Linux kerneld mini-HOWTO

Henrik Storner

kerneld-howto@linuxdoc.org

Diario delle Revisioni

Revisione v2.0 22 Maggio 2000
conversione da HTML a DocBook SGML.

1. A proposito del kerneld mini-HOWTO

Questo documento spiega come installare e utilizzare il caricatore automatico di moduli per il kernel "kerneld". L'ultima versione originale rilasciata di questo documento può essere trovata presso il Linux Documentation Project (http://tldp.org).

[NdT: la traduzione italiana è reperibile presso il sito dell'Italian Linux Documentation Project (http://it.tldp.org). Traduzione a cura di Lorenzo Cappelletti (lorenzo.cappelletti(at)email.it) e revisione a cura di Luca Bruno (luca(at)unofree.it)]

1.1. Ringraziamenti

Questo documento è basato su un HTML originale versione 1.7 datata 19 luglio 1997 di Henrik Storner `<storner@osiris.ping.dk>` ed è stata rivista e tradotta per il DocBook DTD da Gary Lawrence Murphy `<garym@teledyn.com>` 20 maggio 2000.

Le persone seguenti hanno contribuito in qualche modo a questo mini-HOWTO:

- 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

Se si dovessero trovare degli errori in questo documento, si prega di spedire una e-mail a `<kerneld-howto@linuxdoc.org>`. Commenti, incoraggiamenti e suggerimenti sono benvenuti e apprezzati e aiutano ad assicurare che questa guida rimanga aggiornata ed accurata.

2. Cos'è kerneld?

kerneld è una caratteristica introdotta durante lo sviluppo dei kernel 1.3 da Bjorn Ekwall. Essa permette ai moduli del kernel come driver di periferica, driver di rete e filesystem di venir caricati all'occorrenza automaticamente invece di doverlo fare manualmente con **modprobe** o **insmod**.

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

- può essere impostato per lanciare un programma utente, invece del solito schermo nero, permettendo così di usare qualsiasi programma come uno screen-saver;
- in modo analogo al supporto dell'oscuramento dello schermo, è anche possibile cambiare il «beep» di console in qualche cosa di completamente differente.

kerneld consiste di due componenti:

- 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 le richieste del kernel.

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

2.1. 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 c'è la necessità di compilare dei kernel per svariati sistemi che si distinguono di poco, come ad esempio differenti tipi di schede di rete, è possibile preparare un singolo kernel e alcuni moduli invece di essere costretti a compilare un kernel per ogni sistema.
- I moduli sono più 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, cioè si possiede più memoria disponibile per le applicazioni. La memoria usata dal kernel non è *mai* trasferita alla memoria di swap, così, 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 di Linux non devono compilare 284 differenti immagini di boot: ogni utente carica i driver di cui ha bisogno per il suo hardware. Distribuzioni Linux più moderne rileveranno l'hardware in uso e caricheranno solamente i moduli effettivamente necessari.

Certamente ci sono anche ragioni per le quali si potrebbe non volerlo usare (ad esempio quando si preferisce un unico file d'immagine per il kernel con tutti i driver compilati staticamente). In tal caso state leggendo il documento sbagliato.

2.2. Dove posso prelevare i componenti necessari?

Il supporto nel kernel di Linux fu introdotto con Linux 1.3.57. Se si possiede una versione di kernel precedente, sarà necessario aggiornarla se si desidera il supporto per kerneld. Tutti i maggiori siti ftp di Linux contengono i sorgenti del kernel inclusi:

- Kernel.Org Archive (ftp://ftp.kernel.org/pub/linux/kernel/)
- Metalab Linux Archive (ftp://metalab.unc.edu/pub/Linux/kernel/)
- TSX-11 presso il MIT (ftp://tsx-11.mit.edu/pub/linux/sources/system/)

Il demone a livello utente è incluso nel pacchetto modules. Questi sono normalmente disponibili dallo stesso sito in cui sono stati trovati i sorgenti del kernel.

> **Nota:** se si vuole provare a caricare i moduli con gli ultimi kernel in fase di *sviluppo*, è necessario procurarsi il più recente pacchetto modutils e non modules. Controllare sempre il file `Documentation/Changes` nei sorgenti del kernel per il numero di versione minimo richiesto per la propria immagine del kernel. Si veda anche la sezione sui problemi comuni per i problemi che si possono avere con i moduli ed il kernel 2.1.

3. Come impostare il tutto?

Primo procurarsi i componenti necessari: un kernel adatto e l'ultimo pacchetto di modules. Successivamente si devono installare le utilità per i moduli come indicato nelle istruzioni incluse nel pacchetto. Molto semplice: decomprimere i sorgenti e lanciare **make install**. Questo compila e installa in `/sbin` i seguenti programmi: **genksysm**, **insmod**, **lsmod**, **modprobe**, **depmod** e **kerneld**. Si raccomanda di aggiungere le seguenti linee ai propri script di avvio che effettuano alcune impostazioni necessarie ogni volta che Linux viene avviato. Aggiungere le seguenti linee al proprio file `/etc/rc.d/rc.S` (se si utilizza Slackware) o a `/etc/rc.d/rc.sysinit` (se si utilizza SysVinit, cioè Debian, Corel, RedHat, Mandrake o 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/kerneld ]
        then
                /sbin/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
```

Questi comandi potrebbero essere già installati nel proprio script di inizializzazione SysV. La prima parte lancia kerneld. La seconda parte chiama **depmod -a** all'avvio per costruire una lista di moduli disponibili e analizzare le loro inter-dipendenze. La mappa di depmod quindi dice a kerneld 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 si abilita questa opzione quando si compilano le utilità per i moduli, *kerneld non partirà se libgdbm non è disponibile*, caso che potrebbe accadere se si possiede `/usr` su una partizione separata e si fa partire kerneld prima che `/usr` sia stata montata. La soluzione consigliata è di spostare `libgdbm` da `/usr/lib` a `/lib` o linkarla staticamente a kerneld.

Successivamente decomprimere i sorgenti del kernel, configurarlo e compilarne uno che sia soddisfacente. Se questa operazione non è mai stata fatta prima, è necessario leggere attentamente il file README che si trova al primo livello dei sorgenti di Linux. Quando si lancia *make config* per configurare il kernel, si presti attenzione ad alcune domande che appaiono verso l'inizio:

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

Si rende necessario selezionare il supporto per i moduli caricabili altrimenti non ci saranno moduli per kerneld da caricare! Rispondere 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. Si vedranno domande come:

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

dove si può rispondere con una M che sta per "Modulo". Generalmente solo i driver necessari per far partire il sistema dovrebbero essere compilati nel kernel; gli altri possono essere compilati come moduli.

Driver essenziali

I driver essenziali all'avvio del sistema devono essere compilati nel cuore del kernel e non possono venir caricati come moduli. Tipicamente questi includeranno i driver per l'hard-disk e il driver per il filesystem principale. Se si possiede una macchina con più sistemi operativi e ci si basa su file che si trovano su una partizione estranea, è necessario compilare anche il supporto per quel tipo di filesystem nel cuore del kernel.

Una volta finito con **make config**, compilare ed installare il nuovo kernel e i moduli con **make dep clean bzlilo modules modules_install**.

Fiuuu!!!

Compilare un'immagine del kernel: Il comando **make zImage** mette la nuova immagine del kernel nel file `/arch/i386/boot/zImage`. Sarà quindi necessario copiarla dove risiede la propria immagine di boot e installarla manualmente con LILO.

Per maggiori informazioni su come configurare, compilare e installare il proprio kernel, controllare il Kernel-HOWTO postato regolarmente in `comp.os.linux.answers` e disponibile sul Linux Documentation Project (http://www.linuxdoc.org) e relativi mirror.

3.1. Proviamo kerneld

Ora si faccia il reboot con il nuovo kernel. Quando il sistema è di nuovo pronto, si impartisca un **ps ax** con il quale si dovrebbe vedere una linea per kerneld:

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

Una delle cose carine con kerneld è che, una volta che il kernel e il demone sono stati installati, poche impostazioni sono ancora necessarie. Per un inizio si provi ad usare uno dei driver compilati come modulo (in generale tutto funziona senza ulteriori configurazioni). Personalmente ho compilato il driver per il floppy come modulo, così posso mettere un dischetto DOS nel drive e digitare

```
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
```

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 è effettivamente caricato, è possibile lanciare **/sbin/lsmod** che lista tutti i moduli attualmente caricati:

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

"(autoclean)" sta ad indicare che il modulo verrà automaticamente rimosso da kerneld dopo che non viene usato per più di un minuto. Così 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 più di un minuto, verranno liberate). Alquanto carino, se sei a corto di memoria per le proprie applicazioni!

4. Come fa kerneld a sapere quale modulo caricare?

Nonostante kerneld venga fornito con informazioni già pronte sui tipi più comuni di moduli, ci sono situazioni in cui 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 c'è più di un possibile modulo da poter caricare.

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

- driver di periferica a blocchi
- driver di periferica a caratteri
- formato binario
- discipline di linea tty
- filesystem
- periferica di rete
- servizio di rete (per esempio rarp)
- protocollo di rete (per esempio IPX)

Kerneld determina quale modulo dobba essere caricato cercando nel file di configurazione `/etc/conf.modules`[1]. 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 per un dato servizio). Se non si possiede già questo file, è possibile crearlo con il comando:

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

Se si deisdera aggiungere un'altra direttiva «path» ai percorsi di default, *si deve 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 si avrà 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 si desidera aggiungere un «alias» o una direttiva «option», le nuove voci in `/etc/conf.modules` verranno *aggiunte* a quelle che modprobe già conosce. Se si deve *ridefinire* un «alias» o «options», le nuove voci in `/etc/conf.modules` sovrascriveranno quelle precompilate.

4.1. Periferiche a blocchi

Se viene eseguito **/sbin/modprobe -c**, si otterrà una lista di moduli conosciuti da kerneld e di richieste alle quali questi corrispondono. Per esempio, la richiesta che termina con il caricamento del driver per il floppy è per la periferica a blocchi con 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 dispositivi con major 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 nulla per default del driver per l'unità a nastro. Infatti non è presente nella lista ottenuta con **/sbin/modprobe -c**. Così 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

È possibile 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 si necessita di passare alcune opzioni al driver, per esempio per informare il modulo su quale IRQ la scheda di rete stia usando, aggiungere 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 si carica.

4.4. Formati binari

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

```
alias binfmt-267 binfmt_aout
```

poiché il magic number per i file ZMAGIC è 267. Se si controlla il file `/etc/magic` si troverà 0413. Si tenga presente che `/etc/magic` usa numeri in formato ottale, mentre kerneld usa il formato decimale e il numero 413 in base ottale corrisponde al numero decimale 267.

In realtà ci sono tre varianti leggermente diverse per gli eseguibili a.out (NMAGIC, QMAGIC e ZMAGIC), così per un pieno supporto del modulo binfmt_aout abbiamo bisogno di:

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

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

4.5. Discipline di linea (slip, cslip e ppp)

Le discipline 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 si desidera che kerneld carichi il modulo bsd_comp per la compressione dei dati per ppp, allora è necessario aggiungere le due linee seguenti al proprio `/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 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`). Così per caricare automaticamente il modulo IPX è necessario aggiungere al file `/etc/conf.modules` una linea come questa:

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

Si veda anche la sezione riguardante i problemi comuni per informazioni su come evitare alcuni noiosi messaggi all'avvio relativi a famiglie di protocolli indefiniti.

4.7. I file system

Le richieste di kerneld per i filesystem sono semplicemente i nomi del tipo di filesystem. Un comune uso potrebbe essere quello di caricare il modulo isofs per il filesystem del CD-ROM, cioè per il 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 di una periferica per un modulo.

- Periferiche a caratteri a major number 10: periferiche varie
- Periferiche SCSI
- Periferiche che richiedono inizializzazioni speciali

5.1. char-major-10 : mouse, watchdog e casualità

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

- mouse di vario genere (bus mouse, mouse PS/2),
- periferiche watchdog,
- il dispositivo `random` del kernel,
- interfaccia APM (Advanced Power Management).

Queste periferiche sono controllate da altrettanti moduli, 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      # For PS/2 mouse
alias char-major-10-130 wdt      # For WDT watchdog
```

Si necessita 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 si utilizza (per esempio un hard-disk, un CD-ROM o un'unità a nastro). Tutti possono essere caricati come moduli. Perciò, quando desidera 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 nè, tanto meno, quale modulo caricare per supportare il controller SCSI.

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

```
alias scd0 sr_mod                # sr_mod for SCSI CD-ROM's ...
alias scsi_hostadapter aha1542   # ... need the Adaptec driver
```

Questo funziona solo con versioni del kernel 1.3.82 o superiori.

Il metodo va bene se si possiede un solo controller SCSI. Se ce ne sono più d'uno, le cose diventano un po' più difficili.

In generale è possibile fare in modo che kerneld carichi un driver per un adattatore SCSI se un driver per un altro adattatore SCSI è già installato. Si è costretti a compilare entrambi i driver direttamente nel kernel (non come moduli) o caricare i moduli manualmente.

> **Suggerimento:** Tuttavia un modo per caricare più driver SCSI *esiste*. James Tsiao ha avuto questa idea:
>
> > È facile fare in modo che kerneld carichi il secondo driver scsi: basta impostare le dipendenze in `modules.dep` a mano. Serve solo 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 le periferiche scsi generiche. L'altra faccia della medaglia sta nel fatto che **depmod -a** non può rilevare automaticamente queste dipendenze, così 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.

Si faccia presente che questa tecnica funziona solo se si possiedono 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 sufficiente: la voce `post-install` entry

Qualche volta caricare solo il modulo non è abbastanza per far funzionare le cose. Per esempio, se la seconda scheda sonora è stata compilata 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 (<`bgallia@luc.edu`>):

> La soluzione definitiva ha richiesto l'installazione del pacchetto setmix (ftp://sunsite.unc.edu/pub/Linux/apps/sound/mixers/) e l'aggiunta delle 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`. Così 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, si necessità di una versione di kerneld 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 (più 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 proprio sistema.

6. Spiare kerneld

Se è stata provata qualsiasi cosa e non è proprio stato possibile immaginare cosa il kernel sta 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'utilità kdstat.

Questo piccolo e grazioso programma fa parte del pacchetto dei moduli, ma non viene compilato nè installato per default. Per ottenerlo è sufficiente andare nella directory dove risiedono i sorgenti di kerneld e digitare **make kdstat**. Poi, per fare in modo che kerneld mostri informazioni su cosa sta facendo, lanciare **kdstat debug** e kerneld comincerà a vomitare messaggi sulla console riferendo cosa sta facendo. Se poi si prova ad eseguire il comando che si desidera utilizzare, si vedranno comparire le richieste di kerneld; queste possono essere messe in `/etc/conf.modules` utilizzando un alias per il modulo necessario a completare il lavoro.

Per fermare la fase di debug, eseguire un **/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` del pacchetto dei moduli ha un paio di patch per il supporto di kerneld di screen-saver e del beep della console; queste non fanno ancora parte del kernel ufficiale, così sarà necessario installarle e ricompilare il kernel.

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

```
cd /usr/src/linux
patch -s -p1 /usr/src/modules-*/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 file può essere qualsiasi cosa, per esempio uno script di shell che lancia il proprio screen-saver preferito).

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. Ricordarsi di ripristinare lo schermo alla modalità testo originale!

8. Problemi comuni e cose di cui non si capisce il motivo

1. Perché ottengo il messaggio Cannot locate module for net-pf-X (Non riesco a trovare il modulo per net-pf-X) quando lancio **/sbin/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 di loopback. Questi non indicano nulla di pericoloso ed è possibile disabilitarli aggiungendo le linee:

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

a `/etc/conf.modules`. Ovviamente, se si utilizza IPX come modulo, non si deve aggiungere la linea che lo disabilita.

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 riportati per questo problema. Sembra essere dovuto ad una sfortunata interazione fra kerneld e lo script tkPPP che viene usato su alcuni sistemi per impostare e monitorare la connessione PPP. Apparentemente lo script 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.

3. kerneld non carica il mio driver SCSI!

Aggiungere una voce per il proprio adattatore SCSI al file `/etc/conf.module`. Si veda la descrizione della voce `scsi_hostadapter` più sopra.

4. modprobe si lamenta che gcc2_compiled non è definito

Questo è un errore nelle utilità 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. Si legga il documento in proposito oppure ci si procuri un aggiornamento per le utilità dei moduli che porgano rimedio.

5. Il driver della mia scheda sonora continua a dimenticarsi le impostazioni per il volume, etc.

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

Si può dire a kerneld di configurare un modulo eseguendo un programma dopo che il modulo è stato auto-caricato. Vedere il comando pre/post-install alla voce `post-install`.

6. DOSEMU ha bisogno di alcuni moduli. Come posso fare affinché kerneld li carichi?

Non è possibile. Nessuna delle versioni di dosemu (ufficiali o di sviluppo) supporta il caricamento dei moduli di dosemu attraverso kerneld. Però, se si utilizza un kernel 2.0.26 o successivi, non c'è più bisogno di alcun modulo speciale dosemu. Semplicemente aggiornarsi alla versione 0.66.1.

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 finalmente 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 carico elevato, 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, si provi a far ripartire kerneld. Uccidere il processo kerneld e farlo ripartire ancora con il comando **/usr/sbin/kerneld**. Se il problema persiste, si dovrebbe inviare un messaggio con l'errore a `<linux-kernel@vger.rutgers.edu>`, ma *si prega* di accertarsi che la propria versione di kernel, kerneld e delle utilità per i moduli sia aggiornata prima di spedire messaggi sul problema. Controllare i prerequisiti in `linux/Documentation/Changes`.

8. mount non aspetta che kerneld carichi il modulo per il filesystem

Ci sono stati un certo numero di casi riportati per i quali il comando mount(8) non aspettava che kerneld finisse di caricare il modulo del filesystem. **lsmod** mostra che kerneld carica il modulo e, ripetendo il comando mount subito dopo, questo in effetti viene fatto. Questo sembra essere dovuto ad un errore nella versione 1.3.9f delle utilità per i moduli che affligge alcuni utenti Debian. Può essere eliminato procurandosi una versione successiva delle utilità per i moduli.

9. kerneld fallisce nel caricare il modulo `ncpfs`

È necessario compilare le utilità per ncpfs con l'opzione -DHAVE_KERNELD. Vedere il `Makefile` di ncpfs.

10. kerneld fallisce nel caricare il modulo per `smbfs`

Si sta usando un versione delle utilità per smbmount troppo vecchia. Ci si procuri l'ultima versione (0.10 o successive) dall'archivio SMBFS su TSX-11 (ftp://tsx-11.mit.edu/pub/linux/filesystems/smbfs/)

11. Ho compilato tutto come modulo e ora il mio sistema non si avvia più o kerneld non riesce a caricare il modulo del root filesystem!

Non è possibile 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[2]. Così 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.

12. kerneld non viene caricato all'avvio; si lamenta di libgdbm

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

13. 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 (probabilmente anche 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. Probabilmente si è riconfigurato il kernel escludendo uno o più dei moduli che vengono testati nel proprio rc.modules, da cui il messaggio/i di errore. Aggiornare il proprio `rc.modules` commentando ogni modulo che non viene più utilizzato, o cancellare `rc.modules` completamente e lasciare che kerneld carichi i moduli quando ne ha bisogno.

14. Ho ricompilato il kernel ed i moduli ma continuo ad ottenere messaggi di simboli non risolti all'avvio

Probabilmente il kernel è stato riconfigurato/ricompilato escludendo alcuni moduli, ma ci sono ancora alcuni vecchi moduli non più utilizzati sparsi in giro per la directory `/lib/modules`. La soluzione più 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 cambiare versione. Se si nota questo errore quando si sta eseguendo un aggiornamento ad una versione più nuova, il tipo di problema è un altro.

15. Ho installato Linux 2.1/2.3 e ora non riesco più a caricare *nessun* modulo!

Numeri dispari di Linux indicano kernel di sviluppo. Come tale ci si può aspettare che le cose smettano di funzionare di quando in quando. 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.

In breve, se si desidera utilizzare i moduli con un kernel di sviluppo, si deve:

- leggere il file `Documentation/Changes` e vedere quali pacchetti hanno bisogno di essere aggiornati sul sistema
- usare l'ultimo pacchetto modutils, disponibile sotto AlphaBits da Red Hat (ftp://ftp.redhat.com/pub/alphabits/) o dal sito mirror TSX-11 (ftp://tsx-11.mit.edu/pub/linux/packages/alphabits/)

Si raccomanda di utilizzare almeno un kernel 2.1.29 se si desidera usare i moduli con un kernel 2.1.

16. 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; provare a spedire pacchetti ad una rete senza essere connessi, faceva lanciare a kerneld lo script `/sbin/request_route` per instaurare la connessione PPP o SLIP.

Questo si rivelò essere una cattiva idea. Alan Cox, personaggio illustre nel networking di Linux, scrisse sulla mailing-list linux-kernel:

> Il materiale su request_route è obsoleto, inefficiente e inutile [...]. Inoltre è stato rimosso dall'albero 2.1.x.

Invece di usare lo script request-route e kerneld, consiglio vivamente di installare il pacchetto diald (http://www.dna.lth.se/~erics/diald.html) di Eric Schenk.

Note

1. Alcune distribuzioni chimano questo file `modules.conf`
2. 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 in `linux/Documentation/initrd.txt` contenuto con i sorgenti del kernel.