Linux Bootdisk HOWTO

Graham Chapman, grahamc@zeta.org.au v1.02, 25 June 1995

Traduzione di Davide Barbieri, paci@dei.unipd.it. Questo documento descrive come creare i dischi di boot, root/boot e di utilità. Questi dischi possono essere usati come dischi di recovery o per testare nuovi kernel.

Contents

1 Introduzione

1.1 Perché creare un disco di boot?

I dischi di boot di Linux sono utili in diverse situazioni, quali:

- si deve testare un nuovo kernel
- si deve ripristinare il sistema, bloccato per qualche causa, che può essere di varia natura (un boot sector perso o la rottura di una testina del disco)

Ci sono diversi modi per creare i dischi di boot:

- Usare un disco di boot preso da una distribuzione come la Slackware. Questo dovrebbe permettervi di eseguire almeno il boot.
- Usare uno dei pacchetti pensati proprio per creare i dischi da usare come “rescue disks”, o dischi di salvezza.
- Imparare cosa è necessario fare per ogni tipo di disco, e crearselo.

Io ho scelto l’ultima possibilità - imparare come lavorano e crearli da solo. In questo modo, se qualcosa va storto, potrete capire cosa non ha funzionato e lavorarci sopra. In più imparate molto su come Linux lavora. Quando io ho imparato come funzionava il tutto, mi sono creato il Bootkit Rescue Package per mantenere i miei dischi di boot.

Gli utenti esperti potrebbero non trovare molto utile questo documento. Comunque gli utenti nuovi nel sistema di amministrazione del Linux, i quali vogliano proteggersi dalla mancanza di un disco di root o qualcos’altro, potrebbero trovarlo utile.

Una osservazione sulle versioni - questo documento è stato rinnovato per supportare i seguenti pacchetti e versioni:

- Linux 1.2.0
- LILO 0.15

2 Dischi

2.1 Sommario dei tipi di dischi

Ho classificato i dischi relativi al boot in 4 tipi. In questo discorso e attraverso tutto questo documento si usa il termine “disco” per riferirsi ai dischetti, se non è specificato. Gran parte del discorso può essere comunque applicata agli hard-disks.

Un sommario dei tipi di dischi e del loro uso:

boot

Un disco che contiene il kernel e che può eseguire il boot. Il disco può contenere un filesystem e usa un boot loader per eseguire il boot, o può semplicemente contenere il kernel all’inizio del disco. Il disco può essere usato per eseguire il boot del kernel usando un root file system su un altro disco. Questo può essere utile se perdete il vostro boot loader a causa, per esempio, di una installazione incorretta.

root

Un disco con un file system contenente tutto ciò che è necessario per lavorare con un sistema Linux. Non è necessariamente provvisto di un kernel o di un boot loader.

Questo disco può essere usato per eseguire il sistema indipendentemente da qualunque altro disco, una volta che è stato eseguito il boot del kernel. Una caratteristica particolare del kernel permette di eseguire il mount di un altro disco root, con il disco root iniziale che viene automaticamente copiato in memoria.

Potete usare questo tipo di disco per controllare se un disco è rovinato senza eseguirne il mount, o per recuperare un altro disco dopo un errore del disco o la perdita di qualche file.

boot/root

Un disco che è identico al disco root, però contiene un kernel e un boot loader. Può essere usato per eseguire il boot e far partire il sistema. Il vantaggio di questo tipo di disco è che è compatto - tutto si trova in un solo disco. Comunque il continuo aumentare della grandezza di un po' tutto impone che non sempre sia possibile avere tutto su un singolo dischetto.

utility

Un disco che contiene un file system, ma non è fatto per essere usato come root file system. È un disco che contiene qualche utilità addizionale. Potreste usare questo tipo di disco per avere sempre a disposizione questi programmi quando avete problemi di spazio sul vostro disco di root.

Il termine "utility" si riferisce solamente al dischetto, il cui utilizzo potrebbe servire per conservare del software per il recupero del sistema.

2.2 Boot

2.2.1 Introduzione

Tutti i PC iniziano il processo di boot eseguendo un codice residente sulla ROM per caricare il settore del disco di boot dal settore 0, cilindro 0, e provare a eseguirlo. Nella maggior parte dei dischi di boot, il settore 0, cilindro 0 contiene inoltre:

- del codice appartenente a un boot loader come il LILO, che localizza il kernel, lo carica e lo esegue.
- la parte iniziale di un sistema operativo, come il Linux.

Se un kernel del Linux è stato scritto su un dischetto come un "raw device", allora il primo settore sarà il primo settore del kernel stesso, e questo settore continuerà il processo di boot caricando il resto del kernel e eseguendo Linux. Per una più dettagliata descrizione del contenuto del boot sector, fate riferimento alla documentazione che accompagna il lilo-0.15 o maggiore.

Un metodo alternativo per salvare il kernel su un disco di boot è quello di creare un file system, non un root file system, ma semplicemente installando il LILO e le relative opzioni per i comandi da eseguire durante il boot. Per esempio, lo stesso kernel può essere usato per eseguire il boot usando un root file system che si trova sul disco fisso, o che si trova su un dischetto. Questo può essere utile se state cercando di ricostruire il file system del vostro disco fisso, e volete testare ripetutemente i risultati.

2.2.2 Sistemare il puntatore a Root

Il kernel deve in qualche modo ottenere un puntatore al drive e alla partizione di cui deve essere eseguito il mount come root drive. Questo può essere fatto in diversi modi:

- Settando `ROOT_DEV = device` nel makefile del kernel e ricompilando il kernel (per sapere come ricompilare il kernel, leggete le Linux FAQ e guardate nella directory `/usr/src/linux`). Ci sono dei commenti nel makefile del kernel che spiegano quali sono i valori validi per `device`.
- usando l'utility rdev:

  ```
  rdev nomefile device
  ```

  Questo indicherà che il root device del kernel contenuto nel `nomefile` sarà `device`. Per esempio:

  ```
  rdev zImage /dev/sda1
  ```

Questo comando imposta il root device del kernel Image sulla prima partizione SCSI drive.

Ci sono altri modi per usare il comando rdev. Provate:

```
rdev -h
```

e questo vi mostrerà come si usa il comando.

Di solito non c'è bisogno di configurare il root device per usare i dischetti di boot, perché il kernel attualmente usato per eseguire il boot probabilmente punta già al root device. Sarà necessario, comunque, se ottenete il kernel da un'altra macchina, per esempio, da una distribuzione, o se volete che il kernel esegua il boot di un dischetto di root. Non fa male comunque controllare. Per usare rdev per controllare il root device corrente nel kernel, digitate questo comando:

```
rdev <nomefile>
```

È possibile cambiare il root device settato nel kernel attraverso altri mezzi, oltre che tramite rdev. Per dettagli, andate a vedere le FAQ alla fine di questo capitolo.

2.2.3 Copiare il kernel su un dischetto di boot

Una volta che il kernel è stato configurato è necessario copiarlo su un dischetto di boot.

I comandi descritti di seguito (e in tutto l'HOWTO) funzionano regolarmente se si assume che i dischi siano stati formattati. In caso contrario, usate fdformat per formattare i dischetti prima di continuare.

Se non si intende avere un file system sul dischetto di boot, allora il kernel va copiato usato il comando dd, come segue:

```
dd if=nomefile of=device

dove    nomefile e' il nome del kernel
e       device e' il device del dischetto,
        solitamente /dev/fd0
```

Può essere usato anche il comando:

```
cp nomefile device
```

Per esempio:

```
        dd if=zImage of=/dev/fd0
oppure
        cp zImage /dev/fd0
```

Il parametro seek del comando dd NON dovrebbe essere usato. Il file deve essere copiato all'inizio del boot sector (settore 0, cilindro 0), e si ottiene questo omettendo il parametro seek.

Il nome del device può variare. Molti sistemi usano `/dev/fd0` come un nome alternativo per il "vero" nome del device per il drive del dischetto. Per esempio, quando il drive di default (p.e. "drive A:" in DOS) è un drive ad alta densità da 3 1/2, il nome del device sarà `/dev/fd0H1440`, ma di solito `/dev/fd0` punta allo stesso device.

Se si vuole copiare il kernel in un dischetto di boot contenente un filesystem, allora basta eseguire il mount del disco in una directory (p.e. `/mnt`) e usare il comando cp. Per esempio:

```
mount -t ext2 /dev/fd0 /mnt
cp zImage /mnt
umount /mnt
```

Notate che per quasi tutte le operazioni descritte in questo HOWTO, l'utente dovrebbe agire come root.

2.3 Root

2.3.1 Introduzione

Un disco di root contiene un sistema Linux completamente funzionante, ma senza necessariamente contenere un kernel. In altre parole, il disco può non essere di boot, ma una volta che il kernel è stato attivato, il disco di root contiene tutto ciò che è necessario a supportare un sistema Linux completo. Per poter fare questo, il disco deve contenere i minimi requisiti per un sistema Linux:

- Un File system.
- Un set minimo di directory - dev, proc, bin, etc, lib, usr, tmp.
- Un set di base di utility - bash (per avere una shell), ls, cp etc.
- Un set minimo di file di configurazione - rc, inittab, fstab etc.
- Alcune librerie per fornire le funzioni base usate dalle utility.

Ovviamente, qualunque sistema diventa utile solo quando potete fare qualcosa, e un dischetto di root di solito diventa utile solo quando potete fare qualcosa come:

- Controllare il file system di un altro drive, per esempio per controllare il vostro root file system del vostro disco fisso, dovete essere in grado di eseguire il boot di un sistema Linux da un altro drive, così come lo potete fare con un dischetto di root. A questo punto potete eseguire fsck sul vostro disco di root originale finché non è stato eseguito il mount.
- Recuperare tutte le parti del vostro disco di root originale da un backup usando una utility di archiviazione e/o compressione come cpio, tar, gzip and ftape.

2.4 Boot/Root

Essenzialmente la stessa cosa di un disco di root, con in più il kernel e un boot loader come LILO.

Con questa configurazione, un kernel è copiato nel root file system, e LILO poi viene eseguito per installare una configurazione che punta all'imagine del kernel nel disco. Alla partenza, LILO eseguirà il boot del kernel dal disco.

Molti file devono essere copiati su disco affinché questo metodo funzioni. Verranno dati dettagli su questi file e sulla configurazione richiesta da LILO, incluso un esempio, nella sezione intitolata "LILO".

2.4.1 RAM disk e Root Filesystem su Disco

Affinché un dischetto con un root filesystem sia efficiente, bisogna che questo sia in grado di partire da un ramdisk, p.e. un disco emulato in memoria. Questo evita che il sistema sia lento, cosa che accadrebbe usando un disco.

C'è anche un altro vantaggio nell'usare un ramdisk - il kernel del Linux ha la capacità di usare un root ramdisk automaticamente, attraverso il quale copia automaticamente il contenuto del disco di root su un disco RAM, e poi usa il disco RAM come disco root. Questo comporta due grossi vantaggi:

- Il sistema è più veloce.
- Il drive è liberato consentendo di utilizzare altri dischi.

Per usare questa capacità del kernel:

- Il file system sul dischetto deve essere un file system minix o ext2. Il file system ext2 è generalmente quello favorito. Attenti che se avete un kernel più vecchio della versione 1.1.73 allora dovreste guardare i commenti nella sezione intitolata "File System" per vedere se il vostro kernel supporta l'ext2. Se avete un kernel vecchio allora probabilmente dovete usare minix. Questo non causa comunque nessun problema.
- Un disco RAM deve essere configurato nel kernel, e deve essere grande almeno quanto un dischetto.

Un disco RAM può essere configurato nel kernel in diversi modi:

- Togliendo il commento alla macro RAMDISK nel makefile del kernel, in modo che appaia:

  ```
  RAMDISK = -DRAMDISK=1440
  ```

  per definire un ramdisk di 1440 blocchi da 1K, la dimensione di un disco ad alta densità.
- Usando la utility rdev, disponibile su praticamente tutti i sistemi Linux. Questa utility mostra o setta alcuni valori per diverse cose nel kernel, incluso la grandezza desiderata per il disco RAM. Per configurare un disco RAM di 1440 blocchi nel kernel la cui immagine è il file Image, usate:

  ```
  rdev -r zImage 1440
  ```

  questo potrebbe cambiare in futuro, sicuro. Per vedere quale versione di rdev fa questo, usate il comando:

  ```
  rdev -?
  ```

  e dovrebbe mostrare le sue opzioni.
- Usando il boot loader LILO per configurare il disco RAM durante il boot. Questo può essere fatto usando il parametro di configurazione di LILO:

  ```
  ramdisk = 1440
  ```

  per avere un disco RAM di 1440 blocchi di un 1K al boot.
- Interrompendo un boot automatico da parte di LILO e aggiungendo alla riga di comando `ramdisk=1440`. Per esempio, un riga di comando come questa:

  ```
  zImage ramdisk=1440
  ```

  Per maggiori dettagli vedere la sezione LILO.
- Editando il file del kernel e alterando i valori all'inizio del file che registrano la grandezza del disco RAM. Questa è sicuramente l'ultima risorsa da usare, ma è comunque valida. Vedete le FAQ alla fine di questo documento per maggiori dettagli.

Il modo più facile consiste nel configurare LILO, siccome dovete comunque configurare LILO, perché non aggiungere la grandezza del disco RAM?

La configurazione di LILO è brevemente descritta in una sezione intitolata “LILO”, ma è meglio ottenere l'ultima versione stabile di LILO dal più vicino Linux site e leggere la documentazione che lo accompagna.

I dischi RAM possono essere più capienti di un dischetto, e fatti in modo tale da contenere un filesystem largo quanto il disco RAM. Ciò può essere utile per caricare tutto il software necessario a un lavoro di recupero in un disco RAM. Il metodo per creare un disco del genere è descritto nella sezione FAQ, nella domanda “Come posso creare un filesystem in un disco RAM di grossa capienza?”

2.5 Utility

Spesso un disco non è sufficiente per contenere tutto il software necessario per fornire le funzioni necessarie di analisi, riparazione e recupero di un disco danneggiato. Anche se includendo tar, gzip e2fsck, fdisk, Ftape e cosi via, dovreste avere tutto ciò che vi serve.

Questo significa che un set per il recupero richiede un disco di utility, con un file system che contenga ogni altro file necessario. Di questo file system si può eseguire il mount in una directory conveniente, come `/usr`, sul sistema root/boot.

Creare un file system è abbastanza facile, e la spiegazione è data nella sezione titolata “File System”.

3 Componenti

3.1 File System

Il kernel del Linux supporta due file system per i dischi di root che possono essere copiati automaticamente in un disco RAM. Questi sono i file system minix e ext2, dei quali l'ext2 è il migliore. Il supporto per l'ext2 è stato aggiunto tra le versioni 1.1.17 e 1.1.57. Non sono sicuro quale esattamente. Se avete un kernel di una di queste versioni allora guardate nel file `/usr/src/linux/drivers/block/ramdisk.c` e cercate la parola “ext2”.

Se non c'è, allora dovrete usare un file system minix, e quindi il comando “mkfs” per crearlo. (Se usate mkfs, usate l'opzione -i per specificare un maggior numero di “inode” rispetto al valore di dafault; -i 2000 è il valore suggerito).

```
mke2fs -m 0 /dev/fd0
```

Il comando mke2fs riconoscerà automaticamente lo spazio libero e si configurerà automaticamente da solo. Non c'è quindi bisogno di nessun parametro.

Un facile modo per testare il risultato è quello di creare un sistema usando il comando sopra o uno simile, e poi tentare di eseguire il mount del dischetto. Se è un sistema ext2, allora il comando:

```
mount -t ext2 /dev/fd0 /<mount point>
```

dovrebbe funzionare.

3.2 Kernel

3.2.1 Personalizzare il Kernel

Nella maggior parte dei casi sarà possibile copiare il vostro kernel corrente e eseguire il boot da questo. Comunque ci possono essere casi in cui volete compilarne uno diverso.

Una ragione è la grandezza. Il kernel è uno dei file più grossi in un sistema minimizzato, quindi se avete bisogno di creare un disco di root/boot allora dovete anche ridurre la grandezza del kernel il più possibile. Il kernel ora supporta il cambio dei dischetti dopo aver eseguito il boot e prima di eseguire il mount del disco root, così non è più necessario inglobare il kernel nello stesso dischetto come tutto il resto, quindi questi commenti si rivolgono solo a chi decide di creare un disco di root/boot.

Ci sono due modi per ridurre la grandezza del kernel:

- Compilando il kernel con un minimo di supporto per il sistema desiderato. Questo significa lasciare fuori qualcosa di cui non avete bisogno. Il supporto per il networking è una buona cosa da lasciare fuori, come il supporto per i device di cui non avete bisogno quando usate il vostro sistema root/boot.
- Compressandolo, usando l'opzione per il kernel inserita nel makefile:

  ```
  make zImage
  ```

  Fate riferimento alla documentazione inclusa nei sorgenti del kernel per una informazione più aggiornata su come compilare un kernel compresso. Notate che i sorgenti del kernel sono generalmente in `/usr/src/linux`.

Una volta che avete deciso cosa non inserire nel kernel, dovete decidere cosa includere. Probabilmente l'uso più comune che si fa di un disco root/boot è quello di esaminare e recuperare il file system di root rovinato, e per fare questo potreste aver bisogno che il kernel supporti determinate cose.

Per esempio, se i vostri backup sono tutti conservati su nastro e usate Ftape per accedere al drive, allora, se perdete la vostra installazione contenente Ftape, non sarete più in grado di recuperare i vostri backup dal nastro. Dovrete reinstallare Linux, reinstallare Ftape, e poi potrete accedere ai vostri backup.

È certamente preferibile mantenere una copia della stessa utility, così non dovrete perdere tempo installando versioni che non possono leggere i vostri backup.

Il punto è che, qualunque supporto I/O che voi avete aggiunto al vostro kernel per supportare i backup dovrebbe essere anche aggiunto al vostro disco di root/boot. Notate inoltre che il modulo Ftape (o almeno quello che ho io) è piuttosto grosso e non potrà risiedere sul vostro disco di root/boot. Dovrete aggiungerlo su un dischetto di utility - ciò è descritto in una sezione intitolata "AGGIUNGERE DISCHETTI DI UTILITÀ"

Attualmente la procedura per compilare il kernel è descritta nella documentazione che accompagna i sorgenti dello stesso. È abbastanza semplice, incominciate a guardare in `/usr/src/linux`. Attenti che se avete problemi nel compilare il kernel, allora probabilmente è meglio che non tentiate di creare un disco di boot/root.

3.3 Device

La directory `/dev`, che contiene dei file speciali per tutti i device che devono essere usati dal sistema, è d'obbligo per qualsiasi sistema Linux. Questa directory è una directory normale, e può essere creata con il comando mkdir nel modo usuale. I file speciali dei device, però, devono essere creati in un modo particolare, usando il comando mknod.

C'è anche una via più semplice, comunque - copiare la vostra directory `/dev` esistente, e cancellare i device che non vi servono. L'unica accortezza è che i file devono essere copiati usando l'opzione `-R`. Così verrà copiata la directory senza tentare di copiare il contenuto dei file. Notate che usare le lettere minuscole, cioè "`-r`", fa una grossa differenza, perché finirete per copiare tutto il contenuto del vostro disco fisso!

Quindi, state attenti, e usate il comando:

```
cp -dpR /dev /mnt
```

assumendo che sia stato eseguito il mount del dischetto in /mnt. L'opzione dp assicura che i link simbolici siano copiati come link (anziché come i file a cui puntano) e che siano conservati gli attributi originali, preservando anche le proprietà.

Se volete farlo con il metodo più difficile, usate `ls -l` per vedere i numeri 'mayor' e 'minor' dei device che volete, e createli sul dischetto usando mknod.

Molte distribuzioni includono uno script di shell chiamato MAKEDEV nella directory `/dev`. Questo script di shell può essere usato per creare i file, ma probabilmente è più facile copiare quelli esistenti, specialmente per un disco destinato al recupero dei sistemi danneggiati.

In qualunque modo venga copiata la directory, è bene controllare che ogni device speciale di cui avete bisogno sia stato copiato nel dischetto. Per esempio, Ftape usa i device per i nastri, quindi avete bisogno di copiarli tutti.

3.4 Directory

È possibile usare solamente le directory `/dev`, `/proc` e `/etc` per far funzionare un sistema Linux. Non sono sicuro - non l'ho mai provato. Comunque un ragionevole set minimo di directory è il seguente:

`/dev`

Richiesta per motivi di I/O con i device

`/proc`

Richiesta dal comando ps

`/etc`

File di configurazione del sistema

`/bin`

Eseguibili e utility per il sistema

`/lib`

Shared libraries per il supporto run-time

`/mnt`

Un "mount point" per la gestione di un disco in generale

`/usr`

Utility e applicazioni generali

Notate che la struttura delle directory presentata qui è da usare solo nel disco di root. Fate riferimento al Linux File System Standard per maggiori informazioni su come i file system dovrebbero essere strutturati su un sistema Linux "standard".

Quattro di queste directory possono essere create molto facilmente:

- /dev è descritta sopra nella sezione intitolata DEVICE.
- /proc /proc necessita solo di esistere :-). Una volta che la directory è creata con il comando mkdir, non ha più bisogno di niente.
- Per quanto riguarda le altre, /mnt e /usr sono incluse in queste directory soltanto come “mount point” da usare dopo che il sistema root/boot è partito. Ancora una volta, queste directory hanno solo bisogno di essere create.

Le rimanenti 3 directory sono descritti nelle sezione seguenti.

3.4.1 `/etc`

Questa directory deve contenere un certo numero di file di configurazione. Nella maggior parte dei sistemi, questi possono essere suddivisi in 3 gruppi:

- I file richiesti in ogni caso, p.e. rc, `fstab`, `passwd`.
- I file che potrebbero essere richiesti, ma nessuno è troppo sicuro.
- Junk that crept in.

I file che non sono essenziali possono essere identificati con il comando:

```
ls -ltru
```

Questo lista in ordine inverso i file che sono stati modificati ultimamente (dal più recente al più vecchio), quindi se ci sono file che non vengono usati, questi possono essere tralasciati in un eventuale disco di root.

Sui miei dischi di root, io ho al massimo 15 file di configurazione. Questo riduce il mio lavoro con questi tre insiemi di file:

- Quelli necessari alla configurazione per un sistema boot/root:

  ```
  rc      system startup script
  fstab   lista dei file systems di cui fare il mount
  inittab parametri per il processo di init - il
          primo processo a partire al boot del sistema.
  ```

- Quelli che dovrei avere per un sistema boot/root:

  ```
  passwd  lista delle login
  shadow  password
  ```

  Queste dovrebbero essere presenti in un sistema sicuro in modo tale che eseguendo il boot dal dischetto siano rigettate le login non abilitate.

- I rimanenti. I precedenti bastano, quindi lasciamo solo questi per il momento.

A parte questo, ho veramente bisogno di solo 2 file, e ciò che contengono è sorprendentemente poco.

- `rc` dovrebbe contenere:

  ```
  #!/bin/sh
  /etc/mount -av
  /bin/hostname boot_root
  ```

e non ho bisogno di eseguire hostname - semplicemente risulta più simpatico se lo faccio. Anche mount è necessario solamente, per ora, ad eseguire il mount di `/proc` per supportare il comando ps - Linux funziona anche senza hostname.

- `fstab` dovrebbe essere:

```
/dev/fd0        /            ext2    defaults
/proc           /proc        proc    defaults
```

Non penso che la prima riga sia veramente necessaria, ma ho trovato che se la lascio fuori, mount non riesce ad eseguire il mount di /proc.

`inittab` dovrebbe essere a posto così com'è, a meno che non vogliate essere sicuri che gli utenti non possano connettersi attraverso le porte seriali. Per prevenire questo, togliete i commenti a tutte le righe con `/etc/getty` che includono ttys o ttyS alla fine della riga. Lasciate però le porte tty altrimenti non potrete collegarvi tramite la console.

`inittab` definisce cosa il sistema eseguirà in varie situazioni incluso l'inizio della sessione, la chiusura, il passaggio a un modo multi-utente. Bisogna controllare attentamente che i comandi passati a inittab siano presenti e nella directory corretta. Se inserite i vostri comandi nel vostro dischetto di recupero usando il listato della directory di esempio di questo HOWTO, e poi copiate il vostro inittab nel dischetto di recupero senza controllare, allora avrete grosse possibilità che qualcosa vada storto, perché metà delle righe nell'inittab faranno riferimento a programmi inesistenti o residenti in directory sbagliate.

Molti programmi non possono essere spostati da una directory a un'altra altrimenti non riuscriranno a funzionare correttamente. Per esempio sul mio sistema, `/etc/shutdown` chiama il comando `/etc/reboot`. Se sposto reboot in `/bin/reboot`, e poi chiamo il comando shutdown, questo fallirà, perché non può trovare il file reboot.

Per il resto, copiate semplicemente tutti i file di testo nella vostra directory `/etc`, più gli eseguibili nella directory `/etc` di cui non potete essere sicuri se vi serviranno. Come guida, consultate la lista nella sezione "Listato della directory di un Root Disk usando ls-lR" - questo è quello che io ho, così probabilmente sarà sufficiente per voi se copiate solo questi file.

In pratica, un solo file `rc` è restrittivo; molti sistemi ora usano una directory `/etc/rc.d` contenente gli script di shell, ma è probabilmente più semplice copiare la directory inittab e `/etc/rc.d` dal vostro sistema esistente, e controllare poi gli script di shell per togliere quelle parti che non servono in un sistema che partirà da dischetto.

3.4.2 `/bin`

Ecco una directory adatta a contenere tutte le utility extra che vi servono per avere a disposizioni le funzionalità di base; utility come ls, mv, cat, dd ecc. Andate alla sezione intitolata "Esempio di ls-lR su un disco di Boot/Root" per una lista di file che io ho messo nella mia directory `/bin` del mio disco di boot/root. Noterete che non contengono utility per recuparare i backup, come cpio, tar, gzip etc. Questo perché io ho messo queste utility su un dischetto separato, per salvare spazio sul disco di root/boot. Una volta che ho fatto partire il sistema dal mio disco di root/boot, esso si copia in memoria come ramdisk lasciando il disk drive libero per un altro dischetto: il dischetto utility! Di solito io ne faccio il mount in `/usr`.

La creazione di un disco di utility è descritta successivamente nella sezione intitolata "Aggiungere un Dischetto di Utility".

3.4.3 `/lib`

Due librerie sono necessarie per il funzionamento di molte utility sotto Linux:

- ld.so
- libc.so.4

Se non sono nella directory `/lib` allora il sistema non sarà in grado di partire. Se siete fortunati potreste vedere un messaggio di errore che vi dice perché.

Queste dovrebbero essere presenti nella vostra directory `/lib`. Notate che `libc.so.4` può essere un link simbolico a un libreria libc con la versione nel nome del file. Se usate il comando:

```
ls -l /lib
```

dovreste vedere qualcosa del genere:

```
libc.so.4 -> libc.so.4.5.21
```

In questo caso, la libreria libc che vi serve è la libc.so.4.5.21.

3.5 LILO

3.5.1 Introduzione

Un dischetto di root/boot perché sia utilizzabile, deve essere in grado di eseguire il boot. Per ottenere ciò, la via più semplice (forse l'unica?) è quella di installare un boot loader, che è una parte di codice eseguibile salvata sul settore 0, cilindro 0 del disco. Andate alla sezione intitolata "Dischetti di Boot" per saperne di più sui processi di boot.

LILO è un boot loader disponibile su ogni Linux site. Permette di configurare il boot loader, compreso:

- Quale device deve essere "montato" come drive root.
- Se deve essere usato un ramdisk.

3.5.2 Esempio di Configurazione di LILO

È un modo molto conveniente per specificare come il kernel deve comportarsi al momento del boot. Il mio file di configurazione per LILO 0.15 sul disco di root/boot è:

```
boot = /dev/fd0
install = ./mnt/boot.b
map = ./mnt/lilo.map
delay = 50
message = ./mnt/lilo.msg
timeout = 150
compact
image = ./mnt/zImage
        ramdisk = 1440
        root = /dev/fd0
```

Notate che boot.b, lilo.msg e il kernel devono essere copiati sul dischetto usando un comando del tipo:

```
cp /boot/boot.b ./mnt
```

Se questo non viene fatto, LILO non potrà essere eseguito correttamente durante il boot se l'hard disk non è disponibile, e c'è da perdere tempo nel creare un dischetto di salvezza che richieda un hard disk per partire.

Io eseguo LILO con il comando:

```
/sbin/lilo -C <configfile>
```

Lo eseguo da una directory contenente la directory mnt nella quale ho “messo” il disco. Questo significa che sto dicendo a LILO di installare un boot loader nel device di boot (/dev/fd0 in questo caso), di eseguire il boot del kernel nella directory principale del disco.

Ho anche specificato che voglio che il root device sia il dischetto, e che voglio un RAM disk di 1440 blocchi da 1K, la stessa grandezza del dischetto. Siccome ho creato un dischetto con un file system di tipo ext2, ho fatto tutto ciò che dovevo fare affinché il sistema usi come root device il ramdisk, e copi il contenuto del dischetto in quest’ultimo.

Le caratteristiche di un ram disk sono descritte più approfonditamente nella sezione intitolata “RAM DISK E SISTEMI BOOT/ROOT”.

È anche utile considerare la possibilità di usare il parametro “singolo” per fare in modo che Linux esegua il boot in modalità utente-singolo. Questo può essere utile per prevenire la connessione di utenti attraverso le porte seriali.

Uso inoltre le opzioni “DELAY” “MESSAGE” e “TIMEOUT” in modo che quando il sistema parte dal disco, LILO mi darà l’opportunità di inserire dei comandi da passare al kernel. Non ne ho bisogno ora come ora, ma potrebbe tornarmi utile se volessi usare un’altro root device o eseguire il mount di un file system in sola lettura.

Il file di messaggio che io uso è così fatto:

```
Linux Boot/Root Diskette
========================

Inserisci un linea di comando del tipo:

          zImage [ command-line options]

Se non scrivi niente linux partira‘ dopo 15 secondi.
```

Mi ricorda semplicemente quali sono le mie scelte.

I lettori sono invitati a leggere la documentazione di LILO attentamente prima di tentare di installare qualcosa. È facile rovinare le partizioni se usate il parametro “`boot =`” in modo sbagliato. Se non siete esperti non usate LILO finché non siete sicuri di avere capito e avete controllato 3 volte i vostri parametri.

Notate che dovete rieseguire lilo ogni volta che cambiate kernel, così LILO può settare la sua mappa dei file per descrivere correttamente il nuovo file del kernel. È di fatto possibile sostituire il file del kernel con uno quasi identico senza rieseguire LILO, ma è molto meglio non rischiare - se cambiate kernel, rieseguite LILO.

3.5.3 Rimuovere LILO

Finché sono sull’argomento LILO è bene che vi dica un’altra cosa: se avete installato lilo su un drive contenente DOS, potete sempre ripristinare il boot sector con il comando:

```
FDISK /MBR
```

dove MBR sta per “Master Boot Record”. Alcuni non vedono di buon occhio questo metodo, però funziona.

3.5.4 Opzioni Utili di LILO

LILO ha molte opzioni utili le quali sono difficili da ricordare quando si deve creare un disco di boot:

- opzioni di linea di comando - potete dare delle opzioni di linea di comando per specificare il root device, la grandezza del ramdisk, parametri per device speciali o altre cose. Se usate l'opzione DELAY=nn nel file di configurazione, LILO si fermerà permettendovi di selezionare un'immagine del kernel con cui eseguire il boot e di passare qualunque opzione vogliate al kernel. Per esempio:

  ```
  zImage aha152x=0x340,11,3,1 ro
  ```

  passa il parametro aha152x al driver per i dischi scsi aha152x (ammesso che il kernel sia stato compilato con il supporto per questi dischi) e chiede che venga eseguito il mount del root file system in sola lettura.

- L'opzione "lock" - questa opzione chiede a LILO di salvare la linea di comando che si inserisce come la linea di comando di default da usare in tutti i boot successivi. Questa opzione è particolarmente utile quando avete un device che non può essere riconosciuto automaticamente. Usando l'opzione "lock" evitate di inserire i parametri del device ogni volta che accendete il computer. Per esempio:

  ```
  zImage aha152x=0x340,11,3,1 root=/dev/sda8 ro lock
  ```

- L'opzione di configurazione APPEND - permette di memorizzare la stringa con i parametri per un device nella configurazione, come alternativa all'opzione "lock". Attenti all'uso delle virgolette. Per esempio:

  ```
  APPEND = "aha152x=0x340,11,3,1"
  ```

- DELAY - ferma il boot automatico per un determinato tempo, in modo da dare il tempo all'utente di inserire una linea di comando, magari diversa da quella di default.

4 Esempi

4.1 Lista delle Directory

Questa lista contiene i file e le directory presenti nei miei root e utility disk. Queste liste sono fornite solo come esempio dei file inclusi per creare un sistema funzionante. Queste dischetti sono stati creati usando il pacchetto Bootkit, che copia sul dischetto solo i file che voi volete che siano copiati.

4.1.1 Listato della directory di un Root Disk usando ls-lR

La lista si riferisce ad un disco di root di cui è stato fatto il mount in /mnt:

```
total 27
drwx------   2 root     root         1024 Jun 11 23:23 bin/
drwxr-xr-x   2 root     root         3072 Jun 11 23:24 dev/
drwxr-xr-x   3 root     root         1024 May 30 06:38 etc/
drwxr-xr-x   2 root     root         1024 Jun 11 23:24 home/
drwxr-xr-x   2 root     root         1024 Jun 11 23:24 lib/
drwxr-xr-x   2 root     root        12288 Jun 11 23:23 lost+found/
drwxr-xr-x   2 root     root         1024 Jun 11 23:24 mnt/
drwxr-xr-x   2 root     root         1024 Jun 11 23:24 proc/
drwxr-xr-x   2 root     root         1024 May 30 05:56 root/
drwxr-xr-x   2 root     root         1024 Jun  3 23:39 sbin/
drwxr-xr-x   2 root     root         1024 Jun 11 23:24 tmp/
drwxr-xr-x   3 root     root         1024 May 30 05:48 usr/
drwxr-xr-x   2 root     root         1024 Jun 11 23:24 util/
drwxr-xr-x   5 root     root         1024 May 30 05:58 var/
```

```
/mnt/bin:
total 664
-rwxr-xr-x   1 root     root       222208 Sep  7  1992 bash*
-rwxr-xr-x   1 root     other        4376 Sep  8  1992 cat*
-rwxr-xr-x   1 root     other        5088 Sep  4  1992 chmod*
-rwxr-xr-x   1 root     other        4024 Sep  4  1992 chown*
-rwxr-xr-x   1 root     other       12104 Sep  4  1992 cp*
-rwxr-xr-x   1 root     other        4376 Sep  5  1992 cut*
-rwxr-xr-x   1 root     other        7592 Sep  4  1992 dd*
-rwxr-xr-x   1 root     other        4656 Sep  4  1992 df*
-rwxr-xr-x   1 root     root        37892 May  5  1994 e2fsck*
-rwx--x--x   1 root     root        14396 Sep 20  1992 fdisk*
-r-x--x--x   1 bin      bin          3536 Feb 19 19:14 hostname*
-rwxr-xr-x   1 root     other        5292 Sep  4  1992 ln*
-rws--x--x   1 root     root        24352 Jan 16  1993 login*
-rwxr-xr-x   1 root     other        4104 Sep  4  1992 mkdir*
-rwxr-xr-x   1 root     root        21508 May  5  1994 mke2fs*
-rwxr-xr-x   1 root     other        3336 Sep  4  1992 mknod*
-rwx--x--x   1 root     root         2432 Sep 20  1992 mkswap*
-rwxr-xr-x   1 root     root         9596 Jun 10 22:12 mount*
-rwxr-xr-x   1 root     other        6724 Sep  4  1992 mv*
-rwxr-xr-x   1 root     root        11132 Apr 10  1993 ps*
-rwxr-xr-x   1 root     other        5056 Sep  4  1992 rm*
-rwxr-xr-x   1 root     root       222208 Sep  7  1992 sh*
-rws--x--x   1 root     root        16464 Jan 16  1993 su*
-rwxr-xr-x   1 root     root         1204 Sep 17  1992 sync*
-rwxr-xr-x   1 root     root         6188 Apr 17  1993 umount*

/mnt/dev:
total 72
-rwxr-xr-x   1 root     root         8331 Mar 14  1993 MAKEDEV*
lrwxrwxrwx   1 root     root            4 Jun 11 23:24 console -> tty0
crw-rw-rw-   1 root     tty        5,  64 Apr  1  1993 cua0
crw-rw-rw-   1 root     tty        5,  65 Mar 19 19:35 cua1
crw-rw-rw-   1 root     tty        5,  66 Apr 10  1993 cua2
crw-rw-rw-   1 root     tty        5,  67 Apr 10  1993 cua3
brw-r--r--   1 root     root       2,   0 Aug 29  1992 fd0
brw-r--r--   1 root     root       2,  12 Aug 29  1992 fd0D360
brw-r--r--   1 root     root       2,  16 Aug 29  1992 fd0D720
brw-r--r--   1 root     root       2,  28 Aug 29  1992 fd0H1440
brw-r--r--   1 root     root       2,  12 Aug 29  1992 fd0H360
brw-r--r--   1 root     root       2,  16 Aug 29  1992 fd0H720
brw-r--r--   1 root     root       2,   4 Aug 29  1992 fd0d360
brw-r--r--   1 root     root       2,   8 Jan 15  1993 fd0h1200
brw-r--r--   1 root     root       2,  20 Aug 29  1992 fd0h360
brw-r--r--   1 root     root       2,  24 Aug 29  1992 fd0h720
brw-r--r--   1 root     root       2,   1 Aug 29  1992 fd1
brw-r--r--   1 root     root       2,  13 Aug 29  1992 fd1D360
brw-r--r--   1 root     root       2,  17 Aug 29  1992 fd1D720
brw-r--r--   1 root     root       2,  29 Aug 29  1992 fd1H1440
brw-r--r--   1 root     root       2,  13 Aug 29  1992 fd1H360
brw-r--r--   1 root     root       2,  17 Aug 29  1992 fd1H720
brw-r--r--   1 root     root       2,   5 Aug 29  1992 fd1d360
brw-r--r--   1 root     root       2,   9 Aug 29  1992 fd1h1200
brw-r--r--   1 root     root       2,  21 Aug 29  1992 fd1h360
```

```
brw-r--r--   1 root     root       2,  25 Aug 29  1992 fd1h720
brw-r-----   1 root     root       3,   0 Aug 29  1992 hda
brw-r-----   1 root     root       3,   1 Aug 29  1992 hda1
brw-r-----   1 root     root       3,   2 Aug 29  1992 hda2
brw-r-----   1 root     root       3,   3 Aug 29  1992 hda3
brw-r-----   1 root     root       3,   4 Aug 29  1992 hda4
brw-r-----   1 root     root       3,   5 Aug 29  1992 hda5
brw-r-----   1 root     root       3,   6 Aug 29  1992 hda6
brw-r-----   1 root     root       3,   7 Aug 29  1992 hda7
brw-r-----   1 root     root       3,   8 Aug 29  1992 hda8
brw-r-----   1 root     root       3,  64 Aug 29  1992 hdb
brw-r-----   1 root     root       3,  65 Aug 29  1992 hdb1
brw-r-----   1 root     root       3,  66 Aug 29  1992 hdb2
brw-r-----   1 root     root       3,  67 Aug 29  1992 hdb3
brw-r-----   1 root     root       3,  68 Aug 29  1992 hdb4
brw-r-----   1 root     root       3,  69 Aug 29  1992 hdb5
brw-r-----   1 root     root       3,  70 Aug 29  1992 hdb6
brw-r-----   1 root     root       3,  71 Aug 29  1992 hdb7
brw-r-----   1 root     root       3,  72 Aug 29  1992 hdb8
crw-r-----   1 root     kmem       1,   2 Aug 29  1992 kmem
crw-rw-rw-   1 root     root       6,   0 Aug 29  1992 lp0
crw-rw-rw-   1 root     root       6,   1 Aug 29  1992 lp1
crw-rw-rw-   1 root     root       6,   2 Aug 29  1992 lp2
crw-r-----   1 root     sys        1,   1 Aug 29  1992 mem
lrwxrwxrwx   1 root     root            4 Jun 11 23:24 mouse -> cua1
crw-rw-rw-   1 root     root      27,   4 Jul 31  1994 nrft0
crw-rw-rw-   1 root     root      27,   5 Jul 31  1994 nrft1
crw-rw-rw-   1 root     root      27,   6 Jul 31  1994 nrft2
crw-rw-rw-   1 root     root      27,   7 Jul 31  1994 nrft3
crw-------   1 root     root       9, 128 Jan 23  1993 nrmt0
crw-rw-rw-   1 root     root       1,   3 Aug 29  1992 null
crw-r-----   1 root     root       6,   0 Aug 29  1992 par0
crw-r-----   1 root     root       6,   1 Aug 29  1992 par1
crw-r-----   1 root     root       6,   2 Aug 29  1992 par2
crw-r-----   1 root     root       1,   4 Aug 29  1992 port
crw-rw-rw-   1 root     root       4, 128 Jun 10 00:10 ptyp0
crw-rw-rw-   1 root     root       4, 129 Apr 10 14:51 ptyp1
crw-rw-rw-   1 root     root       4, 130 Aug 21  1994 ptyp2
crw-rw-rw-   1 root     root       4, 131 Apr 12  1993 ptyp3
crw-rw-rw-   1 root     tty        4, 132 Jan  3  1993 ptyp4
crw-rw-rw-   1 root     tty        4, 133 Jan  3  1993 ptyp5
crw-rw-rw-   1 root     tty        4, 134 Jan  3  1993 ptyp6
crw-rw-rw-   1 root     tty        4, 135 Jan  3  1993 ptyp7
crw-rw-rw-   1 root     tty        4, 136 Jan  3  1993 ptyp8
crw-rw-rw-   1 root     tty        4, 137 Jan  3  1993 ptyp9
crw-rw-rw-   1 root     tty        4, 138 Jan  3  1993 ptypa
crw-rw-rw-   1 root     tty        4, 139 Jan  3  1993 ptypb
crw-rw-rw-   1 root     tty        4, 140 Jan  3  1993 ptypc
crw-rw-rw-   1 root     tty        4, 141 Jan  3  1993 ptypd
crw-rw-rw-   1 root     tty        4, 142 Jan  3  1993 ptype
crw-rw-rw-   1 root     tty        4, 143 Jan  3  1993 ptypf
brw-rw----   1 root     root       1,   0 Jun  8 18:49 ram
crw-rw-rw-   1 root     root      27,   0 Jul 31  1994 rft0
crw-rw-rw-   1 root     root      27,   1 Jul 31  1994 rft1
crw-rw-rw-   1 root     root      27,   2 Jul 31  1994 rft2
```

```
crw-rw-rw-   1 root     root      27,   3 Jul 31  1994 rft3
crw-------   1 root     root       9,   0 Jan 23  1993 rmt0
brw-r-----   1 root     root       8,   0 Aug 29  1992 sda
brw-r-----   1 root     root       8,   1 Aug 29  1992 sda1
brw-r-----   1 root     root       8,   2 Aug 29  1992 sda2
brw-r-----   1 root     root       8,   3 Aug 29  1992 sda3
brw-r-----   1 root     root       8,   4 Aug 29  1992 sda4
brw-r-----   1 root     root       8,   5 Aug 29  1992 sda5
brw-r-----   1 root     root       8,   6 Aug 29  1992 sda6
brw-r-----   1 root     root       8,   7 Aug 29  1992 sda7
brw-r-----   1 root     root       8,   8 Aug 29  1992 sda8
brw-r-----   1 root     root       8,  16 Aug 29  1992 sdb
brw-r-----   1 root     root       8,  17 Aug 29  1992 sdb1
brw-r-----   1 root     root       8,  18 Aug 29  1992 sdb2
brw-r-----   1 root     root       8,  19 Aug 29  1992 sdb3
brw-r-----   1 root     root       8,  20 Aug 29  1992 sdb4
brw-r-----   1 root     root       8,  21 Aug 29  1992 sdb5
brw-r-----   1 root     root       8,  22 Aug 29  1992 sdb6
brw-r-----   1 root     root       8,  23 Aug 29  1992 sdb7
brw-r-----   1 root     root       8,  24 Aug 29  1992 sdb8
brw-------   1 bin      bin        8,  32 Jun 30  1992 sdc
brw-------   1 bin      bin        8,  33 Jun 30  1992 sdc1
brw-------   1 bin      bin        8,  34 Jun 30  1992 sdc2
brw-------   1 bin      bin        8,  35 Jun 30  1992 sdc3
brw-------   1 bin      bin        8,  36 Jun 30  1992 sdc4
brw-------   1 bin      bin        8,  37 Jun 30  1992 sdc5
brw-------   1 bin      bin        8,  38 Jun 30  1992 sdc6
brw-------   1 bin      bin        8,  39 Jun 30  1992 sdc7
brw-------   1 bin      bin        8,  40 Jun 30  1992 sdc8
brw-------   1 bin      bin        8,  48 Jun 30  1992 sdd
brw-------   1 bin      bin        8,  49 Jun 30  1992 sdd1
brw-------   1 bin      bin        8,  50 Jun 30  1992 sdd2
brw-------   1 bin      bin        8,  51 Jun 30  1992 sdd3
brw-------   1 bin      bin        8,  52 Jun 30  1992 sdd4
brw-------   1 bin      bin        8,  53 Jun 30  1992 sdd5
brw-------   1 bin      bin        8,  54 Jun 30  1992 sdd6
brw-------   1 bin      bin        8,  55 Jun 30  1992 sdd7
brw-------   1 bin      bin        8,  56 Jun 30  1992 sdd8
brw-------   1 bin      bin        8,  64 Jun 30  1992 sde
brw-------   1 bin      bin        8,  65 Jun 30  1992 sde1
brw-------   1 bin      bin        8,  66 Jun 30  1992 sde2
brw-------   1 bin      bin        8,  67 Jun 30  1992 sde3
brw-------   1 bin      bin        8,  68 Jun 30  1992 sde4
brw-------   1 bin      bin        8,  69 Jun 30  1992 sde5
brw-------   1 bin      bin        8,  70 Jun 30  1992 sde6
brw-------   1 bin      bin        8,  71 Jun 30  1992 sde7
brw-------   1 bin      bin        8,  72 Jun 30  1992 sde8
crw-rw-rw-   1 root     root       5,   0 Apr 16  1994 tty
crw-rw-rw-   1 grahamc  other      4,   0 Jun 11 23:21 tty0
crw--w--w-   1 root     root       4,   1 Jun 11 23:23 tty1
crw-rw-rw-   1 root     root       4,   2 Jun 11 23:21 tty2
crw-rw-rw-   1 root     root       4,   3 Jun 11 23:21 tty3
crw-rw-rw-   1 root     other      4,   4 Jun 11 23:21 tty4
crw-rw-rw-   1 root     other      4,   5 Jun 11 23:21 tty5
crw-rw-rw-   1 root     root       4,   6 Jun 11 23:21 tty6
```

```
crw--w--w-   1 grahamc  other      4,   7 Apr 15  1993 tty7
crw--w--w-   1 root     root       4,   8 Apr 15  1993 tty8
crw-rw-rw-   1 root     root       4,  64 Mar 30  1993 ttyS0
crw-rw-rw-   1 root     users      4,  65 Mar 31  1993 ttyS1
crw-rw-rw-   1 root     root       4,  66 Jan 23  1980 ttyS2
crw-rw-rw-   1 root     root       4, 192 Jun 10 00:10 ttyp0
crw-rw-rw-   1 root     root       4, 193 Apr 10 14:51 ttyp1
crw-rw-rw-   1 root     root       4, 194 Aug 21  1994 ttyp2
crw-rw-rw-   1 root     root       4, 195 Apr 12  1993 ttyp3
crw-rw-rw-   1 root     tty        4, 196 Jan  3  1993 ttyp4
crw-rw-rw-   1 root     tty        4, 197 Jan  3  1993 ttyp5
crw-rw-rw-   1 root     tty        4, 198 Jan  3  1993 ttyp6
crw-rw-rw-   1 root     tty        4, 199 Jan  3  1993 ttyp7
crw-rw-rw-   1 root     tty        4, 200 Jan  3  1993 ttyp8
crw-rw-rw-   1 root     tty        4, 201 Jan  3  1993 ttyp9
crw-rw-rw-   1 root     tty        4, 202 Jan  3  1993 ttypa
crw-rw-rw-   1 root     tty        4, 203 Jan  3  1993 ttypb
crw-rw-rw-   1 root     tty        4, 204 Jan  3  1993 ttypc
crw-rw-rw-   1 root     tty        4, 205 Jan  3  1993 ttypd
crw-rw-rw-   1 root     tty        4, 206 Jan  3  1993 ttype
crw-rw-rw-   1 root     tty        4, 207 Jan  3  1993 ttypf
-rw-------   1 root     root        63488 Mar 14  1993 ttys0
crw-rw-rw-   1 root     root       4,  67 Oct 14  1992 ttys3
crw-r--r--   1 root     root       1,   5 Aug 29  1992 zero

/mnt/etc:
total 108
-rw-r--r--   1 root     root           94 May 30 06:15 fstab
-rwx------   1 root     root        25604 Mar 17  1993 getty*
-rw-------   1 root     root          566 Dec 30  1992 gettydefs
-rw-rw-r--   1 root     shadow        321 Oct  3  1994 group
-rwxr-xr-x   1 bin      bin          9220 Mar 17  1993 halt*
-rw-r--r--   1 root     root           26 Feb 19 19:07 host.conf
-rw-r--r--   1 root     root          506 Feb 19 19:07 hosts
-rwxr-xr-x   1 bin      bin         17412 Mar 17  1993 init*
-rw-r--r--   1 root     root         1354 Jun  3 23:42 inittab
-rwxr-xr-x   1 root     root         1478 Mar 17 18:29 issue*
-rw-rw----   1 root     shadow       5137 Dec  4  1992 login.defs
-rw-r--r--   1 sysadmin bin            42 Mar 17 18:30 motd
-rw-r--r--   1 root     shadow        525 Jun 11 23:24 passwd
-rwxr-xr-x   1 root     root         1476 Aug 17  1994 profile*
-rw-r--r--   1 root     root          715 Feb 19 19:02 protocols
drwxr-xr-x   2 root     root         1024 May 30 06:05 rc.d/
-rwxr-xr-x   1 bin      bin          9220 Mar 17  1993 reboot*
-r--r--r--   1 bin      bin            57 Nov 28  1992 securetty
-rw-r--r--   1 root     root         3316 Feb 19 19:01 services
-rwxr-xr-x   1 bin      bin         13316 Mar 17  1993 shutdown*
-rwxr-xr-x   1 root     root         3212 Apr 17  1993 swapoff*
-rwxr-xr-x   1 root     root         3212 Apr 17  1993 swapon*
-rw-r--r--   1 root     root          817 Jun 11 23:23 termcap
-rwxr-xr-x   1 root     root         6188 Apr 17  1993 umount*
-rw-r--r--   1 root     root        12264 Jun 11 23:22 utmp
-rw-r--r--   1 root     root           56 Jun 11 23:22 wtmp

/mnt/etc/rc.d:
```

```
total 4
-rwxr-xr--   1 root     root          450 May 30 06:05 rc.0*
-rwxr-xr--   1 root     root          390 May 30 06:05 rc.K*
-rwxr-xr--   1 root     root          683 May 30 06:06 rc.M*
-rwxr-xr--   1 root     root          498 Jun 11 18:44 rc.S*

/mnt/home:
total 0

/mnt/lib:
total 287
-rwxr-xr-x   1 root     root        17412 Jun 11 23:24 ld.so*
lrwxrwxrwx   1 root     root           14 Jun 11 23:24 libc.so.4 -> libc.so.4.5.21*
-rwxr-xr-x   1 root     root       623620 May 22  1994 libc.so.4.5.21*

/mnt/lost+found:
total 0

/mnt/mnt:
total 0

/mnt/proc:
total 0

/mnt/root:
total 0

/mnt/sbin:
total 15
-rwxr-xr-x   1 root     root        16885 Jun 13  1994 update*

/mnt/tmp:
total 0

/mnt/usr:
total 1
drwxr-xr-x   2 root     root         1024 May 30 05:49 bin/

/mnt/usr/bin:
total 217
-rwxr-xr-x   1 root     root         1560 Sep 17  1992 basename*
-rws--x--x   1 root     root         8232 Jan 16  1993 chsh*
-rwxr-xr-x   1 root     root         1308 Jan 23  1980 clear*
-rwxr-xr-x   1 root     other       91136 Sep  4  1992 elvis*
-rwxr-xr-x   1 root     root        13252 Sep 17  1992 ls*
-rwxr-xr-x   1 bin      bin         21504 Oct  2  1992 more*
-rwxr-xr-x   1 root     other       91136 Sep  4  1992 vi*

/mnt/util:
total 0

/mnt/var:
total 3
drwxr-xr-x   2 root     root         1024 May 30 05:58 adm/
drwxr-xr-x   2 root     root         1024 Jun 11 23:24 logs/
```

```
drwxr-xr-x   2 root     root         1024 Jun 11 23:24 run/

/mnt/var/adm:
total 0
-rw-r--r--   1 root     root            0 May 30 05:58 utmp
-rw-r--r--   1 root     root            0 May 30 05:58 wtmp

/mnt/var/logs:
total 0

/mnt/var/run:
total 0
```

4.1.2 Listato della directory di un Utility Disk usando ls-lR

```
total 15
drwx------   2 root     root         1024 Jun 18 19:57 bin/
drwxr-xr-x   2 root     root        12288 Jun 18 19:57 lost+found/
drwx------   2 root     root         1024 Jun 18 19:57 sbin/
drwxr-xr-x   4 root     root         1024 May  5 16:30 usr/

/mnt/bin:
total 13
-rwxr-xr-x   1 root     root         3180 Apr 10  1993 free*
-rwxr-xr-x   1 root     root        10687 Feb 10  1994 pwd*
-rwx--x--x   1 root     root         3672 Nov 17  1992 rdev*

/mnt/lost+found:
total 0

/mnt/sbin:
total 18
-rwxr-xr-x   1 root     root        16336 Jun 18 14:31 insmod*
-rwxr-xr-x   1 root     root           68 Jun 18 14:31 lsmod*
lrwxrwxrwx   1 root     root            6 Jun 18 19:57 rmmod -> insmod*

/mnt/usr:
total 2
drwx------   2 root     root         1024 Jun 18 19:57 bin/
drwxr-xr-x   3 root     root         1024 Jun 18 19:57 local/

/mnt/usr/bin:
total 411
-rwxr-xr-x   1 root     bin        111616 Sep  9  1992 awk*
-rwxr-xr-x   1 root     root        41984 Dec 23  1992 cpio*
-rwxr-xr-x   1 root     root        50176 Dec 23  1992 find*
-rwxr-xr-x   1 root     root       115712 Sep 17  1992 gawk*
-rwxr-xr-x   1 root     bin         37888 Sep  4  1992 grep*
-rwxr-xr-x   1 root     root        63874 May  1  1994 gzip*
-rwxr-xr-x   1 root     root         2044 Sep 17  1992 kill*
-rwx--x--x   1 root     root         3132 Jan 24  1993 mt*
-rwxr-xr-x   1 root     root         3416 Sep 22  1992 strings*
-rwxr-xr-x   1 root     other        3848 Sep  4  1992 who*

/mnt/usr/local:
```

```
total 1
drwx------   2 root     root         1024 Jun 18 19:57 bin/

/mnt/usr/local/bin:
total 374
-rwxr-xr-x   1 root     root       155542 Jun 18 17:07 ftape.o*
-rwxr-xr-x   1 root     root       226308 Jun 13  1994 tar*
```

4.2 Script di Shell per Creare un Dischetto

Questi script di shell sono forniti solo come esempi. Io non li uso più perché ora uso e raccomando il pacchetto `Bootkit` di Scott Burkett per creare i dischetti di recupero. Bootkit è basato su questi script, e fa essenzialmente le stesse cose, ma è molto più semplice da usare. Comunque, questi esempi di script funzionano correttamente nel creare i dischetti.

Ci sono due script disponibili:

- mkroot - crea un disco root o boot/root.
- mkutil - crea un disco utility.

Entrambi sono configurati in modo da funzionare nella directory radice dei dischi boot_disk e util_disk, ognuna delle quali contiene contiene tutto ciò che deve essere copiato sul relativo disco. Notate che questi script NON copiano automaticamente tutti i file per voi - voi decidete quali file copiare, create le directory e ci copiate questi file. Questi script di shell sono degli esempi che copiano il contenuto di queste directory.

Entrambi gli script contengono delle variabili di configurazione all'inizio che gli permettono di essere facilmente configurati per funzionare dovunque. Primo, create le directory modello e copiate tutti i file richiesti in esse. Poi controllate le variabili di configurazione degli script e cambiatele come richiesto, prima di eseguire gli script.

4.2.1 mkroot - Crea un disco di Root o di Root/Boot

```
# mkroot: make a boot/boot disk - creates a boot/root diskette
#       by building a file system on it, then mounting it and
#       copying required files from a model.
#       Note: the model to copy from from must dirst be set up,
#       then change the configuration variables below to suit
#       your system.
#
# usage: mkroot [nokernel]
#       if the parameter is omitted, then the kernel and LILO
#       are copied.

# Copyright (c) Graham Chapman 1995. All rights reserved.
# Permission is granted for this material to be freely
# used and distributed, provided the source is acknowledged.
# No warranty of any kind is provided. You use this material
# at your own risk.

# Configuration variables...
BOOTDISKDIR=./boot_disk       # name of boot disk directory
MOUNTPOINT=./mnt              # temporary mount point for diskette
LILODIR=/sbin                 # directory containing lilo
```

```
LILOBOOT=/boot/boot.b         # lilo boot sector
LILOMSG=./lilo.msg            # lilo message to display at boot time
LILOCONFIG=./lilo.conf        # lilo parms for boot/root diskette
DISKETTEDEV=/dev/fd0          # device name of diskette drive

echo $0: create boot/root diskette
echo Warning: data on diskette will be overwritten!
echo Insert diskette in $DISKETTEDEV and and press any key...
read anything

mke2fs $DISKETTEDEV
if [ $? -ne 0 ]
then
        echo mke2fs failed
        exit
fi

mount -t ext2 $DISKETTEDEV $MOUNTPOINT
if [ $? -ne 0 ]
then
        echo mount failed
        exit
fi

# copy the directories containing files
for i in bin etc lib
do
        cp -dpr $BOOTDISKDIR/$i $MOUNTPOINT
done

# copy dev *without* trying to copy the files in it
cp -dpR $BOOTDISKDIR/dev $MOUNTPOINT

# create empty directories required
mkdir $MOUNTPOINT/proc
mkdir $MOUNTPOINT/tmp
mkdir $MOUNTPOINT/mnt
mkdir $MOUNTPOINT/usr

# copy the kernel
if [ "$1" != "nokernel" ]
then
        echo "Copying kernel"
        cp $BOOTDISKDIR/zImage $MOUNTPOINT
        echo kernel copied

        # setup lilo
        cp $LILOBOOT $MOUNTPOINT
        cp $LILOMSG $MOUNTPOINT
        $LILODIR/lilo -C $LILOCONFIG
        echo LILO installed
fi

umount $MOUNTPOINT
```

```
echo Root diskette complete
```

4.2.2 mkutil - Crea un Disco di Utility

```
# mkutil: make a utility diskette - creates a utility diskette
#       by building a file system on it, then mounting it and
#       copying required files from a model.
#       Note: the model to copy from from must first be set up,
#       then change the configuration variables below to suit
#       your system.

# Copyright (c) Graham Chapman 1995. All rights reserved.
# Permission is granted for this material to be freely
# used and distributed, provided the source is acknowledged.
# No warranty of any kind is provided. You use this material
# at your own risk.

# Configuration variables...
UTILDISKDIR=./util_disk       # name of directory containing model
MOUNTPOINT=./mnt              # temporary mount point for diskette
DISKETTEDEV=/dev/fd0          # device name of diskette drive

echo $0: create utility diskette
echo Warning: data on diskette will be overwritten!
echo Insert diskette in $DISKETTEDEV and and press any key...
read anything

mke2fs $DISKETTEDEV
if [ $? -ne 0 ]
then
        echo mke2fs failed
        exit
fi

# Any file system type would do here
mount -t ext2 $DISKETTEDEV $MOUNTPOINT
if [ $? -ne 0 ]
then
        echo mount failed
        exit
fi

# copy the directories containing files
cp -dpr $UTILDISKDIR/bin $MOUNTPOINT

umount $MOUNTPOINT

echo Utility diskette complete
```

5 FAQ

5.1 D. Come creo un boot disk con il driver XXX?

Il modo più semplice è quello di prendere uno dei kernel precompilati distribuiti con la Slackware dal più vicino mirror site. I kernel della Slackware sono generici e includono il numero maggiore di driver per tutti i device possibili, e quindi se avete un controller SCSI o IDE, è probabile che il driver adatto sia incluso nel kernel della Slackware.

Andate nella directory `a1` e selezionate il kernel IDE o SCSI a seconda del controller che avete. Controllate il file `xxxxkern.cfg` per il kernel scelto e controllate quali driver sono stati inclusi nel kernel. Se il device che volete è in quella lista, allora il kernel corrispondente dovrebbe essere in grado di far partire il vostro computer. Scaricate il file `xxxxkern.tgz` e copiatelo nel vostro disco di boot che è descritto sopra nella sezione su come costruire un disco di boot.

Dovete poi controllare il root device nel kernel, usando il comando rdev:

```
rdev zImage
```

Rdev vi mostrerà il root device corrente nel kernel. Se questo non è lo stesso che voi volete, allora usate rdev per cambiarlo. Per esempio, il kernel che io ho provato era impostato su `/dev/sda2`, ma la mia partizione scsi root è `/dev/sda8`. Per usare un disco root, dovete usare questo comando:

```
rdev zImage /dev/fd0
```

Se volete sapere come costruire un disco root tipo Slackware, sappiate che questo non è un argomento coperto da questo HOW-TO perché fuori dallo scopo per cui è stato fatto questo documento, quindi vi suggerisco di leggere la Linux Install Guide o di prendervi una distribuzione slackware. Consulate la sezione intitolata “Referenze” per saperne di più.

5.2 D. Come aggiornare il mio disco di boot con un nuovo kernel?

Copiate il kernel sul vostro disco di boot usando il comando dd per un disco di boot senza file system, o il comando cp per un disco boot/root. Fate riferimento alla sezione intitolata “Boot” per maggiori dettagli.

5.3 D. Come tolgo LILO in modo da usare il DOS per eseguire il boot del DOS nuovamente?

Questa non è propriamente un argomento riguardante i dischi di boot, ma è fatta così spesso... La risposta è, usate il comando DOS:

```
FDISK /MBR
```

MBR sta per Master Boot Record, e rimpiazza il boot sector con uno di tipo DOS, senza toccare la tavola delle partizioni. Alcuni non sono d'accordo con questo metodo, ma anche l'autore del LILO, Werner Almesberger, lo suggerisce. È semplice e funziona.

Potete usare anche il comando dd per copiare il backup salvato dal LILO del boot sector vecchio sul boot sector nuovo - fate riferimento alla documentazione del LILO se intendete usare questo metodo.

5.4 D. Come posso eseguire il boot se ho perso il mio kernel _E_ il mio disco di boot?

Se non avete un boot disk, allora la cosa più semplice da fare è procurarsi un kernel adatto dalla Slackware come descritto prima nella sezione "Come creo un disco di boot con il driver XXX?". Potete quindi far partire il computer usando questo kernel, e poi riparare qualunque danno sia successo.

Il kernel che avete preso potrebbe non avere il root device settato con il tipo di disco e la partizione che voi volete. Per esempio, i kernel generici della Slackware per gli scsi hanno il root device settato a /dev/sda2, quando, invece, la mia partizione Linux si trova in /dev/sda8. In questo caso il root device del kernel deve essere cambiato.

Potete cambiare il root device e i settaggi del ramdisk nel kernel anche se tutto ciò che avete è un kernel, e qualche altro sistema operativo come il DOS.

Rdev cambia i parametri del kernel cambiando i valori in appositi punti del file immagine del kernel, in modo che potete fare lo stesso usando un semplice hex editor disponibile su qualunque sistema voi state usando - per esempio le Norton Utilities Disk Editor sotto DOS. Poi dovete controllare e se necessario, cambiare i valori nel kernel in questi punti:

```
0x01F8  Low byte della grandezza del RAMDISK
0x01F9  High byte della grandezza del RAMDISK
0x01FC  Root minor device number - vedi sotto
0X01FD  Root major device number - vedi sotto
```

La grandezza di un ramdisk è il numero di blocchi di un ramdisk da creare. Se volete eseguire il boot da un disco root allora settatelo al decimale 1440, o 0x05A0, oltre ad impostare l'offset 0x01F8 a 0xA0 e l'offset 0x01F9 a 0x05. Questo allocherà sufficiente memoria per un disco da 1.4Mb.

I numeri maggiori e minori dei device devono essere impostati sul device su cui voi volete che il root filesystem sia "mountato". Alcuni valori utili potrebbero essere:

```
device          maggiore minore
/dev/fd0            2      0   primo floppy drive
/dev/hda1           3      1   partizione 1 sul primo drive IDE
/dev/sda1           8      1   partizione 1 sul primo drive scsi
/dev/sda8           8      8   partizione 8 sul primo drive scsi
```

Una volta che avete settato questi valori, potete scrivere il file su un dischetto usando o il Norton Utilities Disk Editor, o un programma chiamato `rawrite.exe`. Questo programma è incluso in molte distribuzioni, inclusa la distribuzione SLS e la Slackware. È un programma DOS che scrive un file su un disco vergine ("raw"), partendo dal boot sector, invece di scriverlo sul file system. Se usate le Norton Utilities, allora dovete scrivere il file su un disco fisico partendo dall'inizio del disco.

5.5 D. Come posso fare più copie di un dischetto boot/root?

Non è mai desiderabile avere un solo set di disci di recupero; 2 o 3 copie dovrebbero essere mantenute nel caso che una non fosse leggibile.

La via più semplice per creare delle copie di qualunque dischetto, inclusi i dischi di boot/root, è di usare il comando dd per copiare il contenuto del dischetto originale su un file sul vostro hard disk, e poi usare lo stesso comando per copiare il file su un nuovo dischetto. Notate che non avete, e non dovreste avere, bisogno di eseguire il mount dei dischi, perché dd scrive in maniere cruda sul device.

Per copiare l'originale, usate il comando:

```
dd if=device of=nomefile
dove    device e‘ il nome del device del dischetto
e       nomefile e‘ il nome di un file su cui volete copiarlo
```

Per esempio, per copiare da `/dev/fd0` su un file temporaneo `/tmp/diskette.copy`, userei il comando:

```
dd if=/dev/fd0 of=/tmp/diskette.copy
```

Omettendo il parametro “count”, come abbiamo fatto noi ora, significa che il dischetto intero di 2880 blocchi sarà copiato.

Per copiare il file risultante su un nuovo dischetto, inserite il nuovo dischetto e usate il comando inverso:

```
dd if=nomefile of=device
```

Notate che il discorso precedente assume che voi abbiate un solo floppy drive. Se ne avete due dello stesso tipo, potete copiare il dischetto usando il comando:

```
dd if=/dev/fd0 of=/dev/fd1
```

5.6 D. Come posso eseguire il boot senza scrivere ogni volta “ahaxxxx=nn,nn,nn”?

Quando un device non può essere riconosciuto automaticamente dal kernel, è necessario passargli un stringa di comando, come:

```
aha152x=0x340,11,3,1
```

Questo parametro può essere passato in molti modi usando LILO:

- Scrivendolo sulla linea di comando ogni volta che si esegue il boot del sistema, via LILO. Ciò è comunque noioso.
- Usando l’opzione di LILO “lock”, in modo da memorizzare la linea di comando come la linea di comando di default; così facendo LILO userà la stessa linea di comando ogni volta.
- Usando l’opzione APPEND nel file di configurazione del LILO. Notate che la linea di comando deve essere racchiusa da virgolette.

Per esempio, una riga di comando d’esempio che usa il parametro lock, dovrebbe essere:

```
zImage  aha152x=0x340,11,3,1 root=/dev/sda1 lock
```

Questo dovrebbe passare la stringa per il device al kernel, oltre a settare il root device su `/dev/sda1` e salva l’intero comando per riutilizzarlo in futuro.

Un esempio dell’uso di APPEND è:

```
APPEND = "aha152x=0x340,11,3,1"
```

Notate che il parametro NON viene incluso nelle virgolette.

Notate inoltre che per la stringa che passate al kernel usando il parametro APPEND, bisogna che il kernel stesso abbia il supporto per quel comando. In caso contrario dovrete ricompilare il kernel con il supporto per il device richiesto. Per i dettagli su come ricompilare il kernel, cd su `/usr/src/linux` e leggete il file `README`, e leggete le LINUX FAQ e l'Installation HOWTO. Alternativamente potete ottenere un kernel generico che faccia al caso vostro.

I lettori sono fortemente invitati a leggere la documentazione del LILO prima della sua installazione. Un uso incauto del parametro "BOOT" può danneggiare le partizioni.

5.7 D. Come creo un filesystem su un disco RAM più capiente?

Un filesystem su un disco RAM è un filesystem residento su un disco in memoria che ha una capacità maggiore del disco di root dal quale è stato caricato. Questo può essere molto utile quando si usa Ftape, che richiede un controllo esclusivo del controller del floppy disk.

Due cose sono richieste: creare un filesystem più capiente sul disco di root e poi applicare una "patch" al kernel in modo che sia in grado di caricare blocchi oltre la fine del dischetto.

Due sono i metodi possibili:

- Usare il parametro "blocks" del programma e2fsck per specificare quanto occupa il filesystem che volete non disco RAM. Per esempio:

  ```
  mke2fs /dev/fd0 3000
  ```

  creerà un filesystem su dischetto di 3000 blocchi da 1Kb. Il dischetto ha solo 1440 blocchi, ma mke2fs funzionerà correttamente lo stesso. L'importante è non usare più di 1440 blocchi per i dati (permettendo l'uso di blocchi come "inodes" o riservati etc).

- Creare una partizione sul vostro hard-disk capiente quanto il filesystem che intendete creare sul disco RAM. Poi creare un filesystem su questa partizione e mettervi i file che desiderate. Poi usate il comando dd per copiare i primi 1440 blocchi sul dischetto, e poi controllare che non sia rimasto nessun file non copiato. Per esempio:

  ```
  dd if=/dev/hdb of=/dev/fd0 bs=1024 count=1440
  dd if=/dev/hdb of=tailpart bs=1024 skip=1440
  cmp -l tailparm /dev/zero
  ```

Scegliete una delle due; personalmente io preferisco la prima - sembra più facile e più sicura.

La seconda cosa richiesta per avere un filesystem di maggior capienza è di far fermare il kernel alla fine fisica del disco quando tenta di caricare il dischetto di root nel disco RAM. Per fare questo bisogna apportare una piccola modifica al kernel nel driver per il disco RAM, che dovrebbe essere situato in `/usr/src/linux/drivers/block/ramdisk.c`. Il seguente "patch" è stato fatto da Bruce Elliot. È da applicare al kernel 1.2.0, ma dovrebbe essere facile applicarlo anche ai kernel più recenti.

```
=================================================================
X--- ramdisk.c~ Mon Jan 23 13:04:09 1995
X+++ ramdisk.c  Mon May 29 00:54:52 1995
X@@ -113,6 +113,7 @@
X               (struct ext2_super_block *)&sb;
X       int             block, tries;
X       int             i = 1;
X+      int             fblocks;
```

```
X       int             nblocks;
X       char            *cp;
X
X@@ -168,12 +169,16 @@
X                                       nblocks, rd_length >> BLOCK_SIZE_BITS);
X                       return;
X               }
X-              printk("RAMDISK: Loading %d blocks into RAM disk", nblocks);
X+              fblocks = blk_size[MAJOR(ROOT_DEV)][MINOR(ROOT_DEV)];
X+              if (fblocks > nblocks)
X+                      fblocks = nblocks;
X+              printk(``RAMDISK: Loading %d blocks into %d block filesystem "
X+                                      "in RAM disk", fblocks, nblocks);
X
X               /* We found an image file system.  Load it into core! */
X               cp = rd_start;
X-              while (nblocks) {
X-                      if (nblocks > 2)
X+              while (fblocks) {
X+                      if (fblocks > 2)
X                               bh = breada(ROOT_DEV, block, BLOCK_SIZE, 0,  PAGE_SIZE);
X                       else
X                               bh = bread(ROOT_DEV, block, BLOCK_SIZE);
X@@ -184,7 +189,7 @@
X                       }
X                       (void) memcpy(cp, bh->b_data, BLOCK_SIZE);
X                       brelse(bh);
X-                      if (!(nblocks-- & 15)) printk(``.'');
X+                      if (!(fblocks-- & 15)) printk(``.'');
X                       cp += BLOCK_SIZE;
X                       block++;
X                       i++;
================================================================
```

Con questa modifica, il kernel si fermerà nel caricare alla fine fisica del disco, lasciando un filesystem più capiente del disco.

Alcuni avvertimenti: io sono stato in grado di creare in questo modo un filesystem su un disco RAM di 3500 blocchi, ma provando con 3600 o più il kernel collassava con un errore come “fixup table corrupt”. Non sono riuscito a evitare questa cosa, ma in ogni caso 3500 blocchi mi sembrano più che sufficienti.

5.8 D. All'atto del boot, rivevo un errore tipo `A: cannot execute B`. Perché?

Ci sono diversi casi di programmi che sono chiamati da altri programmi. Questi casi non avvengono sempre, ma possono spiegare perché un eseguibile apparentemente non può essere trovato su un sistema anche se potete vedere che è effettivamente presente. Potete vedere se un programma chiama un altro programma usando il comando “strings” e filtrando l'output attreverso il programma grep.

Esempi conosciuti sono:

- Shutdown in diversi versioni chiama `/etc/reboot`, quindi `reboot` deve essere posto nella directory `/etc`.
- Init ha causato problemi per almeno una persona, con il kernel che non riusciva a trovarlo.

Per risolvere questi problemi, spostate i programma nelle directory corrette, o cambiate i file di configurazione (per esempio `inittab`) per puntare alla directory corrette. Se siete in dubbio, mettete i programmi nelle stesse directory in cui sono sul vostro disco fisso, e usate lo stesso `inittab` e i file contenuti in `/etc/rc.d`, così come appaiono sul vostro disco fisso.

6 Riferimenti

In questa sezione, vvv è usato in un nome di un pacchetto al posto della versione, per evitare di riferirsi ad una versione specifica. Quando prendete un pacchetto, prendete sempre l'ultima versione a meno che non abbiate delle buoni ragioni per non farlo.

6.1 LILO - Linux Loader

Scritto da Werner Almesberger. eccellente boot loader, come del resto la documentazione che include informazioni sul contenuto del boot sector e i primi stadi del processo di boot.

Ftp da: `tsx-11.mit.edu:/pub/linux/packages/lilo/lilo.vvv.tar.gz` anche sul sunsite e mirror sites.

6.2 Linux FAQ e HOWTO

Questi sono disponibili un po' dappertutto. Date una occhiata ai newsgroup di usenet `news.answers` e `comp.os.linux.announce`.

Ftp da: `sunsite.unc.edu:/pub/Linux/docs`

- Le FAQ sono in `/pub/linux/docs/faqs/linux-faq`
- Gli HOWTO sono in `/pub/Linux/docs/HOWTO`

Gli HOWTO tradotti in italiano si possono reperire via ftp da: `ftp.unipd.it` Per il WWW, partite dalla Linux documentation home page:

```
http://sunsite.unc.edu/mdw/linux.html
```

Se siete proprio disperati, mandate una mail a:

```
mail-server@rtfm.mit.edu
```

con la parola "help" nel messaggio, poi seguite le istruzioni che vi verranno spedite.

Attenzione: se non avete letto le Linux FAQ e i documenti relativi come il Linux Installation HOWTO e il Linux Install Guide, allora non dovreste provare a creare un disco di boot.

6.3 Pacchetti per il recupero

6.3.1 Bootkit

Scritto da Scott Burkett. Bootkit fornisce un metodo, tramite menu a tendine, per gestire la creazione di dischi di recupero. Questo usa il programma Dialog per creare dei menu, e un albero delle directory per definire quello che sarà il contenuto dei dischi di recupero. Il pacchetto contiene anche un esempio dei file principali necessari. Il pacchetto fornisce solamente una struttura di supporto; è dovere dell'utente decidere quale sarà il contenuto dei dischi e, in accordo, sistemare i file di configurazione. Per gli utenti che non hanno problemi a fare ciò, è un ottima scelta. Io stesso uso questo pacchetto.

Ftp da: `sunsite.unc.edu:/pub/Linux/system/Recovery/Bootkit-vvv.tar.gz`

6.3.2 CatRescue

Scritto da Oleg Kibirev. Questo pacchetto mira a salvare spazio sui dischi di recupero usando intensivamente la compressione, e implementando gli eseguibili come script di shell. La documentazione inclusa, comprende alcuni suggerimenti su cosa fare in vari situazioni disastrose :-).

Ftp da: `gd.cs.csufresno.edu:/pub/sun4bin/src/CatRescue100.tgz`

6.3.3 Script di Shell per il Recupero

Scritti da Thomas Heiling. Sono script che producono disco root/boot. Questi però dipendono dalle versioni di qualche altro software come LILO, e quindi potreste avere qualche problema per adattarli al vostro sistema, ma potrebbero essere utili come punto di partenza se vi interessano degli script molto più completi di quelli presentati in questo documento.

Ftp da: `sunsite.unc.edu:/pub/Linux/system/Recovery/rescue.tgz`

6.3.4 SAR - Search and Rescue

Scritto da Karel Kubat. SAR produce un disco di recupero, usando diversi metodi per ridurre lo spazio occupato su disco. Il manuale include una descrizione dei processi di boot e di login del Linux.

Ftp da: `ftp.icce.rug.nl:/pub/unix/SAR-vvv.tar.gz`

Il manuale è disponibile via WWW in:

`http://www.icce.rug.nl/karel/programs/SAR.html`

6.4 Distribuzione Slackware

A parte il fatto che è una delle più popolari distribuzioni Linux che si trovano in giro, è anche un buon posto dove trovare un kernel generico. È disponibile praticamente ovunque, quindi non servirebbe scrivere qui gli indirizzi.