Linux Ext2fs Undeletion mini-HOWTO

Aaron Crane, *aaronc@pobox.com* v1.3, 2 Febbraio 1999

Immaginate: per gli ultimi tre giorni non avete dormito, non avete mangiato, non avete nemmeno fatto la doccia. La vostra foga di hacker vi ha finalmente ricompensato: avete finito il programma che vi porterà fama mondiale. Tutto quello che dovete fare è farne un tar e metterlo su Metalab. Oh, e cancellare tutti quei file di backup di Emacs. Quindi date `rm * ~`. E, troppo tardi, notate lo spazio di troppo in quel comando. Avete appena cancellato la vostra *magnum opus*! Ma ecco l'aiuto. Questo documento vi presenta una discussione su come recuperare i file cancellati da un filesystem Second Extended. Così, forse, potrete fare uscire quel programma...

Indice

1 Introduzione

Questo mini-howto è un tentativo di fornire suggerimenti su come recuperare file cancellati da un filesystem ext2. Contiene anche una breve discussione su come evitare a priori di cancellare dei file.

La mia intenzione è che sia utile sicuramente per le persone che abbiano appena avuto, diciamo, un piccolo incidente con `rm`;, ma spero che venga letto comunque. Non si sa mai, un giorno queste informazioni vi potrebbero salvare la pelle.

Il testo assume una conoscenza di base dei filesystem UNIX in generale, ma spero che sia accessibile alla maggior parte degli utenti Linux. Se siete dei veri principianti, ho paura che il recupero di file sotto Linux *richieda* una certa conoscenza tecnica, almeno al momento.

Non potrete recuperare dei file cancellati da un filesystem ext2 senza avere almeno accesso in lettura al dispositivo fisico in cui il file si trovava. In generale, questo significa che dovete essere root, ma alcune distribuzioni (come la *Debian GNU/Linux*) hanno impostato un gruppo `disk` i cui membri hanno accesso a tali dispositivi. Vi serve anche `debugfs` dal pacchetto `e2fsprogs`, che dovrebbe essere stato installato dalla distribuzione che avete.

Perché ho scritto questo documento? Più che altro è nato dalla mia esperienza con un particolarmente stupido e disastroso comando `rm -r` da root. Ho cancellato circa 97 file JPEG che mi servivano e non avrei potuto quasi sicuramente recuperare da altre fonti. Usando alcuni suggerimenti utili (vedere la sezione 15 (Ringraziamenti e bibliografia)) e molta testardaggine, ho recuperato 91 file integri. Sono riuscito a recuperare almeno delle parti di 5 dei restanti (abbastanza per vedere quale era la figura). Solo uno è andato perso del tutto, e anche di questo sono quasi sicuro che non ne ho persi più di 1024 byte (anche se sfortunatamente erano all'inizio del file: e non so niente del formato dei file JFIF e ho fatto tutto quello che potevo).

Discuterò più avanti quale percentuale di recupero ci si può aspettare per i file cancellati.

1.1 Revisioni

Le varie revisioni pubbliche di questo documento (e le loro date di pubblicazione) sono le seguenti:

- v1.0 del 18 Gennaio 1997
- v1.1 del 23 Luglio 1997 (vedere la sezione 1.1.1 (Modifiche nella v1.1))
- v1.2 del 4 Agosto 1997 (vedere la sezione 1.1.2 (Modifiche nella v1.2))
- v1.3 del 2 Febbraio 1999 (vedere la sezione 1.1.3 (Modifiche nella v1.3))

1.1.1 Modifiche nella versione 1.1

Quali modifiche sono state fatte in questa versione? Prima di tutto, ho corretto il thinko nell'esempio del recupero file. Grazie a tutti quelli che mi hanno scritto: spero di avere imparato ad essere più attento quando scrivo delle interazioni con i programmi.

Seconda cosa, ho riscritto la discussione sullo schema del filesystem UNIX, in modo che sia, spero, più comprensibile. Non ne ero molto contento all'inizio, e i commenti di alcune persone mi hanno indicato che non era chiaro.

Terzo, ho rimosso il .tar.gz uuencodato di `fsgrab` che era nel mezzo del file. Il programma è adesso disponibile dal

mio sito web `http://pobox.com/~aaronc/tech/fsgrab-1.2.tar.gz`

e presto dovrebbe arrivare su

Metalab `http://metalab.unc.edu/pub/Linux/utils/file/`

(e sui mirror).

Quarta cosa, il documento è stato tradotto nel linguaggio SGML usato dal Linux Documentation Project. Questo linguaggio può essere facilmente convertito in molti altri linguaggi (compreso l'HTML e il LaTeX) per essere visualizzato e stampato facilmente. Uno dei vantaggi di questa cosa è che è più facile avere una bella versione cartacea, un'altro è che il documento ha i riferimenti interni e i link ipertestuali quando viene visualizzato sul web.

1.1.2 Modifiche nella versione 1.2

Questa revisione ha solo delle modifiche incrementali. Consiste principalmente di modifiche suggerite dai lettori, una delle quali è particolarmente importante.

La prima modifica è stata suggerita da Egil Kvaleberg *egil@kvaleberg.no* , che ha fatto notare il comando `dump` in `debugfs`. Grazie ancora, Egil.

La seconda è quella di menzionare l'uso di `chattr` per evitare di cancellare i file importanti. Grazie ad Herman Sujis *H.P.M.Suijs@kub.nl* .

L'abstract è stato rivisto. Sono stati aggiunti degli URL per le organizzazioni ed il software. Sono state apportate altre modifiche minori (compresi alcuni errori di battitura e cose del genere).

1.1.3 Modifiche nella versione 1.3

Anche se questa è la prima versione dopo 17 mesi, è stato modificato molto poco. Questa versione corregge solo alcuni piccoli errori (refusi, URL modificati e così via), ed aggiorna alcune parti che erano diventate datatissime. Ah, e ho cambiato 'Sunsite' in 'Metalab'.

Questa versione sarà l'ultima prima della 2.0, che si spera sarà un Howto completo. Sto lavorando su delle modifiche sostanziali che giustificheranno un incremento nel numero della versione.

1.2 Dove si può trovare questo documento

L'ultima versione pubblica del documento dovrebbe essere sempre disponibile in formato testo nel

sito del Linux Documentation Project `http://metalab.unc.edu/LDP/`

(e nei suoi mirror).

L'ultima versione viene anche tenuta sul

mio sito web `http://pobox.com/~aaronc/`

in diversi formati:

- *sorgente SGML* `http://pobox.com/~aaronc/tech/e2-undel/howto.sgml` . Il sorgente come l'ho scritto, usando il pacchetto SGML tools.
- *HTML* `http://pobox.com/~aaronc/tech/e2-undel/html/` . HTML, generato automaticamente dal sorgente SGML.
- *Testo puro* `http://pobox.com/~aaronc/tech/e2-undel/howto.txt` . Testo puro, anch'esso generato automaticamente dal sorgente SGML.

2 Come fare a non cancellare i file

È vitale ricordare che linux non è come l'MS-DOS quando si parla di recuperare file cancellati. Per l'MS-DOS (e la sua progenie bastarda Windows 95) generalmente recuperare un file è una cosa piuttosto semplice - il 'sistema operativo' (uso il termine in un'accezione molto ampia) comprende addirittura anche un programma che automatizza la maggior parte del processo. Questo non è il caso di Linux.

Quindi, regola numero uno (il primo comandamento, se volete):

TENETE I BACKUP

in qualsiasi caso. Pensate a tutti i vostri dati; forse, come me, avete vari anni di email, contatti, programmi, lavori che si sono accumulati sul vostro computer. Pensate a come vi sentireste se il vostro disco si rompesse, o se – Signore salvaci! – un cracker vi cancellasse tutto l'hard disk. Non è poi così inconsueto: io sono stato in contatto come molte persone in situazioni del genere. Esorto tutti i Linuxisti di buona volontà a uscire e comprare un buon mezzo di backup, pensare ad una buona policy di backup ed a *usarla*. Io personalmente uso un secondo disco su un'altra macchina, e faccio un mirror periodico della mia home directory su di esso via ethernet. Per altre informazioni su come fare una policy di backup, leggete il Frisch (1995) (vedere la sezione 15 (Bibliografia e Ringraziamenti)).

In mancanza di backup, cosa fare? (o anche in presenza di backup: se avete dei dati importanti è, meglio andarci coi piedi di piombo).

Provate ad impostare i permessi dei file importanti a 440 (o meno): negarvi il permesso di cancellare dei file implica una conferma esplicita prima di cancellarli (io comunque trovo che, se sto cancellando ricorsivamente una directory con `rm -r`, interrompo il programma alla prima o seconda richiesta di cancellazione e lo ridò come `rm -rf`).

Un buon trucco per i file selezionati è crearne un hard link in una directory nascosta. Ho sentito una storia di un amministratore di sistema che ha cancellato ripetutamente `/etc/passwd` per errore (quindi praticamente distruggendo metà del sistema). Una delle soluzioni sarebbe stata fare una cosa del genere (da root):

```
# mkdir /.backup
# ln /etc/passwd /.backup
```

Cancellare tutto il contenuto richiede un po' di sforzo: se dite

```
# rm /etc/passwd
```

fare

```
# ln /.backup/passwd /etc
```

la recupererà. Certo, non aiuta nel caso in cui abbiate sovrascritto il file, quindi tenete comunque dei backup.

Su un filesystem ext2 è possibile usare degli attributi ext2 per proteggere i file. Questi attributi possono essere modificati con il comando `chattr`. Esiste un attributo 'append-only' (sola aggiunta): un file con questo attributo non può essere cancellato, ma possono essere solo aggiunti dei dati. Se una directory ha questo attributo, qualsiasi file o directory in essa contenuti può essere modificato come il solito, ma non possono essere cancellati dei file. L'attributo 'append-only' si imposta con il comando

```
$ chattr +a FILE...
```

C'è anche un attributo 'immutable' (non modificabile), che si può impostare o togliere solo da root. Un file o una directory con questo attributo non può essere modificato, cancellato, rinominato e non ci si possono fare hard link. Si imposta così:

```
# chattr +i FILE...
```

L'ext2fs ha anche l'attributo 'undeletable' (non cancellabile) (`+u` in `chattr`). L'intenzione è che se un file con quell'attributo viene cancellato, invece di essere veramente riutilizzato, viene soltanto spostato in un posto sicuro per poter essere cancellato in un secondo tempo. Sfortunatamente questa caratteristica non è stata ancora implementata nei principali kernel; anche se in passato c'è stato un qualche interesse nella sua implementazione, a quanto ne so non è disponibile per le versioni recenti del kernel.

Alcuni suggeriscono di rendere `rm` un alias di shell o una funzione che stia per `rm -i` (che chiede conferma su *qualsiasi* file che cancellate). Addirittura la *distribuzione Red Hat* `http://www.redhat.com/`

lo mette per default a tutti gli utenti, compreso root. Personalmente, non riesco a sopportare il software che non funziona da solo, quindi non faccio così. C'è anche il problema che, prima o poi, userete la modalità a utente singolo, o una shell diversa o addirittura una macchina diversa, dove la funzione `rm` che avete definito non esiste. Se vi aspettate che vi venga chiesta conferma, è facile dimenticarsi dove vi trovate e specificare troppi file al comando di cancellazione. Ugualmente, i vari script e i programmi che sostituiscono `rm` sono, IMHO, molto pericolosi.

Una soluzione leggermente migliore è cominciare ad usare un pacchetto che gestisca la cancellazione riciclabile fornendo un comando che non si chiami `rm`. Per avere ulteriori dettagli su questo argomento, vedere Peek, et al (1993) (vedere la sezione 15 (Bibliografia e ringraziamenti)). Queste, però hanno il problema che tendono ad incoraggiare l'utente a non fare attenzione quando cancellano dei file, invece che starci molto attenti, come spesso è richiesto nei sistemi Unix.

3 Che percentuale di recupero mi posso aspettare?

Dipende. Tra i problemi che insorgono nel recupero dei file su un sistema operativo di alta qualità, multitasking e multi-utente come Linux c'è quello che non si sa mai quando qualcuno vuole scrivere su disco. Quindi, quando al sistema operativo viene detto di cancellare un file, assume che i blocchi usati da quel file sono disponibili per allocare spazio per un altro file (è un esempio specifico di un principio generale di Linux: il kernel e gli strumenti relativi assumono che gli utenti non siano idioti). In generale, più viene usata la macchina, meno è probabile che riusciate a recuperare con successo i vostri file.

Inoltre, la frammentazione del disco può influire sulla facilità di recupero dei file. Se la partizione che contiene i file cancellati è molto frammentata, è poco probabile che riusciate a leggere un file intero.

Se la vostra macchina, come la mia, è effettivamente una workstation a utente singolo, e non stavate facendo nessuna attività che scrivesse su disco al momento fatale della cancellazione dei file, potete aspettarvi una percentuale di recupero vicina a quella che ho indicato prima. Io ne ho recuperati all'incirca il 94% (ed erano file binari, notare). Penso che possiate essere contenti con voi stessi se ne riuscite a recuperare più dell'80%.

4 E allora, come faccio a recuperare un file?

La procedura consiste principalmente nel trovare i dati a basso livello nella partizione, e nel renderli di nuovo visibili al sistema operativo. Ci sono principalmente due modi per farlo: uno è modificare il filesystem esistente in modo da rimuovere l'indicazione di cancellato dagli inode dei file in questione, e sperare che i dati ritornino magicamente a posto. L'altro metodo, che è più sicuro ma più lento, è capire dove i dati si trovano nella partizione e scriverli in un nuovo file su un altro sistema.

Ci sono alcuni passi necessari da fare prima di iniziare a cercare di recuperare i dati: per altri dettagli vedere le sezioni 5 (Smontare il filesystem), 6 (Prepararsi alla modifica diretta degli inode) e 7 (Prepararsi alla scrittura dei dati da qualche altra parte). Per sapere come fare in pratica a recuperare i file, vedere le sezioni 8 (Trovare gli inode cancellati), 9 (Ottenere i dettagli degli inode), 10 (Recuperare i blocchi di dati) e 11 (Modificare gli inode direttamente).

5 Smontare il filesystem

Qualsiasi sia il metodo che avete scelto, il primo passo è smontare il filesystem che contiene i file cancellati. Vi sconsiglio fortemente di andare in qualsiasi modo a operare su un filesystem montato. Questo passo dovrebbe essere fatto *al più presto possibile* dopo che vi siete accorti della cancellazione dei file: prima fate umount, minore è la probabilità che i vostri file vengano sovrascritti.

Il metodo più semplice è questo: assumendo che i file cancellati siano nel filesystem `/usr`, fate

```
# umount /usr
```

Potete comunque voler mantenere disponibili alcune cose di `/usr`, quindi montatelo a sola lettura:

```
# mount -o ro,remount /usr
```

Se i file cancellati erano nella partizione di root, dovrete aggiungere l'opzione `-n` per evitare che mount cerchi di scrivere su `/etc/mtab`:

```
# mount -n -o ro,remount /
```

A parte questo, è possibile che ci sia un altro processo che stia usando il filesystem in questione (che farà fallire l'umount con un errore di 'Resource busy'). Esiste un programma che manda un segnale a qualsiasi processo che sta usando un dato file o punto di mount: `fuser`. Provatelo con la partizione `/usr`:

```
# fuser -v -m /usr
```

elencherà i processi coinvolti. Assumendo che nessuno di essi sia vitale, potete dare

```
# fuser -k -v -m /usr
```

per mandare ad ogni processo un `SIGKILL` (che sicuramente lo ucciderà), o ad esempio

```
# fuser -k -TERM -v -m /usr
```

per mandargli un `SIGTERM` (che normalmente farà uscire il processo in maniera pulita).

6 Prepararsi alla modifica diretta degli inode

Il mio consiglio? Non fate così. Non credo che sia una cosa saggia giocare con un filesystem a livello abbastanza basso da fare funzionare questo metodo. Questo metodo dà anche dei problemi nel senso che si possono recuperare in modo affidabile solo i primi 12 blocchi di ciascun file, quindi se avete dei file lunghi da recuperare dovrete usare l'altro metodo (leggete però la sezione 12 (Questo metodo diventerà più facile in futuro?) per altre informazioni).

Se pensate di dover usare questo metodo, il mio consiglio è di copiare la partizione raw su una partizione diversa, e poi montarla usando il loopback:

```
# cp /dev/hda5 /root/working
# mount -t ext2 -o loop /root/working /mnt
```

(Notate che delle versioni obsolete di `mount` possono avere dei problemi con questo metodo. Se il vostro `mount` non funziona, il mio suggerimento è di aggiornarlo all'ultima versione, o almeno alla 2.7, dato che alcune versioni molto vecchie hanno dei gravi problemi di sicurezza.)

Usare il loopback significa che se e quando distruggere completamente il filesystem, tutto quello che dovrete fare è copiare la partizione grezza al suo posto e ricominciare.

7 Prepararsi alla scrittura dei dati da qualche altra parte

Se scegliete questo metodo, dovete assicurarvi di avere da qualche parte una partizione di recupero, cioè un posto in cui scrivere nuove copie dei file che recuperate. Magari, avete diverse partizioni sul vostro sistema: forse una di root, una per `/usr`, e una per `/home`. Con tutte queste partizioni non dovreste avere problema: create semplicemente una nuova directory in una di esse.

Se avete solo una partizione di root, ed immagazzinate tutto su quella, le cose sono leggermente più strane. Forse avete una partizione MS-DOS o Windows da usare? O avete il driver per il ramdisk nel kernel, forse come modulo? Per usare il ramdisk (assumendo che abbiate un kernel più recente dell'1.3.48), fate così:

```
# dd if=/dev/zero of=/dev/ram0 bs=1k count=2048
# mke2fs -v -m 0 /dev/ram0 2048
# mount -t ext2 /dev/ram0 /mnt
```

Questo crea un volume di ramdisk di 2MB, e lo monta su `/mnt`.

Un breve avvertimento: se usate `kerneld` (o il suo sostituto `kmod` nei kernel 2.2.x e negli ultimi 2.1.x) per caricare e scaricare automaticamente i moduli del kernel, allora non montate il ramdisk finché non avete copiato tutti i file che vi si trovano su un supporto non volatile. Una volta smontato, `kerneld` assume che può smontare il modulo (dopo il normale periodo di attesa), e una volta che ciò accade, la memoria viene riutilizzata da altre parti del kernel, perdendo tutte le ore di fatica che avete fatto per recuperare i vostri dati.

Se avete uno Zip, un Jaz, un LS-120 o una cosa simile, probabilmente sono una buona scelta per metterci una partizione di recupero. Altrimenti, dovrete solo usare i floppy.

L'altra cosa di cui avrete bisogno è un programma che può leggere i dati necessari dal mezzo del dispositivo della partizione. Per iniziare, `dd` adrà bene, ma per leggere diciamo, dal 600esimo MB in una partizione da 800 MB, `dd` insiste a leggere ma ad ignorare i primi 600 MB. Questo prende un tempo piuttosto lungo. La mia soluzione per questo problema è stata scrivere un programma che cercasse nel mezzo della partizione. Si chiama `fsgrab`; ne potete trovare il sorgente sul

mio sito web `http://pobox.com/~aaronc/tech/fsgrab-1.2.tar.gz`

e presto arriverà su

Metalab `http://metalab.unc.edu/pub/Linux/utils/file/`

(e sui suoi mirror). Se volete usare questo metodo, il resto di questo mini-Howto assume che abbiate `fsgrab`.

Se nessuno dei file che state provando a recuperare è più lungo di 12 blocchi (dove un blocco di solito è un kilobyte), allora non avrete bisogno di `fsgrab`.

Se dovete usare `fsgrab` ma non volete scaricarlo e compilarlo, è piuttosto semplice tradurre una linea di comando di `fsgrab` in una di `dd`. Se avete

```
fsgrab -c count -s skip device
```

allora il comando di `dd` corrispondente (ma tipicamente molto più lento) è

```
dd bs=1k if=device count=count skip=skip
```

Vi devo avvisare che, anche se `fsgrab` ha funzionato benissimo per me, non posso prendermi alcuna responsabilità per il suo uso. È stata una soluzione veramente veloce e improvvisata per fare funzionare le cose. Per altri dettagli sulla mancanza di garanzia, vedere la sezione 'Nessuna garanzia' nel file `COPYING` incluso con il programma (la GNU General Public License).

8 Trovare gli inode cancellati

Il prossimo passo è chiedere al filesystem quali inode sono stati liberati recentemente. È un compito che si può portare a termine con `debugfs`; avviatelo con il nome del dispositivo in cui è immagazzinato il filesystem:

```
# debugfs /dev/hda5
```

Se volete modificare direttamente gli inode, aggiungete l'opzione `-w` per abilitare la scrittura sul filesystem:

```
# debugfs -w /dev/hda5
```

Il comando di `debugfs` per trovare gli inode cancellati è `lsdel`, quindi, digitate il comando al prompt:

```
debugfs:  lsdel
```

Dopo molti lamenti e stridore dei meccanismi del disco, un lungo elenco viene mandato in pipe nel vostro pager preferito (il valore di `$PAGER`). Ora vorrete salvarne una copia da qualche altra parte. Se avete `less`, potete digitare `-o` seguito dal nome di un file di output, altrimenti dovrete riuscire a mandare l'output da qualche altra parte. Provate così:

```
debugfs:  quit
# echo lsdel | debugfs /dev/hda5 > lsdel.out
```

Ora, basandovi solo sull'ora di cancellazione, la dimensione, il tipo e i permessi e l'owner espressi come numeri, dovrete riconoscere quali di questi inode cancellati sono quelli che vi servono. Con un po' di fortuna, potrete riconoscerli perché sono tutti quelli che avete cancellato circa cinque minuti fa. Altrimenti, cercate con attenzione su tutta la lista.

Suggerisco che, se possibile, stampiate la lista degli inode che volete recuperare. Vi renderà la vita molto più semplice.

9 Ottenere i dettagli degli inode

`debugfs` ha un comando `stat` che stampa i dettagli su un inode. Date questo comando per ciascun inode nella vostra lista. Ad esempio, se siete interessati all'inode numero 148003, provate questo:

```
debugfs:  stat <148003>
Inode: 148003   Type: regular    Mode:  0644   Flags: 0x0   Version: 1
User:   503   Group:   100   Size: 6065
File ACL: 0    Directory ACL: 0
Links: 0   Blockcount: 12
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x31a9a574 -- Mon May 27 13:52:04 1996
atime: 0x31a21dd1 -- Tue May 21 20:47:29 1996
mtime: 0x313bf4d7 -- Tue Mar  5 08:01:27 1996
dtime: 0x31a9a574 -- Mon May 27 13:52:04 1996
BLOCKS:
594810 594811 594814 594815 594816 594817
TOTAL: 6
```

Se dovete recuperare moltissimi file, vorrete rendere questo passaggio automatico. Assumendo che la lista degli inode da recuperare sia nel file `lsdel.out`, provate con questo:

```
# cut -c1-6 lsdel.out | grep "[0-9]" | tr -d " " > inode
```

Questo nuovo file, `inode`, contiene il numero degli inode da recuperare, uno per linea. Lo salviamo perché molto probabilmente tornerà comodo più tardi. Date semplicemente:

```
# sed 's/^.*$/stat <\0>/' inode | debugfs /dev/hda5 > stats
```

e `stats` conterrà l'output di tutti i comandi `stat`.

10 Recuperare i blocchi di dati

Questa parte può essere molto semplice o molto poco, a seconda che il file che state cercando di recuperare sia più lungo di 12 blocchi.

10.1 File corti

Se il file non era più lungo di 12 blocchi, allora i numeri dei blocchi di tutti i suoi dati sono immagazzinati nell'inode: li potete leggere direttamente dall'output di `stat` per l'inode. Inoltre, `debugfs` ha un comando che fa questo compito automaticamente. Per continuare con l'esempio di prima, ripetuto qui:

```
debugfs:  stat <148003>
Inode: 148003   Type: regular    Mode:  0644   Flags: 0x0   Version: 1
User:   503   Group:   100   Size: 6065
File ACL: 0    Directory ACL: 0
Links: 0   Blockcount: 12
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x31a9a574 -- Mon May 27 13:52:04 1996
atime: 0x31a21dd1 -- Tue May 21 20:47:29 1996
mtime: 0x313bf4d7 -- Tue Mar  5 08:01:27 1996
dtime: 0x31a9a574 -- Mon May 27 13:52:04 1996
BLOCKS:
594810 594811 594814 594815 594816 594817
TOTAL: 6
```

Questo file ha sei blocchi. Dato che questo corrisponde a meno del limite di 12, usiamo `debugfs` per scrivere il file in una nuova posizione, come `/mnt/recovered.000`:

```
debugfs:  dump <148003> /mnt/recovered.000
```

Naturalmente, potete fare lo stesso con `fsgrab`; qui sotto presento un esempio del suo uso:

```
# fsgrab -c 2 -s 594810 /dev/hda5 > /mnt/recovered.000
# fsgrab -c 4 -s 594814 /dev/hda5 >> /mnt/recovered.000
```

Sia con `debugfs` che con `fsgrab`, ci saranno dei caratteri inutili alla fine di `/mnt/recovered.000`, ma non è molto importante. Se volete sbarazzarvene, il metodo più semplice è prendere il campo `Size` dall'inode, e metterlo nell'opzione `bs` della linea di comando di `dd`:

```
# dd count=1 if=/mnt/recovered.000 of=/mnt/resized.000 bs=6065
```

Naturalmente, è possibile che uno o più dei blocchi che compongono il vostro file siano stati sovrascritti. Se è così, siete sfortunati: quel blocco è perso per sempre (ma pensate se aveste smontato il filesystem prima!).

10.2 File più lunghi

I problemi compaiono quando il file è più lungo di 12 blocchi di dati. Qui conviene sapere un po' come sono strutturati dei filesystem UNIX. I dati del file sono immagazzinati in unità chiamate 'blocchi'. Questi blocchi possono essere numerati sequenzialmente. Un file ha anche un 'inode', che è il posto in cui vengono immagazzinate informazioni come il proprietario, i permessi ed il tipo. Come i blocchi, gli inode sono numerati sequenzialmente, anche se con una numerazione separata. Una voce di una directory consiste nel nome del file e nel numero di un inode.

Ma con questo stato di cose è ancora impossibile per il kernel trovare i dati corrispondenti ad una voce di una directory, quindi nell'inode vengono anche registrati i numeri dei blocchi di dati del file, così:

- I numeri di blocchi dei primi 12 blocchi vengono immagazzinati direttamente nell'inode; questi vengono a volte chiamati *blocchi diretti*.
- L'inode contiene il numero di blocco di un *blocco indiretto*. Un blocco indiretto contiene i numeri di blocco di altri 256 blocchi di dati.
- L'inode contiene il numero di blocco di un *blocco doppiamente indiretto*. Un blocco doppiamente indiretto contiene i numeri di blocco di altri 256 blocchi doppiamente indiretti.

- L'inode contiene il numero di blocco di un *blocco triplamente indiretto*. Un blocco triplamente indiretto contiene i numeri di blocco di altri 256 blocchi triplamente indiretti.

Leggetelo ancora: lo so che è complesso, ma è anche importante.

Ora, l'implementazione del kernel per tutte le versioni fino alla 2.0.36 compresa sfortunatamente azzera tutti i blocchi indiretti (e quelli doppiamente indiretti e così via) quando cancella un file; quindi se il file in questione era più lungo di 12 blocchi, non avete alcuna garanzia di riuscire a trovare nemmeno i numeri di tutti i blocchi componenti il file, pensate poi i loro contenuti.

L'unico metodo che sono riuscito a trovare finora è di assumere che il file non era frammentato: se lo era, siete nei guai. Assumendo che il file non fosse frammentato, ci sono diversi schemi per i blocchi dei dati, a seconda di quanti blocchi dati il file usava:

da 0 a 12

I numeri di blocco sono registrati nell'inode, come sopra descritto.

da 13 a 268

Dopo i blocchi diretti, contatene uno per il blocco indiretto, e poi ci sono 256 blocchi di dati.

da 269 a 65804

Come prima, ci sono 12 blocchi diretti, uno indiretto (inutile), e 256 blocchi; questi sono seguiti da un blocco doppiamente indiretto (inutile), e 256 ripetizioni di un blocco indiretto (inutile) e 256 blocchi di dati.

65805 o più

Lo schema dei primi 65804 blocchi sono come nel precedente esempio. Poi segue un blocco triplamente indiretto (inutile) e 256 ripetizioni di una 'sequenza doppiamente indiretta'. Ogni sequenza direttamente indiretta consiste di un blocco doppiamente indiretto (inutile), seguito da 256 ripetizioni di un blocco indiretto (inutile) e 256 blocchi di dati.

Naturalmente, anche se questi numeri di blocchi di dati assunti sono corretti, non c'è garanzia che i dati in essi contenuti siano intatti. Oltre a ciò, più lungo era il file, minore è la possibilità che sia stato scritto nel filesystem senza una frammentazione apprezzabile (eccetto in circostanze speciali).

Da notare che ho sempre assunto che la dimensione dei blocchi sia di 1024 byte, dato che è questo il valore standard. Se avete blocchi più grandi, alcune delle cifre riportate qui sopra sono errate. In particolare, dato che ciascun numero di blocco è lungo 4 byte, dimblocchi/4 è il numero di blocchi che può essere immagazzinato in ciascun blocco indiretto. Quindi, ogni volta che il numero 256 appare nella discussione qui sopra, sostituitelo con dimblocchi/4. Dovrete modificare anche i limiti sul 'numero di blocchi richiesto'.

Esaminiamo un esempio di recupero di un file lungo.

```
debugfs:  stat <1387>
Inode: 148004   Type: regular    Mode:  0644   Flags: 0x0   Version: 1
User:   503   Group:   100   Size: 1851347
File ACL: 0    Directory ACL: 0
Links: 0   Blockcount: 3616
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x31a9a574 -- Mon May 27 13:52:04 1996
atime: 0x31a21dd1 -- Tue May 21 20:47:29 1996
mtime: 0x313bf4d7 -- Tue Mar  5 08:01:27 1996
dtime: 0x31a9a574 -- Mon May 27 13:52:04 1996
BLOCKS:
```

```
8314 8315 8316 8317 8318 8319 8320 8321 8322 8323 8324 8325 8326 8583
TOTAL: 14
```

Sembra esserci una possibilità ragionevole che questo file non sia frammentato: di sicuro, i primi 12 blocchi elencati nell'inode (che sono tutti blocchi di dati) sono contigui; quindi possiamo cominciare a recuperare quelli:

```
# fsgrab -c 12 -s 8314 /dev/hda5 > /mnt/recovered.001
```

L'ultimo blocco elencato nell'inode è 8583. Notate che stiamo ancora bene per quanto riguarda la contiguità del file: l'ultimo blocco di dati che abbiamo scritto era l'8582, che è 8327 + 255. Questo blocco 8583 è un blocco doppiamente indiretto, e quindi possiamo ignorarlo. È seguito da 256 ripetizioni di un blocco indiretto (che possiamo ignorare) seguito da 256 blocchi di dati; quindi, facendo un rapido conto, diamo i seguenti comandi. Notate che saltiamo il blocco indiretto 8583, e il blocco indiretto 8584 immediatamente (speriamo) successivo, e cominciamo a cercare i dati dal blocco 8585.

```
# fsgrab -c 256 -s 8585 /dev/hda5 >> /mnt/recovered.001
# fsgrab -c 256 -s 8842 /dev/hda5 >> /mnt/recovered.001
# fsgrab -c 256 -s 9099 /dev/hda5 >> /mnt/recovered.001
# fsgrab -c 256 -s 9356 /dev/hda5 >> /mnt/recovered.001
# fsgrab -c 256 -s 9613 /dev/hda5 >> /mnt/recovered.001
# fsgrab -c 256 -s 9870 /dev/hda5 >> /mnt/recovered.001
```

Facendo le somme, vediamo che finora abbiamo scritto 12 + (7 * 256) blocchi, che fa 1804. I risultati di 'stat' per l'inode ci avevano dato un conteggio di 3616; sfortunatamente questi blocchi sono lunghi 512 byte (una conseguenza di UNIX), quindi in realtà ci servono 3616/2 = 1808 blocchi di 1024 byte. Ciò significa che ci servono ancora quattro blocchi. L'ultimo blocco scritto era il numero 10125. Come abbiamo fatto finora, saltiamo un blocco indiretto (il numero 10126), e poi possiamo scrivere quegli ultimi quattro blocchi.

```
# fsgrab -c 4 -s 10127 /dev/hda5 >> /mnt/recovered.001
```

Ora, con un po' di fortuna avremo recuperato l'intero file.

11 Modificare gli inode direttamente

Questo metodo è, superficialmente, molto più semplice; comunque, come già detto, non può gestire file più lunghi di 12 blocchi.

Per ogni inode che volete recuperare, dovete impostare lo usage count a uno, e il deletion time a zero: si fa con il comando `mi` (modify inode) di `debugfs`. Un esempio di output, modificando l'inode 148003 di cui sopra:

```
debugfs:  mi <148003>
                          Mode    [0100644]
                       User ID    [503]
                      Group ID    [100]
                          Size    [6065]
                 Creation time    [833201524]
             Modification time    [832708049]
                   Access time    [826012887]
                 Deletion time    [833201524] 0
```

```
              Link count   [0] 1
             Block count   [12]
              File flags   [0x0]
               Reserved1   [0]
                File acl   [0]
           Directory acl   [0]
        Fragment address   [0]
         Fragment number   [0]
           Fragment size   [0]
         Direct Block #0   [594810]
         Direct Block #1   [594811]
         Direct Block #2   [594814]
         Direct Block #3   [594815]
         Direct Block #4   [594816]
         Direct Block #5   [594817]
         Direct Block #6   [0]
         Direct Block #7   [0]
         Direct Block #8   [0]
         Direct Block #9   [0]
        Direct Block #10   [0]
        Direct Block #11   [0]
          Indirect Block   [0]
   Double Indirect Block   [0]
   Triple Indirect Block   [0]
```

Cioè imposto il deletion time a 0 e il link count a 1 e premete semplicemente invio per ciascuno degli altri campi. È garantito, è un po' noioso se avete molti file da recuperare, ma ci potete riuscire... altrimenti potete usare un 'sistema operativo' grafico con un simpatico 'Cestino'.

Tra l'altro: l'output di `mi` si riferisce ad un campo 'Creation time' nell'inode. È una bugia! O comunque, trae in inganno. Il fatto è che su un sistema UNIX non si può dire quando è stato creato un file. Il membro `st_ctime` di una `struct stat` si riferisce al 'tempo di modifica dell'inode', cioè il momento in cui sia stato modificato un dettaglio dell'inode. Qui finisce la lezione di oggi.

Notate che le versioni di `debugfs` più recenti di quella che sto usando probabilmente non includono alcuni dei campi che ho elencato sopra (in particolare, `Reserved1` e (alcuni) campi di fragment).

Una volta modificati gli inode, potete uscire da `debugfs` e dire:

```
# e2fsck -f /dev/hda5
```

L'idea è che ciascuno dei file cancellati è stato recuperato, ma nessuno di essi compare in un elenco di directory. Il programma `e2fsck` può accorgersene, ed aggiungere una voce di directory per ciascun file nella directory `/lost+found` del filesystem (quindi se la partizione è normalmente montata su `/usr`, i file ora saranno in `/usr/lost+found` quando la rimonterete). Tutto quello che rimane da fare è scoprire il nome di ciascun file dai suoi countenuti, e rimetterlo al suo posto corretto nell'albero delle directory.

Quando usate `e2fsck` avrete un output molto interessante, ed alcune domande su quale danno riparare. Rispondete 'sì' a tutto quello che si riferisce alle 'summary information' o agli inode che avete modificato. Tutto il resto viene lasciato a voi, anche se di solito è una buona idea rispondere affermativamente a tutte le domande. Quando `e2fsck` ha finito potete rimontare il filesystem.

In realtà c'è un'alternativa a fare lasciare i file in `/lost+found` ad `e2fsck`: potete usare `debugfs` per creare un link all'inode nel filesystem. Usate il comando `link` di `debugfs` dopo aver modificato l'inode:

```
debugfs:  link <148003> pippo.txt
```

Questo crea un file `pippo.txt` in quella che `debugfs` crede sia la directory corrente: `pippo.txt` sarà il vostro file. Dovrete ancora usare `e2fsck` per aggiustare le informazioni di indice e i conteggi dei blocchi e così via.

12 Diventerà più facile in futuro?

Sì. In effetti, credo che sia già così. Mentre sto scrivendo, i kernel stabili (nella serie 2.0.x) azzerano i blocchi indiretti, cosa che non accade a quelli di sviluppo 2.1.x, né ai 2.2.x. Il kernel 2.2.1 è stato rilasciato da poco (siamo nel Febbraio 1999), ed i rivenditori di Linux probabilmente inizieranno a produrre distribuzioni di Linux che contengano i kernel 2.2.x tra un mese o due. Una volta superata questa limitazione nel kernel di produzione, molte delle mie obiezioni alla tecnica della modifica manuale degli inode spariranno; allo stesso tempo sarà anche possibile usare il comando `dump` nel `debugfs` sui file lunghi.

13 Ci sono strumenti per automatizzare questo processo?

In realtà esistono. Sfortunatamente credo che soffrano dello stesso problema della tecnica della modifica manuale degli inode: i blocchi indiretti non possono essere recuperati. Comunque, data la probabilità che questo non sarà a breve un problema, conviene dare un'occhiata a questi programmi già adesso.

Io ho scritto un programma che si chiama `e2recover`, essenzialmente un wrapper Perl su `fsgrab`. Fa un ragionevole sforzo per gestire i blocchi indiretti azzerati, e sembra funzionare sufficientemente bene finché il filesystem non è frammentato. Imposta anche correttamente i permessi (e quando possibile l'owner) dei file recuperati, e si assicura che essi abbiano la lunghezza corretta.

Originariamente ho scritto `e2recover` per il prossimo aggiornamento globale di questo howto; sfortunatamente ciò significa che molta della documentazione utile per `e2recover` uscirà con quel documento. Potrebbe essere però utile già da adesso, quindi è disponibile sul *mio sito web* `http://pobox.com/~aaronc/tech/e2-undel/` e presto su Metalab.

Scott D. Heavner è l'autore di `lde`, il Linux Disk Editor. Può essere usato sia come editor binario di dischi, sia come l'equivalente di `debugfs` per i filesystem ext2 e minix, ed anche per gli xia (anche se il supporto xia non è più disponibile nei kernel 2.1.x e 2.2.x). Ha alcune caratteristiche per aiutare nel recupero dei file dalla cancellazione, sia per cercare un file nell'elenco dei blocchi sia per fare delle ricerche nel contenuto del disco. Ha anche della buona documentazione sui concetti base dei filesystem, oltre ad un documento su come usarlo per recuperare i file. La versione 2.4 di `lde` è disponibile su

Metalab `http://metalab.unc.edu/pub/Linux/system/filesystems/lde-2.4.tar.gz`

e sui suoi mirror, o sul

sito web dell'autore `http://www.geocities.com/CapeCanaveral/Lab/7731/lde.html` .

Un'altra possibilità è il GNU Midnight Commander, `mc`. Si tratta di uno strumento di gestione di file full-screen, basato per quanto ne so su un certo programma MS-DOS comunemente noto come 'NC'. `mc` supporta il mouse sulla console di Linux e sugli xterm, e fornisce un filesystem virtuale che permette trucchi come fare `cd` dentro un file tar. Tra i suoi filesystem virtuali ce n'è uno per il recupero di file su ext2. Sembra molto comodo, anche se ammetto che non l'ho mai usato personalmente – preferisco i vecchi comandi di shell.

Per usare la sua caratteristica di recupero dei file, si deve configurare il programma con l'opzione `-with-ext2undel`; avrete anche bisogno delle librerie di sviluppo e dei file include presenti nel pacchetto `e2fsprog`. La versione fornita nella

Debian GNU/Linux `http://www.debian.org/` è già configurata così; la stessa cosa può essere per le altre distribuzioni. Una volta compilato il programma, potete dire di fare `cd undel:`dev/hda5/, ed avere un 'elenco di directory' dei file cancellati. Come nella maggior parte degli strumenti di recupero dei file, non gestisce bene i blocchi indiretti azzerati, e tipicamente recupera solo i primi 12k dei file lunghi.

Trovate l'ultima versione sul

sito fto di Midnight Commander `ftp://ftp.nuclecu.unam.mx/Midnight/devel/` .

L'ultima versione non di sviluppo è probabilmente la 4.0; come con il kernel stesso, le versioni di sviluppo *non* sono raccomandate ai non sviluppatori. La lista degli oltre 70 siti da cui scaricarlo è disponibile sul

sito web di Midnight Commander 4 `http://mc.blackdown.org/mc4/` , o provate il

sito ftp ufficiale `ftp://ftp.nuclecu.unam.mx/linux/local/mc-4.0.tar.gz`

(che se la memoria non mi inganna è piuttosto lento).

14 Colophon

Intendo produrre aggiornamenti regolari a questo documento finché avrò tempo per farlo e cose interessanti da dire. Ciò significa che sono avido di conoscere i commenti dei lettori. Posso essere più chiaro? Riuscite a pensare a qualcosa che possa rendere le cose più facili? Esiste qualche nuovo strumento che lo fa in automatico?

In ogni caso, se avete qualcosa da dire, su questo documento o su `fsgrab` o `e2recover`, mandatemi due righe a *aaronc@pobox.com* .

15 Crediti e ringraziamenti

> 'If I have seen farther than others, it is because I was standing on the shoulders of giants.'
> (Isaac Newton)

Questo mini-Howto deriva originariamente da un articolo su *comp.os.linux.misc* di Robin Glover *swrglovr@met.rdg.ac.uk* . Vorrei ringraziare Robin per avermi gentilmente permesso di rielaborare le sue idee in questo mini-Howto. Vorrei inoltre sfruttare questa opportunità per ringraziare ancora tutti quelli che mi hanno scritto sull'Howto. Vale la pena di sforzarsi, se si ricevono commenti positivi.

Alcuni riferimenti bibliografici:

- **Frisch**, Æleen (1995), *Essential System Administration*, seconda edizione, O'Reilly and Associates, Inc., ISBN: 1-56592-127-5.

- **Garfinkel**, Simson, Daniel **Weise** and Steven **Strassmann** (1994), *The Unix-Haters Handbook*, IDG Books, ISBN: 1-56884-203-1. Gran parte di questo libro sono solo le lamentele adolescenziali di persone che pensano che il *loro* sistema operativo era molto meglio di Unix, e molto del resto non si applica se avete uno spazio utente ben scritto come quello GNU... ma c'è del buono, ad esempio vale la pena leggere la discussione su come è facile cancellare i file sotto Unix.

- **Glover**, Robin (31 Jan 1996), *HOW-TO : undelete linux files (ext2fs/debugfs)*, Articolo di usenet su comp.os.linux.misc.

- **Peek**, Jerry, Tim **O'Reilly**, Mike **Loukides** et al (1993), *UNIX Power Tools*, O'Reilly and Associates, Inc./Random House, Inc., ISBN: 0-679-79073-X, seconda edizione 1998.

16 Legalese

Tutti i marchi registrati sono proprietà dei rispettivi possessori. In particolare:

- *MS-DOS* e *Windows* sono marchi registrati della *Microsoft* `http://www.microsoft.com/` .
- *UNIX* è un marchio registrato di *the Open Group* `http://www.opengroup.org/` .
-
- *Linux* è un marchio registrato di Linus Torvalds negli USA ed in alcune altre nazioni.

L'autore si augura ma non richiede che chi intenda vendere copie di questo documento, sia su un mezzo leggibile su computer o da persone, informi lui o il coordinatore dei Linux HOWTO delle sue intenzioni.

Il coordinatore dei Linux HOWTO è al momento Tim Bynum *linux-howto@sunsite.unc.edu* .

Traduzione di Eugenia Franzoni, `eugenia@pluto.linux.it`.