puri e semplici, directory, file di dispositivo, collegamenti, ecc.) e così pure i loro attributi (permessi, date, ecc.). Il sistema di archiviazione che si utilizza deve essere in grado riprodurre correttamente tutti i dati del tipo di file system che si utilizza. Per esempio, non sarebbe possibile archiviare i dati di un file system Unix in un archivio '`.zip`' che è nato per gli ambienti Dos.

21.4.2 Percorsi assoluti o relativi

Quando si intende archiviare una porzione di file system e quindi solo ciò che si trova a partire da una certa directory in poi, è importante sapere come si comporta il programma di archiviazione al riguardo della registrazione dei percorsi (*path*). Se si vuole archiviare la directory '`/home/tizio/esempi/`', il programma di archiviazione potrebbe registrare il suo contenuto in uno dei tre modi seguenti:

1. '`/home/tizio/esempi/*`'
2. '`home/tizio/esempi/*`'
3. '`./*`'

Nel primo caso, quando dovesse rendersi necessario il recupero dei dati, questi verrebbero collocati esattamente nella directory indicata, in modo assoluto. Nel secondo, verrebbero collocati in modo relativo a partire dalla directory corrente, ottenendo così la directory '`./home/tizio/esempi/*`'. Nel terzo caso si avrebbe il recupero del contenuto di quella directory senza informazioni sul percorso precedente.

21.4.3 Strategia nelle copie

Le copie di sicurezza permettono di conservare la situazione dei dati in un istante determinato, ma i dati sono soggetti a continui aggiornamenti. Per questo occorre una procedura attraverso la quale si possa avere una gestione ordinata e ragionevolmente sicura delle copie.

A parte i rischi connessi con il tipo di supporto utilizzato per le copie e il luogo in cui queste vengono conservate, vanno almeno considerate le modalità sequenziali con cui queste possono essere eseguite. È importante rispettare un paio di regole elementari:

1. non si riutilizzano i supporti contenenti la copia effettuata la volta precedente;
2. non si utilizzano supporti in cattive condizioni.

21.4.4 Generazioni delle copie

Per distinguere una copia effettuata in un momento rispetto a quella fatta in un altro, si parla di ***generazione***. In pratica, l'ultima copia di sicurezza effettuata è l'ultima generazione, mentre le altre sono tutte generazioni precedenti. Questo termine si riferisce naturalmente a copie fatte sullo stesso insieme di dati.

Il buon senso suggerisce di utilizzare almeno tre generazioni di copie: l'ultima, quella precedente e quella ancora precedente.

21.4.5 Livelli delle copie

La copia di un file system intero può comportare un certo impegno, sia in termini di tempo, sia nella quantità di supporti impiegati. Il programma che si utilizza per le copie, oppure un gruppetto opportuno di script di shell, potrebbe permettere di effettuare la copia successiva dei soli file che sono stati modificati nel frattempo.

Quando si esegue una copia dei soli file che risultano diversi rispetto all'ultima copia completa, si parla di ***copia di primo livello***; quando se ne esegue un'altra in un momento successivo per le variazioni avvenute dopo quella di primo livello, si parla di ***secondo livello*** e così di seguito.

A parte le difficoltà legate alla conservazione dell'informazione sullo stato dei file (di solito si tratta della data di modifica e di creazione), si pongono poi dei problemi nel momento in cui dovesse essere necessario un ripristino dalle copie. Si dovrebbe ripristinare l'ultima copia completa, seguita da tutte quelle aggiuntive dei soli file modificati, nello stesso ordine in cui sono state fatte: dalla più vecchia alla più recente.

Sotto questo aspetto, quando non si vuole ripetere una copia completa troppo frequentemente, si cerca almeno di eseguire copie successive sempre di primo livello (si hanno quindi più generazioni di copie di primo livello). In tal modo, un eventuale recupero richiederebbe solo il ripristino dell'ultima generazione di copia completa e dell'ultima generazione di copia di primo livello.

Questo sistema potrebbe non tenere conto dei file cancellati dopo l'ultima generazione di copia completa: in tal modo, un eventuale recupero dalle copie di sicurezza potrebbe comportare il ripristino di file che non servono più.

21.4.6 Distribuire le responsabilità

In un sistema monoutente, l'unico utilizzatore è anche l'amministratore del proprio sistema e di conseguenza anche l'unico responsabile. È quindi lui (o lei) a sapere esattamente cosa ha fatto e cosa è necessario copiare per sicurezza.

In un sistema multiutente o comunque quando si condividono dati in gruppo, anche se a prima vista potrebbe sembrare conveniente la gestione delle copie in modo centralizzato, è comunque utile affidarla in parte anche alla responsabilità dei singoli:

- periodicamente, un amministratore potrebbe occuparsi di eseguire la copia complessiva di tutto il sistema;
- ogni giorno, gli utenti dovrebbero preoccuparsi di eseguire le copie dei dati di loro competenza.

Nel momento in cui dovesse essere necessario, si dovrebbero recuperare i dati dalle copie generali fatte dall'amministratore e successivamente da quelle particolari dei singoli utenti.

Se determinate attività vengono svolte in gruppo, si potrebbe eleggere ugualmente un responsabile all'interno di questo che si occupi delle copie di quell'attività.

Il vantaggio di questo metodo sta nell'alleggerimento delle responsabilità dell'amministratore e nella soluzione più facile di piccoli problemi locali:

- se un gruppo di lavoro ha alterato i dati a causa di un'operazione errata, è sufficiente recuperare i dati di quel gruppo senza disturbare l'intero sistema;
- la distribuzione della responsabilità aumenta la consapevolezza da parte degli utenti.

21.4.7 Supporti

La scelta del supporto di conservazione della copia è importante e comporta alcune conseguenze:

- il costo;
- la disponibilità di unità in grado di utilizzarli;
- la facilità o difficoltà nel recupero di porzioni dei dati archiviati.

Si possono trovare unità di memorizzazione di massa di grandi capacità a prezzi ragionevolmente bassi, come i dischi fissi esterni (collegati attraverso la porta USB). Questi hanno il vantaggio di poter essere utilizzati come unità normali, pertanto, il recupero di dati parziali diventa molto più facile, anche quando la copia di sicurezza avviene per mezzo di un'archiviazione tradizionale. Per le copie che devono essere conservate in tempi lunghi, diventano convenienti i dischi ottici, data la loro economicità.

21.4.8 Compressione

Il problema della dimensione dei dati da archiviare può essere ridotto parzialmente con l'aiuto della compressione. La tecnica della compressione può essere applicata all'archiviazione in due modi possibili:

- prima della costruzione dell'archivio, ottenendo così un archivio di file compressi;
- dopo la costruzione dell'archivio, ottenendo così un archivio compresso.

La differenza è enorme. La compressione introduce un elemento di rischio maggiore nella perdita di dati: se una copia di sicurezza viene danneggiata parzialmente, l'effetto di questo danno si riflette in una quantità di dati maggiore (spesso è compromesso tutto l'archivio).

I programmi di archiviazione compressa maggiormente diffusi negli ambienti Dos utilizzano la tecnica della compressione prima dell'archiviazione. È questo il caso degli archivi '`.zip`', '`.arj`', '`.lzh`' e di altri ancora. Tale sistema ha il vantaggio di permettere una facile scansione dell'archivio alla ricerca di file da estrarre (e decomprimere) o un ampliamento dell'archivio in un momento successivo alla sua creazione. Un altro vantaggio è la minore sensibilità alla perdita dei dati: se una parte dell'archivio è danneggiato, dovrebbe essere possibile ripristinare almeno il resto. Lo svantaggio principale è che la compressione fatta in questo modo, a piccoli pezzi, non è molto efficiente.

La compressione fatta dopo l'archiviazione elimina ogni possibilità di accedere ai dati contenuti nell'archivio e di poterlo ampliare, se non dopo averlo decompresso. Questo significa anche che un danneggiamento parziale dell'archivio implica la perdita di tutti i dati da quel punto in poi.[8] Un altro tipo di problema deriva dalla difficoltà di distribuire un archivio compresso suddividendolo su più unità di memorizzazione. In questo caso però, l'efficienza della compressione è massima. Negli ambienti Unix, di fatto, è questa la scelta preferita.

21.5 Archiviazione e recupero attraverso tar e gzip

La coppia Tar e Gzip rappresenta lo standard nell'archiviazione dei dati: Tar genera un archivio non compresso che può comprendere anche collegamenti simbolici e file speciali; Gzip lo comprime generando un archivio più piccolo. La coppia funziona così bene che la versione GNU di Tar è in grado di utilizzare Gzip direttamente senza dover far uso di condotti, purché il risultato dell'archiviazione non debba essere suddiviso su più supporti. L'origine del nome Tar è *Tape archive*, ma questo programma permette ugualmente di gestire qualunque altro tipo di sistema di memorizzazione. La versione GNU di Tar (quella utilizzata normalmente nelle distribuzioni GNU/Linux), non memorizza percorsi assoluti. I programmi Tar e Gzip sono descritti rispettivamente nelle sezioni 20.13.2 e 20.14.1.

Negli esempi seguenti si immagina di dover archiviare il contenuto della directory '`~/lettere/`', equivalente a '`/home/tizio/lettere/`', comprese eventuali sottodirectory discendenti.

Negli esempi si cerca di utilizzare la forma tradizionale per l'indicazione delle opzioni standard di Tar. Alcune di queste possono fare a meno del trattino iniziale, come nel caso di '**`c`**' e '**`x`**'. Altre opzioni hanno quel trattino, ma possono essere aggregate in un'unica serie di lettere, come nel caso di '**`czvf`**', dove si ha l'unione di: '**`c`**', '**`-z`**', '**`-v`**' e '**`-f`**'.

21.5.1 Archiviazione diretta su dispositivi di memorizzazione

L'archiviazione attraverso la registrazione diretta sui dispositivi utilizza completamente il supporto di memorizzazione destinatario, anche se la quantità di dati da archiviare è molto piccola. Gli esempi che vengono proposti qui si rifanno a dispositivi di memorizzazione del passato, di capacità limitate, servendo oggi solo come casi estremi di utilizzo.

Quello sotto indicato è un esempio di archiviazione in un nastro magnetico singolo: l'opzione '**`c`**' sta per *Create*; '**`f`**' sta per *File* e permette di definire la destinazione dell'archiviazione; '**`z`**' attiva la compressione attraverso '**`gzip`**'. Dal momento che si utilizza la compressione, l'archiviazione multivolume non è ammissibile.

```
# tar czf /dev/ftape ~/lettere [Invio]
```

I dischetti possono essere utilizzati come i nastri, in modo sequenziale, ma questo lo si fa soltanto quando l'archivio generato non è contenibile in un solo dischetto: si ha quindi una copia multivolume e in tal caso non è ammissibile l'uso della compressione.

```
# tar cf /dev/fd0u1440 -M ~/lettere [Invio]
```

In questo caso, l'opzione '**`-M`**' sta proprio per *Multivolume* indicando quindi la possibilità che il supporto di destinazione non sia in grado di contenere l'intero archivio. In tal modo, Tar si prende cura di sospendere l'archiviazione ogni volta che viene raggiunta la capienza massima. Tar non è in grado di determinare da solo questa capacità, per cui, nell'esempio, il file di dispositivo dell'unità a dischetti è stato indicato in modo da riconoscerne la geometria, ma in alternativa si potrebbe utilizzare l'opzione '**`-L`**' seguita dalla dimensione:

```
# tar cf /dev/fd0 -M -L 1440 ~/lettere [Invio]
```

Quando si utilizzano in questo modo, i dischetti non contengono un file system e di conseguenza non possono essere innestati. La lettura del loro contenuto avviene nello stesso modo della scrittura, attraverso il nome del dispositivo.

L'archiviazione su dischetti, attraverso l'indicazione del file di dispositivo, richiede comunque che questi siano già stati inizializzati (a basso livello) secondo il formato che viene indicato. Non conta che siano vuoti: è importante che ci siano le tracce e i settori come previsto.

21.5.2 Archiviazione normale su file

Quando l'archiviazione può essere fatta su dischi (con file system) di dimensione sufficiente a contenere l'archivio intero, invece di utilizzare l'opzione '**`f`**' per specificare un file di dispositivo, si può indicare direttamente un normalissimo file al loro interno, come nell'esempio seguente:

```
$ tar cf /mnt/sdc1/lettere.tar ~/lettere [Invio]
```

In pratica, nel caso appena visto, si utilizza un file system innestato nella directory '`/mnt/sdc1/`' e si crea il file '`lettere.tar`' al suo interno.

L'archiviazione compressa, con l'utilizzo di '**`gzip`**', può essere ottenuta semplicemente con l'opzione '**`z`**', come nell'esempio seguente:

```
$ tar czf /mnt/sdc1/lettere.tar.gz ~/lettere [Invio]
```

In tal caso l'estensione standard utilizzata (ma non obbligatoria) è '`.tar.gz`' che rende esplicito il fatto che la compressione è stata fatta dopo l'archiviazione. In alternativa si può usare anche '`.tgz`', diffusa nei sistemi Dos.

21.5.3 Archiviazione e percorsi

Gli esempi seguenti, pur archiviando gli stessi dati, mostrano un modo diverso di registrare i percorsi all'interno dell'archivio. La directory di lavoro nel momento in cui si avvia il comando, è '`/home/tizio/`', corrispondente alla directory personale dell'utente.

```
/home/tizio$ tar czf /mnt/sdc1/lettere.tar.gz ~/lettere [Invio]

/home/tizio$ tar czf /mnt/sdc1/lettere.tar.gz ↩
↪               /home/tizio/lettere [Invio]

/home/tizio$ tar czf /mnt/sdc1/lettere.tar.gz lettere [Invio]

/home/tizio$ tar czf /mnt/sdc1/lettere.tar.gz ./lettere [Invio]
```

Nei primi due esempi, viene archiviata l'indicazione del percorso precedente, ma pur essendo stato dato in modo assoluto ('`/home/tizio/lettere`'), questo viene reso relativo da Tar, eliminando la prima barra obliqua che si riferisce alla directory radice. Questo comportamento riguarda almeno la realizzazione GNU di Tar. Negli ultimi due esempi, viene archiviata l'indicazione della sola directory '`lettere/`', sempre in modo relativo.

21.5.4 Archiviazione di periodi

I file sono forniti di informazioni orarie. In base a queste è possibile eseguire delle copie di sicurezza riferite a dei periodi. Le copie di sicurezza a più livelli possono essere ottenute in modo semplificato attraverso l'uso dell'opzione '**-N**' seguita da una data di partenza: si ottiene l'archiviazione di quanto variato a partire da una certa data; di solito si utilizza quella dell'ultima archiviazione completa. Il concetto di variazione, in questo caso, si deve intendere come variazione del contenuto o degli attributi. Quindi si tratta della data di modifica o della data di «creazione».

```
$ tar czf /mnt/sdc1/lettere.tar.gz -N 20120801 ↩
↪        ~/lettere [Invio]
```

In questo caso, la data che segue l'opzione '**-N**' rappresenta la mezzanotte del primo agosto 2012.

```
$ tar czf /mnt/sdc1/lettere.tar.gz -N "20120801 15:30" ↩
↪        ~/lettere [Invio]
```

Questo ultimo esempio aggiunge alla data l'indicazione di un'ora particolare, 15:30; per evitare che sia interpretato in maniera errata, il gruppo data-orario viene racchiuso tra virgolette.

21.5.5 Archiviazione limitata a un'unità

Quando si eseguono delle copie di sicurezza, è probabile che si voglia archiviare solo la situazione di una certa unità di memorizzazione (partizione o directory condivisa in rete). In tal caso si deve indicare precisamente questo limite con l'opzione '**-l**' (*Limit*).

21.5.6 Estrazione dei percorsi

Quando si accede all'archivio per estrarne il contenuto o per compararlo con i dati originali, entra in gioco il problema dei percorsi. I dati vengono estratti normalmente nella directory corrente, oppure vengono comparati utilizzando come punto di partenza la directory corrente. Quindi, se l'archivio contiene la directory degli esempi precedenti, registrati a partire dalla radice (ma come già spiegato, senza l'indicazione della radice stessa), questi vengono estratti in '`./home/tizio/lettere/`', oppure comparati con i dati contenuti a partire da questo percorso.

Se in fase di estrazione o comparazione si vuole fare riferimento a percorsi assoluti, si può utilizzare l'opzione '**-P**'. In questo modo si afferma esplicitamente che i percorsi indicati nell'archivio vanno considerati come discendenti dalla directory radice.

21.5.7 Recupero

Per poter effettuare un recupero di dati da un archivio è necessario conoscere in particolare il modo in cui questo è stato creato: normale, compresso, multivolume.

In generale, per recuperare dati da un archivio si utilizza l'opzione '**x**' (*Extract*) al posto di '**c**' e a essa si devono eventualmente aggiungere '**z**' nel caso di estrazione da un archivio compresso con '**gzip**' o '**-M**' nel caso di un archivio multivolume.

Durante il recupero di una copia di sicurezza è importante fare in modo che i dati riprodotti mantengano gli stessi attributi originali (permessi e proprietà). Per questo si aggiungono le opzioni '**-p**' (riproduce i permessi) e '**--same-owner**' (riproduce le proprietà: UID e GID).

L'esempio seguente mostra un recupero da un archivio multivolume su dischetti:

```
~$ tar x -M -p --same-owner -f /dev/fd0u1440 [Invio]
```

L'esempio seguente mostra un recupero con percorso assoluto: i percorsi indicati all'interno dell'archivio vengono aggiunti alla directory radice:

```
$ tar xz -P -p --same-owner -f /mnt/mo1/lettere.tar.gz [Invio]
```


21.5.8 Recupero parziale

Il recupero parziale del contenuto di un archivio Tar può essere fatto per file singoli o per directory, oppure attraverso l'uso di caratteri jolly. In questo ultimo caso però, occorre fare attenzione a evitare che la shell esegua l'espansione: è compito di Tar determinare a cosa corrispondano all'interno dei suoi archivi. Questo è un po' quello che accade a Find con l'opzione '**-name**': è Find stesso ad analizzare i caratteri jolly.

Valgono le regole solite: l'asterisco rappresenta un insieme di caratteri qualunque; il punto interrogativo rappresenta un carattere qualsiasi; le parentesi quadre rappresentano un carattere a scelta tra un insieme o tra un intervallo determinato.

Quando si indicano nomi di file o directory, o quando si utilizzano i caratteri jolly, occorre tenere presente che si sta facendo riferimento ai dati contenuti nell'archivio, con i percorsi memorizzati originariamente. Inoltre, se con i caratteri jolly si determina la corrispondenza con una directory, si ottiene l'estrazione del contenuto complessivo di quella.

L'esempio seguente mostra in che modo potrebbero essere recuperate le lettere contenute nella directory '`home/tizio/lettere/nuove/`' (l'esempio appare diviso su due righe, a causa della sua lunghezza):

```
$ tar xz -P -p --same-owner -f /mnt/sdc1/lettere.tar.gz ↩
↪     home/tizio/lettere/nuove [Invio]
```

L'esempio seguente mostra l'estrazione di tutti i file e delle directory corrispondenti a '`home/tizio/lettere/ve*`'. Gli apici sono necessari per evitare che intervenga la shell a espandere l'asterisco, che invece deve essere interpretato da Tar.

```
$ tar xz -P -p --same-owner -f /mnt/sdc1/lettere.tar.gz ↩
↪     'home/tizio/lettere/ve*' [Invio]
```

21.5.9 Elenco e controllo

Per ottenere un elenco del contenuto di un archivio e per compararne il contenuto con i dati originali, valgono le stesse regole del recupero dei dati. In particolare, al posto dell'opzione '**x**' si deve utilizzare '**t**' (*List*) per gli elenchi e '**d**' (*Diff*) per la comparazione.

21.5.10 Archiviazione di un file system

L'archiviazione di un file system intero, va fatta considerando le caratteristiche di questo, in particolare della sua struttura fisica: partizioni e condivisione attraverso la rete. In generale dovrebbe essere conveniente l'archiviazione separata per ogni partizione e per ogni file system condiviso in rete.

Oltre a questo occorre evitare di archiviare anche l'archivio che si sta creando: quando la destinazione dell'archiviazione è un file, il file system che lo contiene deve essere innestato da qualche parte e per questo si potrebbe creare un circolo vizioso.

Ci sono directory che, per la loro natura, non conviene o non devono essere archiviate: per '`/tmp/`' non conviene; con '`/proc/`' e '`/sys/`' dei sistemi GNU/Linux non si deve. In questi casi serve solo ricordare di ricreare queste directory, nel momento in cui fosse necessario il recupero. Naturalmente non bisogna dimenticare i permessi: '`/tmp/`' 1777_8 e '`/proc/`' 0555_8.

21.5.11 Strumenti per il recupero

L'archiviazione di copie di sicurezza non è sufficiente a garantirsi contro gli incidenti: in che modo si può avviare un elaboratore in cui è appena stato sostituito il disco fisso? Evidentemente, occorre essere più previdenti e predisporre in anticipo gli strumenti necessari per preparare le partizioni di un nuovo disco fisso e per recuperare i dati archiviati precedentemente.

21.6 Riferimenti

«

- Brian Carrier, *Why Recovering a Deleted Ext3 File Is Difficult...*, *http://linux.sys-con.com/node/117909*

[1] **Foremost** In parte di dominio pubblico e in parte GNU GPL

[2] **Ext2 filesystem utilities** GNU GPL

[3] **The Sleuth Kit** IBM Public License *http://www.opensource.org/licenses/ibmpl.php*

[4] **Autopsy** GNU GPL

[5] **E2undel** GNU GPL

[6] **Delsafe** GNU GPL

[7] **Secure-delete** GNU GPL

[8] Ci sono programmi di archiviazione che si comportano così anche se non subiscono compressioni successive.