The Linux Kernel HOWTO

Brian Ward, bri@cs.uchicago.edu v1.0, 5 giugno 1999

Questa è una guida dettagliata alla configurazione, compilazione, aggiornamento e risoluzione dei problemi del kernel per sistemi basati su i386. Traduzione a cura di Giovanni Bortolozzo, *borto@pluto.linux.it*.

Indice

1 Introduzione **3**

1.1 Per prima cosa leggi qua! . . . 3

1.2 Due parole sullo stile . . . 4

2 Domande importanti e relative risposte **4**

2.1 Ad ogni modo, il kernel cosa fa? . . . 4

2.2 Perché dovrei voler aggiornare il mio kernel? . . . 4

2.3 Che tipo di hardware supportano i nuovi kernel? . . . 4

2.4 Di che versione di gcc e libc ho bisogno? . . . 4

2.5 Cos'è un modulo caricabile? . . . 5

2.6 Di quanto spazio ho bisogno sul disco? . . . 5

2.7 Quanto tempo ci mette? . . . 5

3 Come configurare realmente il kernel **5**

3.1 Procurarsi i sorgenti . . . 5

3.2 Decomprimere i sorgenti . . . 5

3.3 Configurare il kernel . . . 6

3.3.1 Kernel math emulation (Processor type and features) . . . 6

3.3.2 Enhanced (MFM/RLL) disk and IDE disk/cdrom support (Block Devices) . . . 6

3.3.3 Networking support (General Setup) . . . 7

3.3.4 System V IPC (General Setup) . . . 7

3.3.5 Processor family (Processor type and features) . . . 7

3.3.6 SCSI support . . . 7

3.3.7 Network device support . . . 7

3.3.8 Filesystems . . . 8

3.3.9 Character devices . . . 8

3.3.10 Sound . . . 9

3.3.11 Altre opzioni di configurazione . . . 9

3.3.12 Kernel hacking . . . 9

3.4 E ora cosa? (Il Makefile) . . . 9

4 Compilare il kernel **9**
4.1 Far pulizia e sistemare le dipendenze . . . 9
4.2 Tempo di compilazione . . . 10
4.3 Altre “make”abilità . . . 10
4.4 Installare il kernel . . . 10

5 “Rattoppare” il kernel **11**
5.1 Applicare una patch . . . 11
5.2 Se qualcosa va storto . . . 12
5.3 Sbarazzarsi dei file .orig . . . 12
5.4 Altre patch . . . 12

6 Pacchetti aggiuntivi **13**
6.1 kbd . . . 13
6.2 util-linux . . . 13
6.3 hdparm . . . 13
6.4 gpm . . . 13

7 Alcuni trabocchetti **14**
7.1 make clean . . . 14
7.2 Kernel grossi o lenti . . . 14
7.3 La porta parallela non funziona/la mia stampante non funziona . . . 14
7.4 Il kernel non si compila . . . 14
7.5 La nuova versione del kernel non fa il boot . . . 14
7.6 Si è dimenticato di lanciare LILO oppure il sistema proprio non si avvia . . . 15
7.7 Dice ‘warning: bdflush not running’ . . . 16
7.8 Non riesco a far funzionare il mio lettore CD-ROM IDE/ATAPI . . . 16
7.9 Dice strane cose a proposito di richieste di instradamento . . . 16
7.10 Il firewalling non funziona in 1.2.0 . . . 16
7.11 “Not a compressed kernel Image file” . . . 16
7.12 Problemi con i terminali console dopo l’aggiornamento al 1.3.x . . . 16
7.13 Non riesco più a compilare quasi niente dopo l’upgrade del kernel . . . 17
7.14 Incrementare i limiti . . . 17

8 Note per l’aggiornamento alle versioni 2.0.x, 2.2.x **17**

9 Moduli **17**
9.1 Installare le utility per i moduli . . . 17
9.2 Moduli distribuiti con il kernel . . . 18

10 Suggerimenti e trucchi **18**

10.1 Ridirezionare l'output dei comandi make o patch 18

10.2 Installazioni condizionali del kernel 19

10.3 Aggiornamenti del kernel 19

11 Altri HOWTO di rilievo che possono essere utili **19**

12 Varie **20**

12.1 Autore 20

12.2 Da fare 20

12.3 Contributi 20

12.4 Questioni legali (in inglese) 21

1 Introduzione

Dovresti leggere questo documento? Beh, vedi se presenti uno dei seguenti sintomi:

- "Arg! Questo pacchetto wizzo-46.5.6 dice che ha bisogno del kernel 2.8.183 e io ho ancora l'1.0.9!"
- Nei kernel più recenti c'è un device driver che assolutamente devi avere.
- Non hai assolutamente idea di come compilare il kernel.
- "La roba nel README è *veramente* l'intera storia?"
- L'hai preso, c'hai provato, ma non funziona.
- Hai bisogno di qualcosa da dare alla gente che insiste nel domandarti di installargli il loro kernel.

1.1 Per prima cosa leggi qua!

Alcuni degli esempi in questo documento assumono che si abbia `tar`, `find` e `xargs` della GNU. Sono abbastanza standard e quindi ciò non dovrebbe causare problemi. È pure sottinteso che si possieda una certa conoscenza della struttura del filesystem del proprio sistema; se non la si conosce, è critico che si conservi una copia scritta dell'output del comando `mount` durante il normale funzionamento del sistema (o un listato di `/etc/fstab`, se si può leggerlo). Queste informazioni sono importanti, e non cambiano a meno che non si ripartizioni il disco, se ne aggiunga uno nuovo, si reinstalli il sistema o qualcos'altro di simile.

Mentre redigevo questo documento, l'ultima versione del kernel di "produzione" era la 2.2.9, il che significa che i riferimenti e gli esempi corrispondono a quella release. Anche se ho provato a rendere questo documento il più possibile indipendente dalla versione, il kernel è costantemente in sviluppo e quindi se ci si procura una nuova versione, inevitabilmente ci saranno alcune differenze. Si noti che ciò non dovrebbe comunque creare grossi problemi, ma potrebbe creare un po' di confusione.

Ci sono due versioni dei sorgenti del kernel Linux, "produzione" e "sviluppo". Le versioni produzione sono le release con il numero secondario pari; 1.2.x era di produzione, come pure il 2.0.x e il 2.2.x. Al rilascio, questi kernel sono considerati versioni piuttosto stabili ed esenti da bug. I kernel in sviluppo (2.1.x, 2.3.x, ecc.) sono intesi come kernel di test, per gente che vuole provare kernel nuovi e probabilmente molto bacati. Si è avvisati.

1.2 Due parole sullo stile

`Il testo come questo` è qualcosa che appare nel proprio schermo, un nome di un file, oppure qualcosa che può essere digitato direttamente, come un comando o un'opzione a un comando (se si sta leggendo la versione testuale di questo documento non si noterà alcuna differenza). Comandi e altri input sono spesso messi tra virgolette (con ' '), il che causa il seguente classico problema di punteggiatura: se una di queste cose appare citata alla fine di una frase, la gente spesso digita un '.' assieme con il comando, perché lo stile di citazione americano dice di mettere il punto dentro le virgolette. Anche se il buon senso (e sfortunatamente questo assume che qualcuno con "buon senso" sia adotto al cosiddetto stile di citazione americano) dovrebbe dire di togliere prima il punto, molta gente semplicemente non se lo ricorda, e così in questi casi l'ho messo fuori dalle virgolette. In altre parole, quando indico che si dovrebbe digitare "`make config`" scriverò '`make config`', e non '`make config`.' (NdT: in italiano questo problema non si pone: il punto è comunque sempre messo fuori dalle virgolette.)

2 Domande importanti e relative risposte

2.1 Ad ogni modo, il kernel cosa fa?

Il kernel Unix funge da mediatore tra i programmi e l'hardware. Per prima cosa si occupa della (o organizza le cose per la) gestione della memoria per tutti i programmi in esecuzione (processi), e assicura che tutti abbiano una giusta (o ingiusta, a seconda dei gusti) porzione dei cicli del processore. Inoltre fornisce una buona interfaccia abbastanza portabile per far dialogare i programmi con l'hardware.

Ci sono certamente molte altre operazioni del kernel oltre a queste, ma queste funzioni basilari sono le più importanti da conoscere.

2.2 Perché dovrei voler aggiornare il mio kernel?

I kernel più nuovi generalmente offrono la possibilità di dialogare con un maggior numero di tipi di hardware (ovvero, hanno più device driver), possono avere una gestione migliore dei processi, possono girare più velocemente delle versioni più vecchie, possono essere più stabili delle versioni precedenti e possono correggere alcuni sciocchi bug delle vecchie versioni. La maggior parte della gente aggiorna il suo kernel perché vuole i device driver e le correzioni dei bug.

2.3 Che tipo di hardware supportano i nuovi kernel?

Si veda l'Hardware-HOWTO. In alternativa si può dare una scorsa al file '`config.in`' nei sorgenti di Linux o scoprirlo quando si fa '`make config`'. Quest'ultimo mostra tutto l'hardware supportato dalla distribuzione standard del kernel, ma non tutto quello che Linux supporta; molti device driver comuni (come i driver PCMCIA e alcuni driver di dispositivi a nastro) sono moduli caricabili mantenuti e distribuiti a parte.

2.4 Di che versione di gcc e libc ho bisogno?

Linus consiglia nel file `README` incluso con i sorgenti del kernel una versione di gcc. Se non si ha questa versione, la documentazione nella versione raccomandata di gcc dovrebbe dire se è necessario aggiornare la propria libc. Questa non è una procedura difficile, ma è importante seguire le istruzioni.

2.5 Cos'è un modulo caricabile?

Sono pezzi del codice del kernel non inclusi direttamente nel kernel. Si possono compilare separatamente e inserirli e toglierli da un kernel in esecuzione praticamente in qualsiasi momento. Grazie a questa flessibilità, questo è ora il modo preferito per programmare certe caratteristiche del kernel. Molti device driver popolari, come i driver PCMCIA e il driver per il dispositivo a nastro QIC-80/40, sono moduli caricabili.

2.6 Di quanto spazio ho bisogno sul disco?

Dipende dalla particolare configurazione del proprio sistema. Per prima cosa, i sorgenti compressi di Linux 2.2.9 sono quasi 14 megabyte. Molti li conservano anche dopo averli decompressi. La decompressione e la compilazione con una configurazione normale, occupano fino ad altri 67 MB.

2.7 Quanto tempo ci mette?

Con le macchine più nuove la compilazione dura decisamente meno tempo che con quelle vecchie; un AMD K6-2/300 con un disco veloce può creare un kernel 2.2.x in circa 4 minuti. Se si ha intenzione di compilarne uno con i vecchi Pentium, 486 e 386 ci si prepari ad aspettare anche ore, giorni.

Se ciò crea problemi, e capita di poter compilare su una macchina più veloce in giro per la rete, è possibile compilare il kernel nella macchina più veloce (assumendo che si siano specificati i parametri giusti, che le proprie utility siano aggiornate e così via) e poi trasferire l'immagine del kernel nella macchina più lenta.

3 Come configurare realmente il kernel

3.1 Procurarsi i sorgenti

È possibile ottenere i sorgenti in ftp anonimo da `ftp.kernel.org` nella directory `/pub/linux/kernel/vx.y`, dove `x.y` sono la versione (es. 2.2) e, come menzionato in precedenza, quelli che finiscono con un numero dispari sono le release di sviluppo e possono essere instabili. Tipicamente il file si chiama `linux-x.y.z.tar.gz`, dove `x.y.z` è il numero della versione. I siti solitamente conservano anche una versione con suffisso `.bz2`, che è stata compressa con bzip2 (questi file saranno più piccoli e richiederanno di minor tempo per il trasferimento).

La cosa migliore è usare `ftp.xx.kernel.org` dove `xx` è il codice della propria nazione; ad esempio `ftp.it.kernel.org` per l'Italia e `ftp.us.kernel.org` per gli Stati Uniti.

3.2 Decomprimere i sorgenti

Si faccia il login (o si usi `su`) come '`root`', e si passi (con `cd`) in `/usr/src`. Se si sono installati i sorgenti del kernel quando si è installato per la prima volta Linux (come fanno molti), là ci sarà già una directory chiamata '`linux`', che contiene l'intero vecchio albero dei sorgenti. Se si ha spazio su disco e si vuole andare sul sicuro, si conservi quella directory. Una buona idea è di scoprire quale versione gira nel proprio sistema e rinominare di conseguenza la directory. Il comando '`uname -r`' mostra la versione del kernel corrente. Quindi, se '`uname -r`' dice '`1.0.9`', si vorrà rinominare (con '`mv`') '`linux`' in '`linux-1.0.9`'. Se ci si sente abbastanza temerari, semplicemente si cancelli l'intera directory. In ogni caso, ci si assicuri che non esista una directory '`linux`' in `/usr/src` prima di decomprimere il codice sorgente.

Ora, in `/usr/src`, si decomprimano ("spacchettino") i sorgenti con '`tar zxpvf linux-x.y.z.tar.gz`' (se si ha semplicemente un file `.tar` senza `.gz` alla fine, basta '`tar xpvf linux-x.y.z.tar`'). Il contenuto dei sorgenti scorrerà sullo schermo. Al termine, ci sarà una nuova directory '`linux`' in `/usr/src`. Si faccia `cd` in `linux` e si dia un'occhiata al file `README`. Ci sarà una sezione etichettata '`INSTALLING the kernel`'. Si seguano le istruzioni quando appropriate: link simbolici da sistemare, rimozione di vecchi file `.o`, ecc.

Se si ha un file `.bz2` e il programma bzip2 (si legga a proposito a `http://www.muraroa.demon.co.uk/`), si faccia questo:

```
bz2cat linux-x.y.z.tar.bz2 | tar xvf -
```

3.3 Configurare il kernel

Nota: Alcune parti di questa sezione sono reiterazioni/chiarificazioni di una sezione simile nel file `README` di Linus.

Il comando '`make config`' lanciato in `/usr/src/linux` avvia uno script di configurazione che fa molte domande. Richiede la bash, e quindi si verifichi che bash sia `/bin/bash`, `/bin/sh` oppure `$BASH`.

Comunque, ci sono alcune alternative più carine a '`make config`' e molto facilmente le si troverà più facili e comode da usare. '`make menuconfig`' è probabilmente la più usata. Qualsiasi si scelga, è meglio che per prima cosa si familiarizzi con l'interfaccia poiché può capitare di doverci tornare prima di quanto si pensi. Quelli che "usano X", possono provare '`make xconfig`' se hanno installato Tk ('click-o-rama' - Nat). '`make menuconfig`' è per quelli che hanno le (n)curses e preferiscono un menu testuale. Queste interfacce hanno un vantaggio piuttosto evidente: se ci si incasina e si fa una scelta sbagliata durante la configurazione, è semplice tornare indietro e sistemarla.

Con '`make menuconfig`' e '`make xconfig`' le opzioni di configurazione appariranno in gerarchie.

Alle domande si risponde tipicamente con '`y`' (sì) o '`n`' (no). I device driver tipicamente hanno un'opzione '`m`', che significa "modulo", intendendo che il sistema lo compilerà, ma non direttamente dentro al kernel ma piuttosto come modulo caricabile. Un modo più comico per descriverla è come "maybe" (può essere). Alcune delle opzioni più ovvie e non critiche non sono qui descritte e per una breve descrizione di alcune altre si veda la sezione "Altre opzioni di configurazione". Con '`make menuconfig`', la barra spaziatrice attiva la selezione.

Nei kernel 2.0.x e successivi, c'è un'opzione '`?`', che fornisce una breve descrizione dei parametri di configurazione. Le informazioni probabilmente sono le più aggiornate. Qui di seguito un elenco di alcune importanti caratteristiche (tra parentesi la gerarchia alla quale appartengono) seguite da una breve spiegazione.

3.3.1 Kernel math emulation (Processor type and features)

Emulazione matematica a livello kernel (Tipo di processore e caratteristiche)

Se non si ha un coprocessore matematico (si ha un semplice 386 o un 486SX), si deve rispondere '`y`' a questa domanda. Se si ha un coprocessore e si risponde comunque '`y`', non ci si preoccupi troppo: è comunque usato il coprocessore ed è ignorata l'emulazione. Per qualsiasi macchina moderna, la risposta sarà negativa, ma non ci si preoccupi se accidentalmente si risponde sì; se non serve, non è usata.

3.3.2 Enhanced (MFM/RLL) disk and IDE disk/cdrom support (Block Devices)

Supporto per dischi Enhanced (MFM/RLL) e dischi/cdrom IDE (Dispositivi a blocchi)

Probabilmente si ha bisogno di questo supporto; significa che il kernel supporterà i dischi fissi standard dei PC, che la maggior parte della gente ha. Questo driver non comprende i dischi SCSI; vengono dopo nella configurazione.

Poi verrà chiesto del supporto per "old disk-only" (solo dischi vecchi) e dei driver "new IDE" (nuovi IDE). Se ne scelga uno; la maggior differenza è che il vecchio driver supporta solo due dischi in una interfaccia, mentre il nuovo supporta una interfaccia secondaria e un lettore cdrom IDE/ATAPI. Il nuovo driver è 4k più grande del vecchio e si suppone sia pure "migliorato", intendendo che a parte contenere un numero diverso di bug, potrebbe migliorare le prestazioni dei dischi, specialmente se si ha dell'hardware recente (di tipo EIDE).

3.3.3 Networking support (General Setup)

Supporto di rete (Impostazioni generali)

All'inizio, conviene rispondere semplicemente '`y`' se la propria macchina è in una rete come Internet o si vuole usare SLIP, PPP, term, ecc per avere l'accesso a Internet. Comunque, poiché molti pacchetti (come l'X Window system) richiedono il supporto di rete, si dovrebbe rispondere '`y`' anche se la propria macchina non risiede in una vera rete. Più tardi, verrà chiesto se si vuole il supporto per il networking TCP/IP; ancora, si risponda '`y`' anche lì a meno che non si sia assolutamente sicuri.

3.3.4 System V IPC (General Setup)

IPC System V (Impostazioni generali)

Una delle migliori definizioni di IPC (Interprocess Communication – Comunicazione Interprocessi) è nel glossario del Perl book. Non è da sorprendersi se alcuni programmatori Perl l'adoperano per permettere ai processi di parlarsi l'un l'altro, come fanno molti altri programmi (DOOM, fra gli altri), e quindi non è una buona idea rispondere `n` a meno che non si sappia esattamente quello che si sta facendo.

3.3.5 Processor family (Processor type and features)

(in older kernels: Use -m486 flag for 486-specific optimizations)

Famiglia del processore (Tipo di processore e caratteristiche)
(nei vecchi kernel: usare il flag -m486 per ottimizzazioni specifiche per il 486)

Tradizionalmente, si traduceva in alcune ottimizzazioni per un particolare processore; il kernel girava bene anche su altri chip, ma era un po' più grande. Comunque, nei nuovi kernel questo non è più vero, e quindi si può specificare il processore per il quale si sta compilando il kernel. Un kernel "386" funzionerà per tutte le macchine.

3.3.6 SCSI support

Supporto SCSI

Se si posseggono dispositivi SCSI, si risponda '`y`'. Saranno chieste ulteriori informazioni, come il supporto per CD-ROM, dischi e quale tipo di adattatore SCSI si possiede. Si veda lo SCSI HOWTO per maggiori dettagli.

3.3.7 Network device support

Supporto dei dispositivi di rete

Se si ha una scheda di rete o si vuole usare SLIP, PPP o un adattatore per porta parallela per connettersi a Internet, si risponda '`y`'. Lo script di configurazione chiederà poi il tipo di scheda che si possiede e quale protocollo usare.

3.3.8 Filesystems

Filesystem

Lo script di configurazione chiede se si vuole il supporto per i seguenti filesystem:

Standard (minix) – Le distribuzioni più nuove non creano filesystem minix e molti non lo usano più, ma può ancora essere una buona idea configurarlo. Alcuni programmi per la creazione di "dischi di ripristino" (rescue disk) lo usano e molti floppy hanno ancora un filesystem minix, poiché minix è meno doloroso da usare su un floppy.

Second extended – Questo è il filesystem standard di Linux. Sicuramente se ne ha uno di questo tipo e si deve rispondere '`y`'.

msdos – Se si vogliono usare partizioni MS-DOS del disco fisso, montare dischetti formattati in DOS, si risponda '`y`'.

Sono disponibili diversi altri tipi di filesystem di altri sistemi operativi.

/proc – (un'idea dei Bell Labs, penso). Uno non crea un filesystem proc in un disco; è un interfaccia filesystem al kernel e ai processi. Molti visualizzatori di processi (come '`ps`') lo usano. Qualche volta si provi '`cat /proc/meminfo`' o '`cat /proc/devices`'. Alcune shell (rc, in particolare) usano `/proc/self/fd` (noto come `/dev/fd` su altri sistemi) per l'I/O. Si deve certamente rispondere '`y`' a questa domanda; molti strumenti importanti ne dipendono.

NFS – Se la propria macchina vive in una rete e si vogliono usare filesystem su altri sistemi con NFS, si risponda '`y`'.

ISO9660 – Presente nella maggior parte dei CD-ROM. Se si ha un lettore CD-ROM e lo si vuole usare sotto Linux, si risponda '`y`'.

Ma io non so di quali filesystem ho bisogno! Ok, si digiti '`mount`'. L'output apparirà simile a questo:

```
blah# mount
/dev/hda1 on / type ext2 (defaults)
/dev/hda3 on /usr type ext2 (defaults)
none on /proc type proc (defaults)
/dev/fd0 on /mnt type msdos (defaults)
```

Si guardi ogni riga; la parola dopo '`type`' è il tipo di filesystem. In questo esempio, i miei filesystem `/` e `/usr` sono second extended, sto usando `/proc`, e c'è un dischetto montato usando il filesystem msdos (bleah).

Si può provare '`cat /proc/filesystems`' se si ha `/proc` attualmente abilitato; elencherà i filesystem presenti nel kernel corrente.

La configurazione di filesystem raramente usati e non critici può causare un ingrossamento del kernel; si veda la sezione sui moduli per un modo di evitarlo e la sezione "Trabocchetti" per capire perché non è desiderabile un kernel grosso.

3.3.9 Character devices

Dispositivi a caratteri

Qui si abilita il driver per la propria stampante (parallela), per il busmouse, per il mouse PS/2 (molti portatili usano il protocollo del mouse PS/2 per le loro trackball), per alcuni dispositivi a nastro e altri dispositivi "a carattere" di questo tipo. Si risponda '`y`' quando appropriato.

Nota: `gpm` è un programma che permette l'uso di un mouse fuori dall'ambiente X per il cut and paste ("taglia e incolla") tra le console virtuali. È piuttosto carino se si ha un mouse seriale, poiché coesiste bene con X, ma sono necessari alcuni altri trucchi per gli altri tipi di mouse.

3.3.10 Sound

Audio

Se si desidera ardentemente di sentire `biff` abbaiare, si risponda '`y`' e si potrà confessare tutto sulla propria scheda audio al programma di configurazione (una nota sulla configurazione della scheda audio: quando viene chiesto se si vuole installare la versione completa del driver, si può rispondere '`n`' risparmiando un po' di memoria scegliendo solo le caratteristiche che si giudicano necessarie).

Se si è seriamente interessati al supporto per la scheda audio, si dia un'occhiata sia al driver libero a `http://www.linux.org.uk/OSS/` sia all'Open Sound System commerciale a `http://www.opensound.com/`.

3.3.11 Altre opzioni di configurazione

Qui non sono state elencate tutte le opzioni di configurazione perché cambiano troppo spesso o sono piuttosto banali (per esempio, 3Com 3C509 support per compilare il device driver per questa particolare scheda Ethernet). Esista una lista esaustiva di tutte le opzioni (più un metodo per metterle nello script `Configure`) in un lavoro avviato e mantenuto da Axel Boldt (`boldt@math.ucsb.edu`) e nel suo help in linea. È pure disponibile come un unico grosso file in `Documentation/Configure.help` nel proprio albero dei sorgenti del kernel a partire dalla versione 2.0.

3.3.12 Kernel hacking

>Dal README di Linus:

> i dettagli della configurazione "kernel hacking" solitamente risultano in un kernel più grosso è più lento (o entrambi) e possono pure rendere il kernel meno stabile configurando alcune routine che provano attivamente a rompere il cattivo codice per trovare i problemi del kernel (kmalloc()). Quindi probabilmente si dovrebbe rispondere 'n' alla domanda nel caso di un kernel di "produzione".

3.4 E ora cosa? (Il Makefile)

Dopo aver finito la configurazione, un messaggio comunica che il proprio kernel è stato configurato, e di "controllare il `Makefile` di più alto livello per ulteriori configurazioni" ("check the top-level `Makefile` for additional configuration"), ecc.

Quindi, si dia un'occhiata al `Makefile`. Probabilmente non sarà necessario modificarlo, ma non può far male guardarlo. Si possono pure cambiare le sue opzioni con il comando '`rdev`' una volta installato il nuovo kernel. Se ci si perde guardando quel file, non è da preoccuparsi.

4 Compilare il kernel

4.1 Far pulizia e sistemare le dipendenze

Quando termina lo script di configurazione, dice pure di fare '`make dep`' e (possibilmente) '`clean`'. Quindi, si faccia il '`make dep`'. Ciò assicura che tutte le dipendenze, come i file include, siano a posto. Non ci mette tanto, a meno che il proprio computer non sia piuttosto lento. Per le vecchie versioni del kernel, quando terminava, si doveva fare un '`make clean`'. Ciò rimuoveva tutte i file oggetto e alcune altre cose che le vecchie versioni lasciano in giro. In ogni caso, *non* si dimentichi questo passo prima di provare a ricompilare il kernel.

4.2 Tempo di compilazione

Dopo il `dep` ed il `clean`, si può fare '`make bzImage`' o '`make bzdisk`' (questa è la parte che richiede un tempo più lungo). '`make bzImage`' compilerà il kernel, e lascerà (tra le altre cose) un file in `arch/i386/boot` chiamato '`bzImage`'. Questo è il nuovo kernel compresso. '`make bzdisk`' fa la stessa cosa, ma piazza pure il nuovo `bzImage` su un dischetto che si spera sia stato messo nel driver "A:". '`bzdisk`' è piuttosto comodo per verificare un nuovo kernel; se si pianta (o semplicemente non fa le cose giuste), semplicemente basta rimuovere il floppy e riavviare con il proprio vecchio kernel. È pure un modo comodo di fare il boot se accidentalmente si cancella il proprio kernel (o altre cose ugualmente terribili). Lo si può pure usare per installare un nuovo sistema quando si è semplicemente scaricato il contenuto di un disco in un altro ("tutto questo e molto altro! ORA quanto vorresti pagare?").

Tutti i kernel ragionevolmente recenti sono compressi, da qui il '`bz`' di fronte al nome. Un kernel compresso si decomprime da solo automaticamente quando eseguito.

Nei kernel più vecchi, non c'era l'opzione per costruire un `bzImage`; era semplicemente un `zImage`. Quella opzione al momento è ancora disponibile ma comunque, data la dimensione del codice dei nuovi kernel, ora è più o meno obbligatorio compilare un `bzImage` perché il vecchio metodo non riesce a gestire kernel troppo grossi.

4.3 Altre "make"abilità

'`make mrproper`' farà un '`clean` molto più accurato. Qualche volta è necessario; si può desiderare di farlo dopo aver applicato una patch. '`make mrproper`' cancellerà anche il file di configurazione e quindi conviene salvarselo (`.config`) se si vede che può essere utile.

'`make oldconfig`' proverà a configurare il kernel da un vecchio file di configurazione; eseguirà da solo tutto il processo '`make config`'. Se non si è mai compilato un kernel prima o non si possiede un vecchio file di configurazione, allora probabilmente non lo si dovrebbe fare, in quanto molto probabilmente si vorrà modificare la configurazione predefinita.

Si veda la sezione sui moduli per una descrizione di '`make modules`'.

4.4 Installare il kernel

Quando si ha un kernel che sembra funzionare nel modo desiderato, allora è il momento di installarlo. Molti usano LILO (Linux Loader) per questo. '`make bzlilo`' installerà il kernel, vi lancerà LILO e preparerà tutto per il boot, MA SOLO se lilo è configurato nel proprio sistema nel modo seguente: il kernel è `/vmlinuz`, lilo è in `/sbin` e il proprio file di configurazione di lilo (`/etc/lilo.conf`) è in accordo con queste cose.

Altrimenti, è necessario usare LILO direttamente. È un pacchetto abbastanza semplice da installare e da usare, ma ha la tendenza a confondere la gente con il file di configurazione. Si dia un'occhiata al file di configurazione (`/etc/lilo/config` per le vecchie versioni oppure `/etc/lilo.conf` per quelle nuove) e si veda qual è l'impostazione corrente. Il file di configurazione è simile a questo:

```
image = /vmlinuz
    label = Linux
```

```
        root = /dev/hda1
        ...
```

‘image =’ è impostato al kernel correntemente installato. La maggior parte della gente usa /vmlinuz. ‘label’ è usata da lilo per determinare quale kernel o sistema operativo avviare, e ‘root’ è la / di quel particolare sistema operativo. Si faccia una copia di backup del vecchio kernel e si copi il bzImage appena creato al suo posto (si dovrà dire ‘cp bzImage /vmlinuz’ se si usa ‘/vmlinuz’). Poi, si rilanci lilo: su un nuovo sistema si può semplicemente lanciare ‘lilo’, ma in uno vecchio si dovrà fare /etc/lilo/install o addirittura /etc/lilo/lilo -C /etc/lilo/config.

Se si vuol saperne di più sulla configurazione di LILO, o non si ha LILO, si prenda la versione più recente dal proprio sito ftp preferito e si seguano le istruzioni.

Per avviare uno dei propri vecchi kernel dal disco fisso (un altro modo per salvarsi nel caso si facciano casini con il nuovo), si copino le righe che seguono ‘image = xxx’ (compresa) alla fine del file di configurazione di LILO, e si modifichi ‘image = xxx’ in ‘image = yyy’, dove ‘yyy’ è il percorso completo del file in cui si fatto il backup del kernel. Poi si cambi ‘label = zzz’ in ‘label = linux-backup’ e si rilanci lilo. Può essere necessario mettere nel file di configurazione una riga che dice ‘delay=x’, dove x è il numero di decimi di secondo che LILO dovrà attendere prima di fare il boot, cosicché lo si può interrompere (con il tasto shift, per esempio), e inserire l’etichetta della immagine di boot di backup (nel caso succedano cose non proprio piacevoli).

5 “Rattoppare” il kernel

5.1 Applicare una patch

Gli aggiornamenti incrementali del kernel sono distribuiti come patch (“toppe”). Per esempio, se si ha la versione 1.1.45 e si ha notizia che c’è una ‘patch46.gz’ in giro, significa che si può aggiornare alla versione 1.1.46 applicando quella patch. Prima può essere una buona idea fare una copia di backup dell’albero dei sorgenti (‘make clean’ e poi ‘cd /usr/src; tar zcvf old-tree.tar.gz linux’ creerà un archivio compresso).

Così, continuando con l’esempio precedente, si supponga di avere ‘patch46.gz’ in /usr/src. Si faccia cd in /usr/src e si faccia ‘zcat patch46.gz | patch -p0’ (o ‘patch -p0 < patch46’ se la patch non è compressa). Si vedranno delle cose sfrecciare (o fluttuare, se il proprio sistema è abbastanza lento) che dicono che sta provando ad applicare dei pezzi (hunk) e quando ha o meno successo. Solitamente questa azione è troppo veloce per poterla leggere e non si è troppo sicuri quando ha funzionato o meno, quindi può essere un’idea usare l’opzione -s con patch, che dice a patch di riportare solo i messaggi d’errore. Per cercare le parti che non sono andate a buon fine, si passi in /usr/src/linux e si cerchino i file con estensione .rej. Alcune versioni di patch (versioni più vecchie che possono essere state compilate con un filesystem inferiore) lasciano i rifiuti (reject) con un estensione #. Si può usare ‘find’ per la ricerca:

```
    find .  -name '*.rej' -print
```

mostrerà nello standard output tutti i file residenti nella directory corrente e in ogni sottodirectory con estensione .rej.

Se tutto va a buon fine, di faccia un ‘make clean’, ‘config’, e ‘dep’ come descritto nelle sezioni 3 e 4.

Ci sono alcune altre opzioni per il comando patch. Come menzionato prima, patch -s sopprimerà tutti messaggi tranne quelli d’errore. Se si conservano i sorgenti del kernel in un posto diverso da /usr/src/linux, patch -p1 (in quella directory) applicherà correttamente la patch. Altre opzioni di patch sono ben documentate nella pagina man.

5.2 Se qualcosa va storto

(Nota: questa sezione fa riferimento principalmente a kernel piuttosto vecchi)

Il problema più frequente che era solito comparire era quando una patch modificava un file chiamato ‘`config.in`’ e questo non sembrava più essere a posto perché lo si era modificato per adattare le opzioni alla propria macchina. Di questo ora si tiene conto, si potrebbe ancora incontrare con vecchie release. Per correggerlo, si veda il file `config.in.rej`, e si veda cosa rimane della patch originale. Le modifiche saranno tipicamente marcate con ‘+’ e ‘-’ all’inizio della riga. Si vedano le righe là attorno e si ricordi se erano stato impostate a ‘`y`’ o ‘`n`’. Ora si editi il `config.in`, e si cambi ‘`y`’ in ‘`n`’ e ‘`n`’ in ‘`y`’ quando appropriato. Si faccia un

```
patch -p0 < config.in.rej
```

e se finisce con successo, allora si può continuare con la configurazione e la compilazione. Il file `config.in.rej` rimarrà, ma si può cancellarlo.

Se si incontrano ulteriori problemi, si può aver installato una patch “fuori uso”. Se patch dice ‘`previously applied patch detected: Assume -R?`’ (rilevata una patch applicata precedentemente: assumo -R?), probabilmente si sta provando ad applicare una patch precedente al numero di versione corrente; se si risponde ‘`y`’ proverà a degradare i sorgenti e molto probabilmente fallirà; sarà quindi necessario prendere in blocco un nuovo albero dei sorgenti (che poteva non essere una cattiva idea sin dall’inizio).

Per tornare indietro (disapplicare) di una patch, si usi ‘`patch -R`’ sulla patch originale.

La miglior cosa da fare quando le patch fanno realmente casino è di partire ancora da un albero dei sorgenti nuovo di zecca (per esempio, da uno dei file `linux-x.y.z.tar.gz`), e ricominciare.

5.3 Sbarazzarsi dei file .orig

Già dopo poche patch, i file `.orig` cominciano ad ammucchiarsi. Per esempio, un albero 1.1.51 che ho ripulito l’ultima volta quand’era 1.1.48: rimuovendo i file .orig ho recuperato oltre mezzo mega.

```
find .  -name '*.orig' -exec rm -f {} ';'
```

se ne occuperà al vostro posto. Le versioni di `patch` che usano `#` per i reject usano una tilde invece di `.orig`.

Ci sono modi migliori per sbarazzarsi dei file `.orig`, che dipendono da GNU `xargs`:

```
find .  -name '*.orig' | xargs rm
```

o il metodo “abbastanza sicuro ma un po’ più prolisso”:

```
find . -name '*.orig' -print0 | xargs --null rm --
```

5.4 Altre patch

Esistono altre patch (io le chiamo “non standard”) oltre a quelle che distribuisce Linus. Se le si applica, le patch di Linus potrebbero non funzionare correttamente e si dovrà o tornare indietro, correggere i sorgenti o la patch, installare un nuovo albero dei sorgenti, o una combinazione di queste cose. Ciò può divenire abbastanza frustrante, se non si vogliono modificare i sorgenti (con la possibità di risultati pessimi), si rimuovano le patch non standard prima di applicare quelle di Linus, o semplicemente si installi un nuovo albero. Poi, si può provare a vedere se le patch non standard funzionano ancora. Se non lo fanno, o si resta con il vecchio kernel, giochicchiando con la patch e i sorgenti per farli funzionare, o si aspetta (probabilmente elemosinando) che esca una nuova versione della patch.

Quanto comuni sono le patch che non sono nella distribuzione standard? Probabilmente se n'è sentito parlare. Ero avvezzo ad usare la patch noblink per le mie console virtuali perché odio il cursore lampeggiante (questa patch è – o almeno era – aggiornata frequentemente per le nuove release del kernel). Con la maggior parte dei device driver sviluppati come moduli caricabili, la frequenza di patch "non standard" è diminuita significativamente.

6 Pacchetti aggiuntivi

Il proprio kernel linux ha molte caratteristiche che non sono spiegate negli stessi sorgenti del kernel; queste caratteristiche sono tipicamente utilizzate attraverso pacchetti esterni. Sono qui elencati alcuni dei più comuni.

6.1 kbd

La console di Linux probabilmente ha più caratteristiche che meriti. Tra queste ci sono la possibilità di cambiare i font, rimappare la propria tastiera, cambiare modalità video (nei kernel più nuovi), ecc. Il pacchetto kbd contiene programmi che permettono all'utente di fare tutto questo, oltre a numerosi font e mappe di tastiera per praticamente qualsiasi tastiera. È disponibile dagli stessi siti che conservano i sorgenti del kernel.

6.2 util-linux

Rik Faith (`faith@cs.unc.edu`) ha messo assieme una grossa collezione di utility per Linux che per una strana coincidenza è stata chiamata util-linux. Sono ora mantenute da Andries Brouwer (`util-linux@math.uio.no`). Disponibili in ftp anonimo da sunsite.unc.edu in `/pub/Linux/system/misc`, contengono programmi come `setterm`, `rdev`, e `ctrlaltdel`, che sono importanti per il kernel. Come dice Rik, *non le si installi senza pensare*; non è necessario installare tutto nel pacchetto, si potrebbero veramente causare seri problemi se lo si fa.

6.3 hdparm

Come con molti pacchetti, questo era un tempo una patch per il kernel e un programma di supporto. La patch è stata integrata nel kernel ufficiale e il programma per ottimizzare o giochicchiare con il proprio disco fisso è ora distribuito separatamente.

6.4 gpm

gpm sta per general purpose mouse. Questo programma permette di fare il cut and paste tra le console virtuali e altre cosette con un gran numero di tipi di mouse.

7 Alcuni trabocchetti

7.1 make clean

Se il proprio nuovo kernel fa veramente cose strane dopo un upgrade, c'è la possibilità che si sia dimenticato di fare `make clean` prima di compilare il nuovo kernel. I sintomi possono essere qualsiasi, da crash senza motivo, a strani problemi di I/O fino a prestazioni pietose. Ci si assicuri di fare anche un `make dep`.

7.2 Kernel grossi o lenti

Se il proprio kernel si succhia un sacco di memoria, è troppo grande e/o ci mette un'eternità per compilarsi anche se si è preso il nuovo Quadbazillium-III/4400, probabilmente si sono configurate un sacco di cose non necessarie (device driver, filesystem, ecc). Se non le si usa, non le si configuri, perché non fanno altro che occupare memoria. Il sintomo più ovvio di un kernel cicciotto è il continuo swap tra la memoria e il disco; se il proprio disco sta facendo un sacco di rumore e non è uno di quei vecchi Fujitsu Eagles che sembrano jet che atterrano quando vengono spenti, si dia un'occhiata alla configurazione del kernel.

Si può scoprire quanta memoria sta usando il kernel prendendo il totale della memoria nella propria macchina e sottraendogli l'ammontare di "total mem" in `/proc/meminfo` o l'output del comando '`free`'.

7.3 La porta parallela non funziona/la mia stampante non funziona

Le opzioni di configurazioni per i PC sono: primo, sotto la categoria 'General Setup', si selezioni 'Parallel port support' e 'PC-style hardware'. Poi sotto a 'Character devices', si selezioni 'Parallel printer support'.

Poi ci sono i nomi. Linux 2.2 chiama il dispositivo della stampante in maniera differente dalle release precedenti. Come risultato finale, se si aveva un `lp1` sotto il proprio vecchio kernel, probabilmente è un `lp0` in quello nuovo. Si usi '`dmesg`' o si dia una scorsa ai log in `/var/log` per capire il problema.

7.4 Il kernel non si compila

Se non si compila, allora probabilmente una patch ha fallito o i propri sorgenti sono in qualche modo corrotti. Inoltre potrebbe non essere corretta la propria versione di gcc, o potrebbe essere essa stessa corrotta (per esempio, i file include potrebbero contenere degli errori). Ci si assicuri che i link simbolici che Linus descrive nel `README` siano impostati correttamente. In generale, se un kernel standard non si compila, nel sistema c'è qualcosa di seriamente sbagliato, ed è probabilmente necessaria la reinstallazione di alcuni strumenti.

In alcuni casi, gcc può andare in crash a causa di problemi hardware. Il messaggio d'errore è qualcosa di simile a "xxx exited with signal 15" e in genere sembrerà molto misterioso. Probabilmente non lo menzionerei se non mi fosse successo una volta – avevo della memoria cache difettata e il compilatore occasionalmente andava in palla a caso. Si provi per prima cosa a reinstallare gcc se si è soggetti al problema. Si dovrebbe essere sospettosi solo se il kernel si compila bene disabilitando la cache esterna, con un ammontare di RAM ridotto, ecc.

Suggerire alla gente che il loro hardware ha dei problemi tende a disturbare i sonni. Bene, non lo farò più. C'è una FAQ per questo – è a `http://www.bitwizard.nl/sig11/`.

7.5 La nuova versione del kernel non fa il boot

Non si è lanciato LILO o non lo si è configurato correttamente. Una cosa che mi ha fregato una volta è stato un problema nel file di configurazione; diceva '`boot = /dev/hda1`' invece di '`boot = /dev/hda`' (questa cosa potrebbe essere un po' noiosa all'inizio ma una volta che si ha un file di configurazione funzionante, non sarà più necessario modificarlo).

7.6 Si è dimenticato di lanciare LILO oppure il sistema proprio non si avvia

Ooops! La cosa migliore che si può fare qui è di fare il boot da un dischetto o da un CDROM e preparare un altro dischetto avviabile (come farebbe '`make zdisk`'). È necessario sapere dov'è il proprio filesystem di root (`/`) e di quale tipo è (e.g. second extended, minix). Nell'esempio seguente è necessario conoscere anche il filesystem in cui risiede il proprio albero dei sorgenti `/usr/src/linux`, il suo tipo e dov'è normalmente montato.

Nell'esempio seguente `/` è `/dev/hda1` e il filesystem che contiene `/usr/src/linux` è `/dev/hda3`, normalmente montato in `/usr`. Entrambi sono filesystem second extended. L'immagine del kernel funzionante è in `/usr/src/linux/arch/i386/boot` ed è chiamata `bzImage`.

L'idea è che se esiste un `bzImage` funzionante, è possibile usarlo per un nuovo dischetto. Un'altra alternativa, che potrebbe o no funzionare bene (dipende dal modo in cui si è incasinato il proprio sistema) è discussa dopo l'esempio.

Per prima cosa si avvii dalla combinazione di un boot e un root disk, oppure da un rescue disk, e si monti il filesystem che contiene l'immagine del kernel funzionante:

```
mkdir /mnt
mount -t ext2 /dev/hda3 /mnt
```

Se `mkdir` dice che la directory esiste già semplicemente si ignori l'errore. Ora, si faccia `cd` nel posto dov'era il kernel funzionante. Si noti che

```
/mnt + /usr/src/linux/arch/i386/boot - /usr = /mnt/src/linux/arch/i386/boot
```

Si inserisca un dischetto formattato nel drive A: (non il proprio boot o root disk!), di scarichi l'immagine nel dischetto e lo si configuri per il proprio filesystem di root:

```
cd /mnt/src/linux/arch/i386/boot
dd if=bzImage of=/dev/fd0
rdev /dev/fd0 /dev/hda1
```

Si faccia `cd` in `/` e si smonti il filesystem `/usr` normale:

```
cd /
umount /mnt
```

Ora si dovrebbe essere in grado di riavviare il proprio sistema normalmente da questo floppy. Non si dimentichi di lanciare lilo (o correggere qualsiasi altra cosa che si è fatto di sbagliato) dopo aver riavviato!

Come menzionato in precedenza c'è un'altra alternativa comune. Se capita di avere un'immagine funzionante del kernel in `/` (`/vmlinuz` per esempio), la si può usare per un dischetto di boot. Si suppongano tutte le condizioni precedenti e che la nostra immagine del kernel sia `/vmlinuz`, si facciano solamente le seguenti modifiche all'esempio precedente: si cambi `/dev/hda3` in `/dev/hda1` (nel filesystem `/`), `/mnt/src/linux` in `/mnt`, e `if=bzImage` in `if=vmlinuz`. La nota che spiega come derivare `/mnt/src/linux` può essere ignorata.

L'uso di LILO con dischi grossi (più di 1024 cilindri) può causare problemi. Si veda il LILO mini-HOWTO o la documentazione per aver aiuto in proposito.

7.7 Dice 'warning: bdflush not running'

Questo può essere un problema serio. A partire dalle release del kernel dopo la 1.0 (attorno al 20 aprile 1994), è stato aggiornato/rimpiazzato un programma chiamato '`update`' che scarica periodicamente i buffer del filesystem. Si prendano i sorgenti di '`bdflush`' (lo si dovrebbe trovare dove si sono presi i sorgenti del kernel) e lo si installi (probabilmente è meglio far funzionare il sistema con il vecchio kernel mentre lo si fa). Si installa come '`update`' e dopo un riavvio, il nuovo kernel non dovrebbe più protestare.

7.8 Non riesco a far funzionare il mio lettore CD-ROM IDE/ATAPI

Piuttosto strano, ma un sacco di gente non riesce a far funzionare i loro lettori ATAPI, probabilmente perché ci sono un po' di cose che vanno storte.

Se il proprio CD-ROM è il solo dispositivo in una particolare interfaccia IDE, dovrebbe essere impostato come "master" o "single". Questo è l'errore più comune.

Creative Labs (per dirne una) ora ha messo interfacce IDE sulle sue schede audio. Comunque, questo porta all'interessante problema che mentre alcuni hanno solo una interfaccia, molti hanno due interfacce IDE integrate nelle loro schede madri (solitamente all'IRQ15), così è pratica comune rendere l'interfaccia della soundblaster una terza porta IDE (IRQ11).

Ciò causa problemi con Linux in quelle versioni 1.2.x che non supportano una terza interfaccia IDE (c'è il supporto a partire da qualche parte nella serie 1.3.x, ma si ricordi che è in sviluppo ed inoltre non rileva automaticamente le interfacce). Per andare a capo di questa cosa non si hanno molte scelte.

Se si ha già una seconda porta IDE, è possibile che non la si stia usando o che non abbia già connessi due dispositivi. Si tolga il lettore ATAPI dalla scheda audio e lo si metta nella seconda interfaccia. Si può poi disabilitare l'interfaccia della scheda audio, il che fa risparmiare anche un IRQ.

Se non si ha una seconda interfaccia, si configuri (con i ponticelli) l'interfaccia della scheda audio (non la parte relativa alla scheda audio) come IRQ15, la seconda interfaccia. Dovrebbe funzionare.

7.9 Dice strane cose a proposito di richieste di instradamento

Ci si procuri una nuova versione del programma `route` e di qualsiasi altro programma per manipolare gli instradamenti. `/usr/include/linux/route.h` (che in realtà è un file in `/usr/src/linux`) è cambiato.

7.10 Il firewalling non funziona in 1.2.0

Si aggiorni almeno alla versione 1.2.1.

7.11 "Not a compressed kernel Image file"

"*Non è un file Immagine del kernel compressa*"

Non si usi il file `vmlinux` creato in `/usr/src/linux` come propria immagine di boot; quello giusto è `[..]/arch/i386/boot/bzImage`.

7.12 Problemi con i terminali console dopo l'aggiornamento al 1.3.x

Si cambi la parola `dumb` in `linux` nella voce termcap della console in `/etc/termcap`. Si potrebbe dover pure creare una voce terminfo.

7.13 Non riesco più a compilare quasi niente dopo l'upgrade del kernel

I sorgenti del kernel Linux includono numerosi file include (quelle robe che finiscono con `.h`) ai quali viene fatto riferimento dagli include standard in `/usr/include`. Solitamente ne viene fatto riferimento in questo modo (dove `xyzzy.h` vorrebbe essere qualcosa in `/usr/include/linux`):

```
#include <linux/xyzzy.h>
```

Solitamente c'è un link chiamato `linux` in `/usr/include` alla directory `include/linux` del propri sorgenti del kernel (in un sistema tipico `/usr/src/linux/include/linux`). Se non c'è questo link, o punta in un posto sbagliato, la maggior parte delle cose non si compileranno affatto. Se si è deciso che i sorgenti del kernel occupano troppo spazio su disco e li si è cancellati allora ovviamente il problema sarà questo. Un'altra cosa che potrebbe non andare bene sono i permessi dei file; se il proprio utente `root` ha una umask che per default non permette agli altri utenti di vedere i suoi file e si sono estratti i sorgenti del kernel senza l'opzione `p` (preserva i permessi dei file), quegli utenti non saranno in grado di usare il compilatore C. Sebbene per correggere questa cosa si possa usare il comando `chmod`, probabilmente è più facile riestrarre i file include. Lo si può fare nello stesso modo dei sorgenti completi all'inizio, aggiungendo solamente un argomento.

```
blah# tar zxvpf linux.x.y.z.tar.gz linux/include
```

Nota: "`make config`" creerà il link `/usr/src/linux` se non c'è.

7.14 Incrementare i limiti

I pochi comandi di *esempio* che seguono possono essere utili a quanti si domandano come incrementare alcuni limiti imposti dal kernel:

```
echo 4096 > /proc/sys/kernel/file-max
echo 12288 > /proc/sys/kernel/inode-max
echo 300 400 500 > /proc/sys/vm/freepages
```

8 Note per l'aggiornamento alle versioni 2.0.x, 2.2.x

I kernel delle versioni 2.0.x e 2.2.x introducono un po' di modifiche per l'installazione del kernel. Il file `Documentation/Changes` nell'albero dei sorgenti del 2.0.x contiene informazioni per entrambe le versioni. Molto probabilmente si dovranno aggiornare diversi pacchetti chiave, come gcc, libc e SysVInit, e forse modificare alcuni file di sistema.

9 Moduli

I moduli caricabili del kernel possono far risparmiare memoria e facilitare la configurazione. Lo scopo dei moduli è cresciuto fino a includere filesystem, driver per schede Ethernet, driver per dispositivi a nastro, driver per la stampante e altro.

9.1 Installare le utility per i moduli

Le utility per i moduli sono disponibili nello stesso posto dove si sono presi i sorgenti del proprio kernel come `modutils-x.y.z.tar.gz`; si scelga il più alto livello di patch `x.y.z` che sia minore o uguale a quello del proprio kernel corrente. Le si decomprima con '`tar zxvf modutils-x.y.z.tar.gz`', si faccia `cd` nella directory creata (`modutils-x.y.z`), si legga il file `README` e si seguano le istruzioni di installazione (solitamente qualcosa di semplice, come `make install`). Ora si dovrebbero avere i programmi `insmod`, `rmmod`, `ksyms`, `lsmod`, `genksyms`, `modprobe` e `depmod` in `/sbin`. Se si vuole si verifichino le utility con il driver di esempio "hw" nella directory `insmod`; si veda il file `INSTALL` in quella sottodirectory per i dettagli.

`insmod` inserisce un modulo dentro un kernel in esecuzione. I moduli solitamente hanno un'estensione `.o`; il driver di esempio summenzionato è chiamato `drv_hello.o`, così per inserirlo uno dovrebbe usare '`insmod drv_hello.o`'. Per vedere i moduli che il kernel sta attualmente usando si usi `lsmod`. L'output è simile a questo:

```
blah# lsmod
Module:        #pages:  Used by:
drv_hello         1
```

'`drv_hello`' è il nome del modulo, usa una pagina (4k) di memoria e al momento nessun'altro modulo del kernel ne dipende. Per rimuovere questo modulo, si usi '`rmmod drv_hello`'. Si noti che `rmmod` vuole un *nome di modulo*, non un nome di file; lo si prenda dall'elenco di `lsmod`. Lo scopo delle altre utility per i moduli è documentato nelle relative pagine man.

9.2 Moduli distribuiti con il kernel

Dalla versione 2.0.30, praticamente qualsiasi cosa è disponibile come modulo caricabile. Per usarli, per prima cosa ci si assicuri di non configurarli dentro il kernel; ovvero, non si risponda `y` ad essi durante il '`make config`'. Si compili un nuovo kernel e si riavvii. Poi, si faccia ancora `cd` in `/usr/src/linux` e si esegua un '`make modules`'. Questo compila tutti i moduli che si sono specificati nella configurazione del kernel e mette dei link a essi in `/usr/src/linux/modules`. Li si può usare direttamente da quella directory o eseguire '`make modules_install`', che li installa in `/lib/modules/x.y.z`, dove `x.y.z` è la versione del kernel.

Questo può essere utile specialmente con i filesystem. Si potrebbero non usare frequentemente i filesystem minix o msdos. Per esempio, se incappo in un dischetto msdos, potrei fare `insmod /usr/src/linux/modules/msdos.o` e poi `rmmod msdos` quando ho finito. Questa procedura fa risparmiare circa 50k di RAM nel kernel durante le normali operazioni. Un piccola nota al contorno per il filesystem minix: si dovrà *sempre* configurarlo direttamente dentro il kernel per l'uso nei dischetti di ripristino.

10 Suggerimenti e trucchi

10.1 Ridirezionare l'output dei comandi make o patch

Se si vuole una registrazione di quel che hanno fatto i comandi '`make`' o '`patch`', è possibile ridirezionare l'output su un file. Per prima cosa si scopra quale shell si usa: '`grep root /etc/passwd`' e si cerchi qualcosa di simile a '`/bin/csh`'.

Se si usa sh o bash,

```
(comando) 2>&1 | tee (file di output)
```

piazzerà una copia dell'output di `(comando)` nel file '`(file di output)`'.

Per csh o tcsh, si usi

```
(comando) |& tee (file di output)
```

Per rc (Nota: probabilmente non si usa rc) è

```
(comando) >[2=1] | tee (file di output)
```

10.2 Installazioni condizionali del kernel

Oltre a usare dischetti, ci sono diversi metodi per verificare un nuovo kernel senza toccare quello vecchio. Diversamente da molte altre versioni di Unix, LILO ha la possibilità di avviare il kernel ovunque si trovi nel disco (se si ha un disco grande – oltre i 500 MB – invito a leggere la documentazione di LILO in quanto questo può causare dei problemi). Quindi, se si aggiunge qualcosa del tipo

```
image = /usr/src/linux/arch/i386/boot/bzImage
    label = new_kernel
```

alla fine del file di configurazione di LILO, si può scegliere se avviare un kernel appena compilato senza toccare il vecchio `/vmlinuz` (dopo aver eseguito `lilo`, naturalmente). Il modo più semplice di dire a LILO di avviare un nuovo kernel è di premere il tasto shift all'avvio (quando viene mostrato `LILO` nello schermo e niente altro), che dà l'accesso a un prompt. A questo punto, si può inserire '`new_kernel`' per avviare il nuovo kernel.

Se si vogliono mantenere versioni differenti dell'albero dei sorgenti nel proprio sistema (attenzione, questa cosa può occupare un *sacco* di spazio nel disco), il modo più comune è di chiamarle `/usr/src/linux-x.y.z`, dove `x.y.z` è la versione del kernel. Si può poi "selezionare" un albero di sorgenti con un link simbolico; per esempio '`ln -sf linux-1.2.2 /usr/src/linux`' renderà l'albero 1.2.2 quello corrente. Prima di creare un link simbolico come questo, ci si assicuri che l'ultimo argomento di `ln` non sia una directory (se è un vecchio link simbolico non c'è problema), altrimenti il risultato potrebbe non essere quello che ci si aspetta.

10.3 Aggiornamenti del kernel

Russell Nelson (`nelson@crynwr.com`) riassume le modifiche nelle nuove versioni del kernel. Sono corte e sarebbe bene darci un'occhiata prima di aggiornare. Sono disponibili in ftp anonimo da `ftp.emlist.com` in `pub/kchanges` o tramite l'URL

```
http://www.crynwr.com/kchanges
```

11 Altri HOWTO di rilievo che possono essere utili

- Sound-HOWTO: schede audio e utility
- SCSI-HOWTO: tutto sui controller e i dispositivi SCSI
- NET-2-HOWTO: networking
- PPP-HOWTO: networking PPP in particolare
- PCMCIA-HOWTO: sui driver per il portatile
- ELF-HOWTO: ELF: cos'è, convertirsi..
- Hardware-HOWTO: panoramica dell'hardware supportato
- Module mini-HOWTO: altro ancora sui moduli del kernel
- Kerneld mini-HOWTO: su kerneld
- BogoMips mini-HOWTO: nel caso si voglia fantasticare

12 Varie

12.1 Autore

L'autore e il manutentore del Linux Kernel-HOWTO è Brian Ward (`bri@cs.uchicago.edu`). Invito a inviarmi qualsiasi commento, aggiunta e correzione (in particolare le correzioni sono le cose per me più importanti).

Si può dare un'occhiata alla mia 'home page' in uno di questi URL:

```
http://www.math.psu.edu/bri/
http://blah.math.tu-graz.ac.at/~bri/
```

Anche se ho provato a essere il più possibile sollecito con le mail, si ricordi che ne ricevo *un sacco* tutti i giorni, e quindi mi ci vuole un po' di tempo per rispondere. Specialmente quando mi si invia una domanda, invito a sforzarsi di essere il più possibile chiari e dettagliati nei propri messaggi. Se si sta scrivendo a proposito di hardware non funzionante (o qualcosa di simile), ho bisogno di conoscere qual è la configurazione hardware. Se si riporta un errore, non si dica semplicemente "ho provato questo ma mi ha dato un errore"; ho bisogno di sapere qual era l'errore. Vorrei anche sapere quale versione del kernel, gcc e libc si sta usando. Se mi si dice semplicemente che si usa questa o quella distribuzione, non mi si dice poi molto. Non mi preoccupo se si fanno domande semplici; si ricordi che se non si chiede non si avrà mai una risposta! Voglio ringraziare tutti quelli che mi hanno dato un qualche riscontro.

Se le domande non sono relative al kernel, o sono in una qualche lingua che non capisco, potrei non rispondere.

Se mi si scrive e non si riceve risposta in un tempo ragionevole (tre settimane o più), allora è possibile che io abbia cancellato accidentalmente il messaggio (spiacente). Si riprovi.

Ho ricevuto un sacco di mail su cose che in realtà sono problemi hardware. Vanno bene, ma si tenga in conto che non ho familiarità con tutto l'hardware esistente al mondo. Uso processori AMD, controller SCSI Adaptec e Symbios e dischi SCSI IBM.

La versione 0.1 è stata scritta il 3 ottobre 1994. Questo documento è disponibile nei formati SGML, PostScript, TeX, roff e testo.

12.2 Da fare

La sezione "Suggerimenti e trucchi" è un po' piccola. Spero di espanderla con suggerimenti da altri.

Così è anche "Pacchetti addizionali".

Sono necessarie più informazioni sul debugging/recupero dai crash.

12.3 Contributi

È inclusa una piccola parte del README di Linus (opzioni di kernel hacking). (Grazie, Linus!)

`uc@brian.lunetix.de` (Ulrich Callmeier): patch -s e xargs.

`quinlan@yggdrasil.com` (Daniel Quinlan): correzioni e aggiunte in molte sezioni.

`nat@nat@nataa.fr.eu.org` (Nat Makarevitch): mrproper, tar -p, molte altre cose

`boldt@math.ucsb.edu` (Axel Boldt): ha raccolto le descrizioni delle opzioni di configurazione del kernel dalla rete; poi mi ha passato la lista.

`lembark@wrkhors.psyber.com` (Steve Lembark): suggerimenti sui boot multipli.

`kbriggs@earwax.pd.uwa.edu.au` (Keith Briggs): alcune correzioni e suggerimenti

`rmcguire@freenet.columbus.oh.us` (Ryan McGuire): aggiunte sulle “makeabilità”.

`dumas@excalibur.ibp.fr` (Eric Dumas): traduzione in francese.

`simazaki@ab11.yamanashi.ac.jp` (Yasutada Shimazaki): traduzione in giapponese.

`jjamor@lml.ls.fi.upm.es` (Juan Jose Amor Iglesias): traduzione in spagnolo.

`mva@sbbs.se` (Martin Wahlen): traduzione in svedese.

`jzp1218@stud.u-szeged.hu` (Zoltan Vamosi): traduzione in ungherese.

`bart@mat.uni.torun.pl` (Bartosz Maruszewski): traduzione in polacco.

`donahue@tiber.nist.gov` (Michael J Donahue): errori di battitura, vincitore della “sliced bread competition”.

`rms@gnu.ai.mit.edu` (Richard Stallman): concetti/notizie di distribuzione sulla documentazione “libera”.

`dak@Pool.Informatik.RWTH-Aachen.DE` (David Kastrup): cose sull’NFS.

`esr@snark.thyrsus.com` (Eric Raymond): diverse notizie piccanti.

Anche la gente che mi ha spedito mail con domande e problemi mi è stata abbastanza utile.

12.4 Questioni legali (in inglese)

Recommendations: Commercial redistribution is allowed and encouraged; however, it is strongly recommended that the redistributor contact the author before the redistribution, in the interest of keeping things up-to-date (you could send me a copy of the thing you’re making while you’re at it). Translators are also advised to contact the author before translating. The printed version looks nicer. Recycle.