kerneld mini-HOWTO

Henrik Storner, <*storner@osiris.ping.dk*> v1.7, 19 luglio 1997

Questo documento spiega come puoi usare la funzione kerneld nei kernel di Linux. L'ultima versione rilasciata di questo documento la puoi trovare all'indirizzo <http://eolicom.olicom.dk/~storner/kerneld-mini-HOWTO.html>. Dopo ogni pubblicazione di questo mini-HOWTO puoi trovare degli aggiornamenti nella mia lista disordinata dei cambiamenti all'indirizzo <http://eolicom.olicom.dk/~storner/kern.html>. Traduzione di Lorenzo Cappelletti, <*L.Cappelletti@POBoxes.com*>.

Indice

1 Ringraziamenti **2**

2 Preliminari **3**

2.1 Cos'è kerneld? . . . 3

2.2 Perché avrei bisogno di usarlo? . . . 3

2.3 Dove posso prelevare i componenti necessari? . . . 4

3 Cominciamo a fare sul serio **4**

3.1 Come impostare il tutto? . . . 4

3.2 Proviamo kerneld . . . 5

4 Come fa kerneld a sapere quale modulo caricare? **6**

4.1 Periferiche a blocchi . . . 7

4.2 Periferiche a caratteri . . . 7

4.3 Periferiche di rete . . . 7

4.4 *Formati degli eseguibili* . . . 8

4.5 Disciplinea di linea (slip, cslip e ppp) . . . 8

4.6 Famiglie di protocolli di rete (IPX, AppleTalk, AX.25) . . . 8

4.7 File system . . . 9

5 Periferiche che richiedono una configurazione speciale **9**

5.1 *char-major-10*: mouse, *watchdog* e *random* . . . 9

5.2 Caricamento di driver SCSI: la voce *scsi_hostadapter* . . . 9

5.3 Quando caricare un modulo non è abbastanza: la voce 'post-install' . . . 10

6 Spiare kerneld **11**

7 Usi particolari di kerneld **11**

8 Problemi comuni e cose di cui non capisci il motivo **12**

8.1 Perché ottengo il messaggio "Cannot locate module for net-pf-*x*" (Non riesco a trovare il modulo per net-pf-*x*) quando lancio ifconfig? . . . 12

8.2 Dopo la partenza di `kerneld` il mio sistema viene messo in ginocchio quando attivo una connessione ppp . . . 12

8.3 `kerneld` non carica il mio driver SCSI! . . . 12

8.4 modprobe si lamenta che «gcc2_compiled» non è definito . . . 12

8.5 Il driver della mia scheda sonora non si ricorda delle impostazioni per il volume, etc. . . . 12

8.6 Dosemu ha bisogno di alcuni moduli. Come posso fare affinché `kerneld` li carichi? . . . 13

8.7 Perché ottengo un messaggio "Ouch, kerneld timed out, message failed" (Oops, tempo scaduto per kerneld, messaggio fallito)? . . . 13

8.8 mount non aspetta che `kerneld` carichi il modulo per il *filesystem* . . . 13

8.9 `kerneld` fallisce nel caricare il modulo ncpfs . . . 13

8.10 `kerneld` fallisce nel caricare il modulo per smbfs . . . 13

8.11 Ho compilato tutto come modulo e ora il mio sistema non si avvia piú . . . 14

8.12 `kerneld` fallisce nel caricare il *filesystem* principale . . . 14

8.13 `kerneld` non viene caricato all'avvio - si lamenta di libgdbm . . . 14

8.14 Ottengo un "Cannot load module *xxx*" (Non posso caricare il modulo *xxx*), ma ho appena riconfigurato il kernel senza il supporto per *xxx*! . . . 14

8.15 Ho ricompilato il kernel e i moduli, e continuo ad ottenere messaggi di simboli non risolti all'avvio . . . 14

8.16 Ho installato Linux 2.1 e ora non riesco piú a caricare alcun modulo . . . 15

8.17 E a proposito del collegamento alla rete su richiesta? . . . 15

9 Messaggio di copyright **15**

1 Ringraziamenti

Se dovessi trovare delle cose sbagliate in questo documento, ti pregherei di farmelo sapere.

Le seguenti persone hanno contribuito per questo mini-HOWTO in alcuni punti:

- Bjorn Ekwall <bj0rn@blox.se>
- Ben Galliart <bgallia@luc.edu>
- Cedric Tefft <cedric@earthling.net>
- Brian Miller <bmiller@netspace.net.au>
- James C. Tsiao <jtsiao@madoka.jpl.nasa.gov>

Ho apprezzato molto gli incoraggiamenti ed i suggerimenti che mi hanno spedito i lettori di questo mini-HOWTO.

2 Preliminari

2.1 Cos'è `kerneld`?

`kerneld` è una caratteristica introdotta durante lo sviluppo dei kernel 1.3 da *Bjorn Ekwall* <`mailto:bj0rn@blox.se`>. Viene acclusa con tutti i kernel delle versioni 2.0 e 2.1. Permette ai «moduli» del kernel (cioé driver di periferica, driver di rete, *filesystem*) di venir caricati automaticamente quando c'è bisogno, invece di doverlo fare manualmente con `modprobe` o `insmod`.

E per motivi piú divertenti, nonostante questi non siano (ancora?) integrati con il kernel standard:

- può essere impostato per lanciare un programma, invece del solito schermo nero, permettendoti cosí di usare ogni programma come uno *screen-saver*;
- similmente al supporto dell'oscuramento dello schermo, puoi anche cambiare il «beep» di console in qualche cosa di completamente differente...

`kerneld` consiste di due entità separate:

- un supporto nel kernel di Linux che permette di inviare delle richieste ad un demone, affinché sappia che un modulo è necessario per una certa operazione;
- un demone a livello utente che capisca quali moduli devono essere caricati per soddisfare alle richieste del kernel.

Entrambe le parti dovranno lavorare per far funzionare il supporto `kerneld`. Non è sufficiente che ne sia configurata una sola.

2.2 Perché avrei bisogno di usarlo?

Ci sono alcune buone ragioni per usare `kerneld`. Quelle che menzionerò sono le mie, gli altri potrebbero volerlo usare per altri motivi.

- Se devi compilare kernel per svariati sistemi che si distinguono di poco (differenti tipi di schede di rete, per esempio), puoi preparare un singolo kernel e alcuni moduli invece di essere costretto a compilare un kernel per ogni sistema.
- I moduli sono piú facili da provare per gli sviluppatori (non c'è bisogno di ravviare il sistema per caricare e scaricare i driver). Questo vale per tutti i moduli, non solo per quelli caricati da `kerneld`.
- Limita l'utilizzo di memoria da parte del kernel, cioé hai piú memoria disponibile per le applicazioni. La memoria usata dal kernel non è **mai** trasferita alla memoria di *swap*, cosí se ci sono 100kB di driver inutilizzati compilati nel kernel, questi sono semplicemente RAM sprecata.
- Alcune delle cose che uso (il driver per l'unità a nastro, per esempio, o l'iBCS) sono solo disponibili come moduli. Ma non voglio scocciarmi con il loro caricamento e scaricamento ogni volta che ne ho bisogno.
- Le persone che curano le distribuzioni Linux non devono compilare 284 differenti immagini di boot: ogni utente carica i driver di cui ha bisogno per il suo hardware. Questo è usato, per esempio, dalla RedHat 4.0 nella sua installazione.

Certo, ci sono anche ragioni per le quali potresti non volerlo usare (potresti preferire un unico file d'immagine per il kernel con tutti i tuoi driver compilati staticamente). In questo caso stai leggendo il documento sbagliato.

2.3 Dove posso prelevare i componenti necessari?

Il supporto nel kernel di Linux fu introdotto con Linux 1.3.57. Se hai una versione di kernel precedente, avrai bisogno di aggiornarla se vuoi il supporto per `kerneld`. Tutti i maggiori siti ftp di Linux contengono i sorgenti del kernel. Ti raccomando di aggiornarti all'ultima versione stabile, 2.0, ora a livello di *patch* 29:

- `<ftp://sunsite.unc.edu/pub/Linux/kernel/v2.0/>`linux-2.0.29.tar.gz
- `<ftp://tsx-11.mit.edu/pub/linux/sources/system/v2.0/>`linux-2.0.29.tar.gz
- `<ftp://ftp.funet.fi/pub/OS/Linux/PEOPLE/Linus/v2.0/>`linux-2.0.29.tar.gz

Il demone a livello utente è incluso nel pacchetto modules-1.2.8 e con il piú aggiornato modules-2.0. Questi sono normalmente disponibili dallo stesso sito in cui avete trovato i sorgenti del kernel, mentre i siti ufficiali includono:

- `<ftp://sunsite.unc.edu/pub/Linux/kernel/>`/modules-2.0.0.tar.gz
- `<ftp://tsx-11.mit.edu/pub/linux/sources/sbin/>`modules-2.0.0.tar.gz
- `<ftp://ftp.funet.fi/pub/OS/Linux/tools/>`modules-2.0.0.tar.gz

NOTA: se vuoi provare a caricare i moduli con gli ultimi kernel 2.1 in fase di **sviluppo**, devi procurarti il pacchetto modutils- (NON modules-). Comunque 8.16 (vedi piú avanti) per i problemi che si possono avere con i moduli ed il kernel 2.1.

3 Cominciamo a fare sul serio

3.1 Come impostare il tutto?

Primo procurarsi i componenti necessari: un kernel adatto e le ultime utility per i moduli. Poi devi installarle. Molto semplice: decomprimi i sorgenti e lancia `make install`. Questo compila e installa in `/sbin` i seguenti programmi: genksysm, insmod, lsmod, modprobe, depmod, kerneld. Ti raccomando di aggiungere le seguenti linee ai tuoi script di avvio che effettuano alcune impostazioni necessarie ogni volta che Linux viene avviato. Aggiungi le seguenti linee al tuo file `/etc/rc.d/rc.S` (se utilizzi Slackware) o a `/etc/rc.d/rc.sysinit` (se utilizzi SysVinit, cioé Debian, RedHat, Caldera):

```
# Lancia kerneld - questo deve accadere molto presto nel processo
# di boot, sicuramente PRIMA che venga avviato fsck sui filesystem,
# in quanto potrebbe richiedere l'autocaricamento di qualche driver
if [ -x /sbin/<tt/kerneld/ ]
then
        /sbin/<tt/kerneld/
fi

# I comandi fsck standard vanno qui
# assieme al comando mount per montare il fs in lettura-scrittura


# Aggiornamento del file per le dipendenze kernel-moduli
# Il fs root ora DEVE essere montato in lettura-scrittura
if [ -x /sbin/depmod ]
then
```

```
        /sbin/depmod -a
fi
```

La prima parte lancia `kerneld`.

La seconda parte chiama `depmod -a` all'avvio. Il programma depmod costruisce una lista di moduli disponibili e analizza le loro inter-dipendenze, cosí si sa se, per un modulo che sta per essere caricato, è necessario caricarne un altro prima.

Nota: nelle recenti versioni di `kerneld` c'è la possibilità di fare il link verso le librerie GNU dbm, libgdbm. Se abiliti questa opzione quando compili le utility per i moduli, *`kerneld` non partirà se libgdbm non è disponibile*, caso che potrebbe accadere se hai `/usr` su una partizione separata e fai partire `kerneld` prima che `/usr` sia montata. La soluzione consigliata è di spostare libgdbm da `/usr/lib` a `/lib` o linkarla staticamente a `kerneld`.

Successivamente decomprimi i sorgenti del kernel, configuralo e compilane uno che ti soddisfi. Se non lo hai mai fatto prima, devi definitivamente leggerti il file README che si trova al primo livello dei sorgenti di Linux. Quando lanci `make config` per configurare il kernel, devi prestare attenzione ad alcune domande che appaiono verso l'inizio:

```
Enable loadable module support (CONFIG_MODULES) [Y/n/?] Y
```

Hai bisogno di selezionare il supporto per i moduli caricabili o non ci saranno moduli per `kerneld` da caricare! Rispondi con *Yes*.

```
Kernel daemon support (CONFIG_KERNELD) [Y/n/?] Y
```

Anche questo, ovviamente, è necessario. A questo punto molte delle cose nel kernel possono essere compilate come moduli. Vedrai domande come:

```
Normal floppy disk support (CONFIG_BLK_DEV_FD) [M/n/y/?]
```

dove puoi rispondere con una «M» che sta per «Modulo». Generalmente solo i driver necessari per far partire il sistema (il driver per l'hard-disk, il driver per il *filesystem* principale) dovrebbero essere compilati nel kernel; gli altri possono essere compilati come moduli.

Una volta finito con `make config`, lancia `make dep`, `make clean`, `make zImage` o `make zlilo`, `make modules` e `make modules_install`.

Uff!

Il comando `make zImage` mette la nuova immagine del kernel nel file `/arch/i386/boot/zImage`. Avrai bisogno di copiarla dove tieni la tua immagine di boot o di configurare LILO in seguito.

Per maggiori informazioni su come configurare, compilare e installare il tuo kernel, dai un'occhiata al *Kernel-HOWTO* `<http://sunsite.unc.edu/mdw/HOWTO/Kernel-HOWTO.html>`, postato regolarmente in comp.os.linux.answers e disponibile su sunsite.unc.edu in /pub/Linux/docs/HOWTO.

3.2 Proviamo `kerneld`

Ora fai il reboot con il nuovo kernel. Quando il sistema è di nuovo pronto, puoi lanciare un `ps -ax` con il quale dovresti vedere una linea dedicata a `kerneld`:

```
PID TTY STAT  TIME COMMAND
 59  ?  S     0:01 /sbin/kerneld
```

Una delle cose carine con `kerneld` è che, una volta che hai installato il kernel e il demone, poche impostazioni sono ancora necessarie. Per un inizio prova ad usare uno dei driver che hai compilato come modulo (in generale tutto funziona senza ulteriori configurazioni). Io ho compilato il driver per il floppy come modulo, cosí posso mettere un dischetto DOS nel drive e

```
osiris:~ $ mdir a:
 Volume in drive A has no label
 Volume Serial Number is 2E2B-1102
 Directory for A:/

binuti~1 gz       1942 02-14-1996  11:35a binutils-2.6.0.6-2.6.0.7.diff.gz
libc-5~1 gz      24747 02-14-1996  11:35a libc-5.3.4-5.3.5.diff.gz
        2 file(s)        26689 bytes
```

Cosí il driver del floppy funziona: viene caricato automaticamente da `kerneld` quando provo ad usare il disco floppy.

Per vedere, invece, che il modulo del floppy viene effettivamente caricato, puoi lanciare `/sbin/lsmod` che lista tutti i moduli attualmente caricati:

```
osiris:~ $ /sbin/lsmod
Module:         #pages:  Used by:
floppy             11    0 (autoclean)
```

L'«autoclean» sta ad indicare che il modulo verrà automaticamente rimosso da `kerneld` dopo che questo non viene usato per piú di un minuto. Cosí le 11 pagine di memoria (=44kB, una pagina è 4kB) verranno utilizzate solo quando accedo al drive del floppy (se non uso il floppy per piú di un minuto, verranno liberate). Alquanto carino, se sei a corto di memoria per le tue applicazioni!

4 Come fa `kerneld` a sapere quale modulo caricare?

Nonostante `kerneld` ci venga fornito con informazioni già pronte sui tipi piú comuni di moduli, ci sono situazioni in cui `kerneld` non saprà come trattare una richiesta del kernel. Questo accade per moduli tipo il driver per il CD-ROM o i driver di rete, dove ci sono piú di un modulo da poter caricare.

Le richieste che il demone `kerneld` riceve dal kernel possono appartenere ad uno dei seguenti tipi:

- 4.1 (driver di periferica a blocchi)
- 4.2 (driver di periferica a caratteri)
- 4.4 (formato binario)
- 4.5 (disciplinea di linea tty)
- *4.7 (filesystem)*
- 4.3 (periferica di rete)
- servizio di rete (per esempio rarp)
- 4.6 (protocollo di rete) (per esempio IPX)

`kerneld` determina quale modulo dovrebbe essere caricato cercando nel file di configurazione `/etc/conf.modules`. Ci sono due tipi di voci possibili in questo file: *path* (dove si trovano i file dei moduli) e *alias* (quale modulo dovrebbe essere caricato). Se non hai già questo file, dovresti crearlo con il comando:

```
/sbin/modprobe -c | grep -v '^path' > /etc/conf.modules
```

Se vuoi aggiungere un'altra direttiva «path» ai percorsi di default, **devi anche includere tutti gli altri percorsi di «default»**, in quanto una direttiva «path» in `/etc/conf.modules` **rimpiazzerà** tutti quelli che modprobe conosce per default!

Normalmente non avrai bisogno di aggiungere alcun percorso, in quanto l'insieme di quelli precompilati dovrebbe essere sufficiente per tutte le configurazioni «normali» (ed altre ancora...). Promesso!

Diversamente, se vuoi aggiungere un «alias» o una direttiva «option», le nuove voci in `/etc/conf.modules` verranno **aggiunte** a quelle che modprobe già conosce. Se devi **ridefinire** un «alias» o «options», le nuove voci in `/etc/conf.modules` sovrascriveranno quelle precompilate.

4.1 Periferiche a blocchi

Se esegui `/sbin/modprobe -c`, otterrai una lista di moduli che `kerneld` conosce, e di richieste alle quali questi corrispondono. Per esempio, la richiesta che termina con il caricamento del driver del floppy è (per una periferica a blocchi che ha un *major number* pari a 2):

```
osiris:~ $ /sbin/modprobe -c | grep floppy
alias block-major-2 floppy
```

Perché block-major-2? Perché le periferiche floppy `/dev/fd*` usano una periferica con *major* di 2 e sono periferiche a blocchi:

```
osiris:~ $ ls -l /dev/fd0 /dev/fd1
brw-rw-rw-   1 root     root       2,   0 Mar  3  1995 /dev/fd0
brw-r--r--   1 root     root       2,   1 Mar  3  1995 /dev/fd1
```

4.2 Periferiche a caratteri

Le periferiche a caratteri sono trattate in modo analogo. Per esempio il driver per l'unità a nastro connessa come floppy ha un *major* di 27:

```
osiris:~ $ ls -lL /dev/ftape
crw-rw----   1 root     disk      27,   0 Jul 18  1994 /dev/ftape
```

Però `kerneld` non conosce per default il driver dell'unità a nastro. Infatti non è presente nella lista ottenuta da `/sbin/modprobe -c`.

Cosí per impostare `kerneld` affinché carichi il driver per l'unità a nastro, si deve aggiungere una linea al file di configurazione di `kerneld` `/etc/conf.modules`:

```
alias char-major-27 ftape
```

4.3 Periferiche di rete

Puoi anche usare il nome della periferica al posto di impostazioni come «char-major-*xxx*» o «block-major-*yyy*». Questo è particolarmente utile per i driver di rete. Per esempio un driver per una scheda di rete tipo ne2000 abilitata come eth0 verrebbe caricato con

```
alias eth0 ne
```

Se hai la necessità di passare alcune opzioni al driver (per esempio per informare il modulo su quale IRQ la scheda di rete sta usando) aggiungi una linea «options»:

```
options ne irq=5
```

Questo farà in modo che `kerneld` carichi il driver NE2000 con il comando:

```
/sbin/modprobe ne irq=5
```

Ovviamente le opzioni effettivamente disponibili sono specifiche al modulo che caricherai.

4.4 *Formati degli eseguibili*

I formati binari sono trattati in modo simile. Ogni volta che provi a lanciare un programma che `kerneld` non sa come caricare, `kerneld` riceve una richiesta per «binfmt-*xxx*», dove *xxx* è il numero determinato dai primi byte dell'eseguibile. Cosí la configurazione di `kerneld` per supportare il modulo `binfmt_aout` per gli eseguibili ZMAGIC (a.out) è:

```
alias binfmt-267 binfmt_aout
```

in quanto il *magic number* (vedi `/etc/magic`) per file ZMAGIC è 267. (Se provi a controllare `/etc/magic`, vedrai il numero 0413, ma `/etc/magic` usa numeri ottali, mentre `kerneld` li usa decimali, e ottale 413 = decimale 267). In realtà ci sono tre varianti leggermente diverse per gli eseguibili a.out (NMAGIC, QMAGIC and ZMAGIC), cosí per un pieno supporto del modulo binfmt_aout abbiamo bisogno di:

```
alias binfmt-264 binfmt_aout  # eseguibile puro (NMAGIC)
alias binfmt-267 binfmt_aout  # eseguibile demand-paged (ZMAGIC)
alias binfmt-204 binfmt_aout  # eseguibile demand-paged (QMAGIC)
```

I formati binari a.out, Java e iBCS sono riconosciuti automaticamente da `kerneld`, senza alcuna configurazione.

4.5 Disciplinea di linea (slip, cslip e ppp)

Le disciplinee di linea sono richieste con «tty-ldisc-*x*», dove *x* assume solitamente i valori 1 (per SLIP) o 3 (per PPP). Entrambi sono riconosciuti da `kerneld` automaticamente.

A proposito di ppp, se vuoi che `kerneld` carichi il modulo bsd_comp per la compressione dei dati per il ppp, allora devi aggiungere le due linee seguenti al tuo `/etc/conf.modules`:

```
alias tty-ldisc-3 bsd_comp
alias ppp0 bsd_comp
```

4.6 Famiglie di protocolli di rete (IPX, AppleTalk, AX.25)

Anche alcuni protocolli di rete possono essere caricati come moduli. Il kernel domanda a `kerneld` di una famiglia di protocolli (per esempio IPX) con una richiesta del tipo «net-pf-*x*», dove *x* è un numero che sta ad indicare la famiglia voluta. Per esempio net-pf-3 è AX.25, net-pf-4 è IPX e net-pf-5 è AppleTalk (questi numeri sono determinati dalle definizioni AF_AX25, AF_IPX, etc. nel file sorgente di Linux `include/linux/socket.h`).

```
alias net-pf-4 ipx
```

Dai un'occhiata anche alla sezione riguardante i 8 (problemi comuni) per informazioni su come evitare alcuni noiosi messaggi all'avvio relativi a famiglie di protocolli indefiniti.

4.7 File system

Le richieste di `kerneld` per i *filesystem* sono semplicemente i nomi dei *filesystem*. Un tipico uso potrebbe essere quello di caricare il modulo isofs per i *filesystem* del CD-ROM, cioé per *filesystem* di tipo «iso9660»:

```
alias iso9660 isofs
```

5 Periferiche che richiedono una configurazione speciale

Alcune periferiche richiedono una configurazione che va leggermente al di là del semplice uso di «alias» del tipo *periferica-modulo*.

- periferiche a caratteri con *major number* 10: 5.1 (Periferiche varie)
- 5.2 (periferiche SCSI)
- 5.3 (periferiche che richiedono inizializzazioni speciali)

5.1 *char-major-10*: mouse, *watchdog* e *random*

Le periferiche hardware sono usualmente identificate con il loro *major number*, cosí per l'unità a nastro si ha un *char-major-27*. Ciò nonostante, se scorri le voci presenti in `/dev` che contengono il *char-major-10*, vedrai che queste formano un bel gruppo di periferiche molto diverse fra loro:

- mouse di vari tipi (*bus mouse*, mouse PS/2)
- periferiche *watchdog*
- la periferica del kernel «*random*»
- interfaccia APM (Advanced Power Management)

Ovviamente queste periferiche sono controllate da altrettanti moduli differenti, non da uno solo. Perciò la configurazione di `kerneld` per queste **periferiche eterogenee** fa uso del *major number* **e** del *minor number*:

```
alias char-major-10-1 psaux     # per mouse PS/2
alias char-major-10-130 wdt     # per il watchdog WDT
```

Hai bisogno di una versione del kernel 1.3.82 o superiore per poter utilizzare questa caratteristica; le versioni precedenti non passano il *minor number* a `kerneld`, impedendogli di capire quale modulo di periferica caricare.

5.2 Caricamento di driver SCSI: la voce *scsi_hostadapter*

I driver per le periferiche SCSI consistono in un driver per l'adattatore SCSI (per esempio un Adaptec 1542) e di un driver per il tipo di periferica SCSI che usi (per esempio un hard-disk, un CD-ROM o un'unità a nastro). Tutti questi possono essere caricati come moduli. Perciò, quando vuoi accedere, per esempio, al lettore di CD-ROM connesso alla scheda Adaptec, il kernel e `kerneld` sanno solo che c'è bisogno di caricare

il modulo *sr_mod* per supportare il CD-ROM SCSI, ma non sanno a quale controller il CD-ROM è connesso né, tanto meno, quale modulo caricare per supportare il controller SCSI.

Per risolvere questo problema puoi aggiungere una voce per il driver SCSI al tuo `/etc/conf.modules` che dica a `kerneld` quale dei possibili moduli per controller SCSI deve caricare:

```
alias scd0 sr_mod               # sr_mod per CD-ROM SCSI...
alias scsi_hostadapter aha1542  # ...necessitano del driver Adaptec
```

Questo funziona solo con versioni del kernel 1.3.82 o superiori.

Il metodo va bene se hai un solo controller SCSI. Se ne hai piú d'uno, le cose diventano un po' piú difficili.

In generale non puoi lasciare che `kerneld` carichi un driver per un adattatore SCSI se un driver per un altro adattatore SCSI è già installato. Sei costretto a compilare entrambi i driver direttamente nel kernel (non come moduli) o caricare i moduli manualmente.

Anche se un modo per caricare piú driver SCSI *c'è*. James Tsiao ha avuto questa idea:

> Puoi facilmente fare in modo che `kerneld` carichi il secondo driver scsi impostando le dipendenze in `modules.dep` a mano. Hai bisogno solo di una voce come:
>
> `/lib/modules/2.0.30/scsi/st.o: /lib/modules/2.0.30/scsi/aha1542.o`
>
> per far caricare a `kerneld` aha1542.o prima che carichi st.o. La mia macchina a casa è configurata praticamente come sopra e tutto funziona bene per le mie periferiche scsi secondarie, inclusa l'unità a nastro, il CD-ROM e periferiche scsi generiche. La controparte sta nel fatto che `depmod -a` non può rilevare automaticamente queste dipendenze, cosí l'utente ha bisogno di aggiungerle manualmente e non può lanciare `depmod -a` all'avvio. Ma, una volta che tutto è configurato, `kerneld` caricherà in automatico l'aha1542.o senza problemi.

Dovresti essere conscio che questa tecnica funziona solo se hai tipi diversi di periferiche SCSI collegate ai due controller (per esempio degli hard-disk su un controller e lettori CD-ROM, unità a nastro o periferiche SCSI generiche sull'altro).

5.3 Quando caricare un modulo non è abbastanza: la voce 'post-install'

Qualche volta caricare solo il modulo non è abbastanza per far funzionare le cose. Per esempio, se hai compilato la scheda sonora come modulo, spesso è conveniente impostare un certo livello per il volume. L'unico problema è che l'impostazione svanisce quando viene caricato il modulo un'altra volta. Ecco, in breve, un trucco di *Ben Galliart* <mailto:bgallia@luc.edu>:

> La soluzione definitiva richiesta dall'installazione del pacchetto *setmix-0.1* <ftp://sunsite.unc.edu/pub/Linux/apps/sound/mixers/setmix-0.1.tar.gz>. E poi aggiungendo le seguenti linee al mio `/etc/conf.modules`:
>
> `post-install sound /usr/local/bin/setmix -f /etc/volume.conf`

Questa linea fa in modo che `kerneld`, dopo che il modulo per l'audio è stato caricato, lanci il comando indicato dalla voce «post-install sound». Cosí il modulo sonoro viene configurato con il comando `/usr/local/bin/setmix -f /etc/volume.conf`.

Ciò può essere utile anche per altri moduli, per esempio il modulo lp può essere configurato con il programma tunelp aggiungendo:

```
post-install lp tunelp <options>
```

Affinché kerneld recepisca queste opzioni, hai bisogno di una versione di kerneld che sia la 1.3.69f o superiore.

Nota: una versione recente di questo mini-HOWTO parlava di un'opzione «pre-remove», che si sarebbe potuta usare per eseguire un comando appena prima che kerneld rimuovesse un modulo. Ciò nonostante questa opzione non ha mai funzionato e il suo uso è perciò scoraggiato (piú appropriatamente, questa opzione scomparirà in una *release* futura di kerneld). L'intera struttura delle «impostazioni» per un modulo sta subendo alcuni cambiamenti in questo momento e potrebbe essere diversa sul tuo sistema quando leggerai questo documento.

6 Spiare kerneld

Se hai provato qualsiasi cosa e non sei proprio capace di immaginare cosa il kernel stia chiedendo di fare a kerneld, c'è un modo di vedere le richieste che kerneld riceve e, da questo, capire cosa deve andare in /etc/conf.modules: l'utility kdstat.

Questo piccolo e grazioso programma fa parte del pacchetto dei moduli, ma non viene compilato né installato per default. Per ottenerlo:

```
cd /usr/src/modules-2.0.0/kerneld
make kdstat
```

Poi, per fare in modo che kerneld mostri informazioni su cosa sta facendo, lancia

```
kdstat debug
```

e kerneld comincerà a vomitare messaggi sulla *console* dicendoti di cosa sta facendo. Se poi provi ad eseguire il comando che vuoi utilizzare, vedrai comparire le richieste di kerneld; queste possono essere messe in /etc/conf.modules utilizzando un sinonimo per il modulo necessario a completare il lavoro.

Per fermare la fase di *debug*, lancia /sbin/kdstat nodebug.

7 Usi particolari di kerneld

Lo sapevo che volevi chiedere come impostare il modulo per lo screen-saver...

La directory kerneld/GOODIES nel pacchetto dei moduli ha un paio di *patch* per il supporto di kerneld di screen-saver e beep della *console*; queste non fanno ancora parte del kernel ufficiale. Cosí avrai bisogno di installarle e ricompilare il kernel.

Per installare una *patch* si usa il comando patch:

```
cd /usr/src/linux
patch -s -p1 < /usr/src/modules-2.0.0/<tt/kerneld//GOODIES/blanker_patch
```

Poi ricompila ed installa il nuovo kernel.

Quando lo screen-saver viene attivato kerneld eseguirà il comando /sbin/screen-blanker (questo può essere uno *script* di *shell* che lancia il tuo screen-saver favorito).

Quando il kernel vuole ripristinare lo schermo invia un segnale SIGQUIT al processo che sta eseguendo /sbin/screenblanker. Lo *script* di *shell* deve poterlo catturare e terminare. Ricordati di ripristinare lo schermo alla modalità testo originale!

8 Problemi comuni e cose di cui non capisci il motivo

8.1 Perché ottengo il messaggio "Cannot locate module for net-pf-x" (Non riesco a trovare il modulo per net-pf-x) quando lancio ifconfig?

Circa alla versione 1.3.80 del kernel, il codice per la gestione delle reti fu cambiato per permettere il caricamento di famiglie di protocolli (per esempio IPX, AX.25 e AppleTalk) come moduli. Ciò causò l'aggiunta di una nuova richiesta di `kerneld`: net-pf-X, dove X è un numero che identifica il protocollo (vedi `/usr/src/linux/include/linux/socket.h` per il significato dei vari numeri). Sfortunatamente ifconfig provoca in modo accidentale questi messaggi, cosicché molte persone ottengono un paio di messaggi di *log* quando il sistema viene avviato e viene lanciato ifconfig per configurare la periferica *loopback*. I messaggi non indicano nulla di pericoloso e puoi disabilitarli aggiungendo le linee:

```
alias net-pf-3 off      # Dimenticati di AX.25
alias net-pf-4 off      # Dimenticati di IPX
alias net-pf-5 off      # Dimenticati di AppleTalk
```

a `/etc/conf.modules`. Ovviamente, se utilizzi IPX come modulo, non devi aggiungere la linea che lo disabiliti.

8.2 Dopo la partenza di `kerneld` il mio sistema viene messo in ginocchio quando attivo una connessione ppp

Ci sono stati un paio di casi di questo problema. Sembra essere dovuto ad una sfortunata interazione fra `kerneld` e lo *script* **tkPPP** che viene usato su alcuni sistemi per impostare e configurare la connessione PPP (lo *script*, apparentemente, si morde la coda mentre esegue ifconfig). Questo richiama `kerneld` per cercare i moduli net-pf-x (vedi sopra), tenendo il sistema in sovraccarico e generando, eventualmente, un sacco si messaggi del tipo «Cannot locate module for net-pf-x» nel *log* di sistema. Non si conosce alcun modo per aggirare il problema, se non quello di evitare tkPPP o di cambiarlo usando qualche altro modo per tenere sotto controllo la connessione.

8.3 `kerneld` non carica il mio driver SCSI!

Aggiungi una voce per il tuo adattatore SCSI al tuo `/etc/conf.module`. Vai a vedere la descrizione della voce 5.2 (scsi_hostadapter) sopra.

8.4 modprobe si lamenta che «gcc2_compiled» non è definito

Questo è un errore nelle utility dei moduli che si verifica solo con le binutils 2.6.0.9 e seguenti, e che è anche documentato nelle note per le binutils. Dacci un'occhiata. Oppure procurati un aggiornamento per le utility dei moduli che porgano rimedio, come per esempio modules-2.0.0.

8.5 Il driver della mia scheda sonora non si ricorda delle impostazioni per il volume, etc.

Le impostazioni per un modulo sono salvate all'interno del modulo stesso quando viene caricato. Cosí, quando `kerneld` auto-scarica il modulo, ogni impostazione che hai fatto viene dimenticata e la volta successiva il modulo ritorna alle impostazioni di default.

Puoi dire a `kerneld` di configurare un modulo, eseguendo un programma, dopo che il modulo è stato auto-caricato. Vedere la 5.3 (sezione sopra) per la voce «post-install».

8.6 Dosemu ha bisogno di alcuni moduli. Come posso fare affinché `kerneld` li carichi?

Non puoi. Nessuna delle versioni di Dosemu (ufficiali o di sviluppo) supporta il caricamento dei moduli di Dosemu attraverso `kerneld`. Però, se stai utilizzando un kernel 2.0.26 o successivi, non hai piú bisogno di alcun modulo speciale Dosemu. Semplicemente aggiornati alla versione 0.66.1.

8.7 Perché ottengo un messaggio "Ouch, kerneld timed out, message failed" (Oops, tempo scaduto per kerneld, messaggio fallito)?

Quando il kernel invia una richiesta a `kerneld`, si aspetta di ottenere un «ricevuto» entro un secondo. Se `kerneld` non invia questo riscontro, il messaggio d'errore viene registrato nel *log* di sistema. La richiesta viene ritrasmessa e dovrebbe arrivare a destinazione.

Questo di solito capita su sistemi con un carico elevato (poiché `kerneld` è un processo eseguito in modalità utente, esso viene *schedulato* come ogni altro processo sul sistema). In momenti di alto carico, può non riuscire ad inviare per tempo il «ricevuto», prima che scada il tempo per kernel.

Se questo accade anche quando il carico è scarso, prova a far ripartire `kerneld`. Uccidi il processo `kerneld` e fallo ripartire ancora con il comando `/usr/sbin/kerneld`. Se il problema persiste, dovresti inviare un messaggio con l'errore a <*linux-kernel@vger.rutgers.edu*>, ma **ti prego** di assicurarti che la tua versione di kernel e `kerneld` sia aggiornata prima di spedire messaggi sul problema.

8.8 mount non aspetta che `kerneld` carichi il modulo per il *filesystem*

Ci sono stati un certo numero di casi per i quali il comando mount(8) non aspettava che `kerneld` finisse di caricare il modulo del *filesystem*. lsmod mostrava che `kerneld` aveva caricato il modulo e, ripetendo il comando mount subito dopo, questo in effetti veniva fatto. Questo sembra fosse dovuto ad un errore nella versione 1.3.9f delle utility per i moduli che affliggeva alcuni utenti Debian (può essere eliminato procurandosi una versione successiva delle utility per i moduli).

8.9 `kerneld` fallisce nel caricare il modulo ncpfs

Hai bisogno di compilare le utility per ncpfs con l'opzione -DHAVE_KERNELD. Vedere il Makefile di ncpfs.

8.10 `kerneld` fallisce nel caricare il modulo per smbfs

Stai usando un versione delle utility per smbmount troppo vecchia. Procurati l'ultima versione (0.10 o successive) da `<ftp://tsx-11.mit.edu/pub/linux/filesystems/smbfs/>`

8.11 Ho compilato tutto come modulo e ora il mio sistema non si avvia piú

8.12 `kerneld` fallisce nel caricare il *filesystem* principale

Non puoi rendere modulare **tutto**: il kernel deve avere abbastanza driver compilati normalmente affinché sia capace di fare il *mount* del *filesystem* principale ed eseguire i programmi necessari ad avviare `kerneld`. Cosí non puoi rendere modulare:

- il driver per l'hard-disk sul quale risiede il *filesystem* principale;
- il driver del *filesystem* stesso;
- il caricatore del formato binario di init, `kerneld` e altri programmi.

In effetti questo non è vero. L'ultimo kernel 1.3.x e tutti i 2.x supportano l'uso di un *ram-disk* iniziale che viene caricato da LILO o LOADLIN, ed è possibile caricare moduli da questo «disco» molto presto all'interno del processo di avvio. Il procedimento per farlo è descritto nel file Documentation/initrd.txt contenuto con i sorgenti del kernel.

8.13 `kerneld` non viene caricato all'avvio - si lamenta di libgdbm

Le versioni piú recenti di `kerneld` hanno bisogno delle librerie dbm di GNU, le libgdbm.so, per girare. Molte installazione hanno questo file in `/usr/lib`, mentre tu stai probabilmente lanciando `kerneld` prima che il *filesystem* sia stato montato. Un sintomo di questo è che `kerneld` non parte durante l'avvio (dai tuoi *script* rc), ma va bene se lo lanci manualmente dopo che il sistema è partito. La soluzione consiste nello spostare l'avvio di `kerneld` dopo che di `/usr` ne sia stato fatto il *mount* oppure spostare la libreria gdbm sul *filesystem* principale, per esempio in `/lib`.

8.14 Ottengo un "Cannot load module *xxx*" (Non posso caricare il modulo *xxx*), ma ho appena riconfigurato il kernel senza il supporto per *xxx*!

L'installazione Slackware (possibilmente altre) crea un file `/etc/rc.d/rc.modules` che opera un esplicito `modprobe` su una varietà di moduli. Quali moduli esattamente vengano testati dipende dalla configurazione originale del kernel. Hai probabilmente riconfigurato il tuo kernel escludendo uno o piú dei moduli che vengono testati in `rc.modules`, cosí il messaggio/i di errore. Aggiorna il tuo rc.modules «commentando» ogni modulo che non usi piú, o cancella `rc.modules` completamente e lascia che `kerneld` carichi i moduli quando ne ha bisogno.

8.15 Ho ricompilato il kernel e i moduli, e continuo ad ottenere messaggi di simboli non risolti all'avvio

Probabilmente hai riconfigurato e ricompilato il tuo kernel escludendo alcuni moduli. Hai ancora alcuni vecchi moduli che non utilizzi piú sparsi in giro per il directory `/lib/modules`. La soluzione piú semplice è quella di cancellare la directory `/lib/modules/x.y.z` e dare un altro `make modules_install` dalla directory dei sorgenti del kernel. Nota che questo problema capita solo quando si riconfigura il kernel senza cambiamenti di versione. Se vedi questo errore quando ti stai aggiornando ad una versione piú nuova, hai un altro tipo di problema.

8.16 Ho installato Linux 2.1 e ora non riesco piú a caricare alcun modulo

Linux 2.1 è il kernel attualmente in fase di sviluppo. Come tale ci si può aspettare che le cose cambino da un momento all'altro. Una delle cose che è cambiata in maniera significativa è il modo con il quale vengono trattati i moduli e dove il kernel e i moduli vengono caricati in memoria. Inoltre ora è Richard Henderson che si occupa dello sviluppo dei moduli del kernel.

In breve, se vuoi utilizzare i moduli con un kernel 2.1, devi:

- leggere il file Documentation/Changes e vedere quali pacchetti hanno bisogno di essere aggiornati sul tuo sistema
- usare l'ultimo pacchetto modutils, disponibile da `<ftp://ftp.redhat.com/pub/alphabits/>` o dal sito *mirror* presso `<ftp://tsx-11.mit.edu/pub/linux/packages/alphabits/>`

Ti raccomando di utilizzare almeno un kernel 2.1.29 se vuoi usare i moduli con un kernel 2.1.

8.17 E a proposito del collegamento alla rete su richiesta?

`kerneld` originariamente aveva qualche supporto per stabilire una connessione di rete in *dial-up* su richiesta; provando a spedire pacchetti ad una rete senza essere connessi, faceva lanciare a `kerneld` lo *script* `/sbin/request_route` per istituire la connessione PPP o SLIP.

Questo si rivelò essere una cattiva idea. Alan Cox, uno dei famosi nel *networking* di Linux, scrisse sulla *mailing list* linux-kernel:

> Il materiale su request_route è obsoleto, inefficiente e inutile [...]. Inoltre è stato rimosso dalla struttura 2.1.x.

Invece di usare lo *script* request_route e `kerneld`, voglio vivamente consigliarti di installare il pacchetto *diald* `<http://www.dna.lth.se/~erics/diald.html>` di Eric Schenk.

9 Messaggio di copyright

This document is Copyright (c) Henrik Storner, 1996, 1997.

Unless otherwise stated, Linux HOWTO documents are copyrighted by their respective authors. Linux HOWTO documents may be reproduced and distributed in whole or in part, in any medium physical or electronic, as long as this copyright notice is retained on all copies. Commercial redistribution is allowed and encouraged; however, the author would like to be notified of any such distributions.

All translations, derivative works, or aggregate works incorporating any Linux HOWTO documents must be covered under this copyright notice. That is, you may not produce a derivative work from a HOWTO and impose additional restrictions on its distribution. Exceptions to these rules may be granted under certain conditions; please contact the Linux HOWTO coordinator at the address given below.

In short, we wish to promote dissemination of this information through as many channels as possible. However, we do wish to retain copyright on the HOWTO documents, and would like to be notified of any plans to redistribute the HOWTOs.

> *L'unica licenza valida è quella originale in lingua inglese. Di seguito ne trovate una traduzione abbastanza fedele che però non ha alcun valore.*

Questo HOWTO è di Henrik Storner (1996-97) ed è distribuito, come gli altri HOWTO di Linux, sotto i termini descritti qui di seguito.

Finché non diversamente asserito, i documenti HOWTO di Linux sono proprietà dei rispettivi autori. I documenti degli HOWTO di Linux possono essere riprodotti e distribuiti in tutto o in parte, con ogni mezzo fisico o elettronico, finché questo avviso di copyright è mantenuto su tutte le copie. La distribuzione commerciale è permessa e incoraggiata; comunque all'autore piacerebbe essere avvisato di ogni distribuzione di questo tipo.

Ogni traduzione, lavoro derivato o comprendente ogni documento degli HOWTO di Linux deve essere coperto sotto questo avviso di copyright. Cioé non potete produrre un lavoro derivato da un HOWTO e imporre restrizioni aggiuntive sulla sua distribuzione. Eccezioni a queste regole possono essere garantite sotto certe condizioni; contattate il coordinatore degli HOWTO di Linux all'indirizzo indicato sotto.

In breve vogliamo promuovere la distribuzione di queste informazioni attraverso quanti piú canali possibile. Ciò nonostante vogliamo che rimanga il copyright sui documenti HOWTO e vorremmo venir avvisati di ogni progetto per la ridistribuzione degli HOWTO.

Se avete domande contattate Tim Bynum, il coordinatore degli HOWTO di Linux, all'indirizzo di posta elettronica *<linux-howto@sunsite.unc.edu>*.