

Quinta dispensa: ricompilare il kernel

[(c)2002 – Jean François Panico]

In una precedente dispensa abbiamo parlato delle origini di Linux soffermandoci sul fatto che esso è soltanto il kernel del sistema operativo GNU/Linux. Abbiamo anche accennato al ruolo che ricopre il kernel; in questo articolo approfondiremo alcuni lati di questa parte del sistema operativo cercando di dare alcuni suggerimenti per la sua compilazione. Come già detto il kernel ha il compito di amministrare le risorse presenti sul nostro sistema, rendendole disponibili ai processi. In questo modo si evita di far accedere i programmi direttamente a memoria, CPU, ecc. Ciò porta a una migliore organizzazione, stabilità, e maggiori performance.

Esistono fondamentalmente due tipi di kernel: "monolitico" e "microkernel". Le differenze tra i due tipi sono enormi e spesso sono nate anche delle forme ibride. Senza soffermarci troppo su questioni tecniche che potrebbero risultare noiose, ci sembra doveroso sottolineare le differenze tra i due e quali sono state le motivazioni che hanno portato Linus a preferirne uno tra questi modelli.

Un kernel monolitico ha il vantaggio essere strutturato in un unico file, risultando quindi piu' veloce, semplice e lineare. Un microkernel invece ha una struttura piu' complessa e articolata: è basato su un unico nucleo che si occupa soltanto di gestire il passaggio di messaggi alle altri componenti che sono staccate dal kernel. Queste ultime inoltre girano in "userspace". In questo modo ogni parte del kernel è completamente distinta dalle altre che vengono coordinate mediante lo scambio di messaggi. Questo tipo di kernel deve preoccuparsi solo di gestire questi messaggi e di amministrare le risorse delegando quest'ultimo compito a dei processi esterni (server). Ciò permette di aver un kernel molto snello a scapito (seppur di poco) delle prestazioni.

Linux è basato sulla prima tipologia. Ciò fu oggetto di discussione tra Linus Torvalds e Andrew S. Tanenbaum, creatore del sistema operativo MINIX, dal quale Torvalds prese spunto per Linux. Nonostante Linux si ispirasse a MINIX che era un microkernel, Linus decise di optare per un kernel monolitico. La tesi di Tanenbaum era fondata soprattutto sul fatto che quella dei kernel monolitici era una tecnologia ormai obsoleta e anche Linux rischiava di diventare obsoleto già prima di nascere. A queste accuse Torvalds rispose che Linux già per il fatto stesso di esistere era vincente: Hurd, il kernel (microkernel) su cui si doveva basare il sistema operativo GNU di Richard Stallman, non era ancora maturo e Linux poteva sopperire a questa mancanza. Tanenbaum (che era professore di Sistemi Operativi) concluse una sua email dicendogli: «I still maintain the point that designing a monolithic kernel in 1991 is a fundamental error. Be thankful you are not my student. You would not get a high grade for such a design» («Rimango ancora dell'idea che progettare un kernel monolitico nel 1991 è un errore di strutturazione. Sei fortunato che non sei un mio studente. Non prenderesti un buon voto per un tale progetto») Chissà se si è ricreduto di tale affermazione...

Tuttavia la complessità alla quale sta arrivando Linux ha di nuovo aperto il dibattito sull'opportunità di rivedere la struttura stessa del kernel. La posizione di Torvalds a proposito però non è cambiata. Recentemente hanno fatto discutere alcune sue affermazioni nella sua biografia "Just for Fun: The Story of an Accidental Revolutionary" nella quale descrive il kernel MACH (microkernel creato dalla Carnegie Mellon University nel 1985 sul quale è basato MacOS X) come pieno di errori di progettazione e spazzatura. Il kernel Linux si differenzia dalla maggior parte dei kernel monolitici per una interessante caratteristica: pur essendo basato su un nucleo compatto ha la possibilità di "collegare" a questo nucleo dei moduli, in modo da alleggerire il kernel e aggiungere alcuni supporti solo se ce ne è l'effettiva necessità. Bisogna prestare attenzione che questo tipo di approccio si differenzia notevolmente da quello dei microkernel (dove i moduli comunicano per mezzo di messaggi, e non vengono "agganciati" al nucleo principale facendone parte), ma che permette di alleggerire notevolmente il kernel.

Per ricompilare il kernel nelle distribuzioni RedHatLike abbiamo bisogno di due pacchetti RPM:

- kernel-headers (da tenere anche quando il kernel è ricompilato)
- kernel-sources (è possibile disinstallarlo dopo la compilazione – anche perché occupa un centinaio di MB)

Ora bisogna scaricare le eventuali patch di aggiornamento. Le patch sono sequenziali, dipendenti cioè dalla versione precedente. Questo vuol dire che se avete già il kernel 2.4.0 per passare al kernel 2.4.6 dovrete scaricare 6 patch e installarle. Per **installare** una patch scrivete:

```
lnxbox1:/usr/src# gzip -cd patchXX.gz | patch -p0
```

Dove al posto di XX vi è la versione della patch. I sorgenti sono anche disponibili come **bz2**, che offre un miglior grado di compressione, ma che non ha ancora soppiantato il gz. Se tutto è andato bene ci ritroveremo i sorgenti del nostro kernel nella directory `/usr/src/linux`.

Entriamo in questa directory:

```
lnxbox1:~# cd /usr/src/linux
```

e diamo:

```
lnxbox1:/usr/src/linux# make mrproper
```

In questo modo elimineremo tutti i **collegamenti** vecchi e dipendenze con la precedente versione. Questa operazione elimina anche il file `.config` che contiene la configurazione per il kernel.

A questo punto dobbiamo scegliere le opzioni da includere nel nuovo kernel. Da utente **root** scriviamo quindi:

```
lnxbox1:/usr/src/linux# make config
```

In questo modo ci verrà chiesto interattivamente cosa includere e cosa no. Esistono tuttavia dei modi più **amichevoli** per la configurazione del kernel. Al posto di `make config` possiamo scrivere:

```
lnxbox1:/usr/src/linux# make menuconfig
```

Per caricare il menu di configurazione testuale. Oppure:

```
lnxbox1:/usr/src/linux# make xconfig
```

Per caricare il menu di configurazione per X.

A questo punto si tratterà di scegliere quali **opzioni del kernel** fanno al caso nostro e quali no. Compilare il kernel è una delle operazioni più **delicate** nella quale potremo imbatterci, perché richiede una buona conoscenza dell'hardware installato sul proprio sistema e anche un minimo errore potrebbe costarci l'impossibilità di poter riavviare Linux.

Putroppo non sarebbe neanche possibile analizzare nei dettagli tutte le opzioni presenti nel kernel. Daremo però dei suggerimenti utili su quali sono le opzioni fondamentali da abilitare e quali possono essere gli errori più comuni. Se si è in dubbio se abilitare o meno un'opzione può tornare sempre di grande utilità l'help in linea. Generalizzando nell'incertezza la scelta meno rischiosa è "abbondare" nelle opzioni.

È buona norma accludere nel kernel solo il supporto per i componenti hardware **fondamentali** (come HD, filesystem ext2, supporto per i multiprocessore, ecc.) mentre compilare come moduli (contraddistinti da una M) le parti non indispensabili per il boot (come supporto a schede audio, schede di rete, filesystem aggiuntivi come FAT, ecc). Analizziamo ora i menu di configurazione.

Code maturity level options

È bene abilitare questa opzione ("Prompt for development and/or incomplete code/drivers") da includere staticamente (built-in, non come modulo quindi).

Loadable module support

È bene abilitare le opzioni in questo menu in modo da poter usare i moduli.

Processor type and features

In questo menu possiamo scegliere l'ottimizzazione per il nostro processore o abilitare il supporto per il

multiprocessore o l'MTRR.

General setup

Da questo menu sono assolutamente da abilitare il supporto per la rete (Networking support), i supporti per le schede PCI, quello per i binari del kernel in formato ELF (Kernel support for ELF binaries) (ma anche quelli a.out e MISC per una maggiore compatibilità con i vecchi formati). Se volete potete abilitare da qui il supporto per APM (il risparmio energetico). Sono anche da abilitare "Sysctl support" e "System V IPC"

Memory Technology Devices (MTD)

Le opzioni in questo menu possono essere disabilitate nella stragrande maggioranza dei casi.

Parallel port support

Da qui possiamo includere il supporto per la porta parallela (anche compilandola come modulo).

Plug and Play configuration

Da questo menu possiamo abilitare il supporto al Plug And Play (anche come modulo)

Block devices

Da qui abilitiamo il supporto per il floppy "Normal PC floppy disk support" (meglio se built-in) e "Loopback device support".

Multi-device support (RAID and LVM)

Se avete un controller RAID o volete usare il Logical Volume Manager potete abilitare le opzioni in questo menu. Ciò potrebbe richiedere però il caricamento di alcuni demoni fatti per gestire queste feature.

Networking options

Da qui abiliterete le opzioni per il networking come il TCP/IP, supporto per firewall, masquerading, IPV6, QoS, IPX. Una delle novità del kernel 2.4 è il supporto per il khttpd, l'accelerazione per il server http (abilitatela solo se avete installato un server http).

Telephony Support

Anche da questo menu quasi sicuramente non dovrete abilitare nulla

ATA/IDE/MFM/RLL support

Abilitate (come built-in) "ATA/IDE/MFM/RLL support" e entrate nel menu sottostante.

IDE, ATA and ATAPI Block devices

Da qui abilitate (come built-in):

"Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support"

"Include IDE/ATA-2 DISK support"

"Include IDE/ATAPI CDROM support"

Abilitate anche i supporti per il DMA se volete usarlo e l'opzione relativa al vostro chip presente sulla scheda madre.

Se avete un masterizzatore IDE dovrete abilitare anche "SCSI emulation support" e disabilitare l'opzione relativa al CDROM ("Include IDE/ATAPI CDROM support").

SCSI support

Da qui potete abilitare il supporto per periferiche SCSI (comprese quelle emulate). Se avete un controller SCSI dovrete abilitare l'opzione relativa al vostro modello dal sottomenu "SCSI low-level drivers".

IEEE 1394 (FireWire) support

A meno che voi non abbiate interfacce FireWire potrete anche ignorare questo menu (disabilitate eventuali opzioni abilitate).

I2O device support

Nella stragrande maggioranza dei casi potete ignorare anche questo menu

Network device support

Da qui potete abilitare il supporto per una scheda di rete, PPP (necessario per collegarsi a Internet col vostro provider), e per altri tipi di connessione (PLIP o SLIP). Potete abilitare le opzioni che fanno al caso vostro

come modulo.

Amateur Radio support

Da questo menu è possibile abilitare il supporto per il Packet Radio, che potrete tranquillamente ignorare.

IrDA (infrared) support

Se avete una porta infrarossi potrete abilitarla da questo menu. Da un sottomenu potrete anche scegliere il driver.

ISDN subsystem

Da questo menu potrete abilitare il supporto per le schede ISDN con i relativi driver.

Old CD-ROM drivers (not SCSI, not IDE)

A meno che non abbiate un lettore CD obsoleto con interfaccia non standard, ignorate questo menu.

Input core support

Da questo menu potrete abilitare il supporto per le interfacce USB come mouse, tastiere, joystick.

Character devices

Da qui potrete abilitare il supporto alle porte seriali, alle stampanti parallele, mouse PS/2, joystick e all'orologio di sistema. Da qui potrete anche abilitare il supporto per l'AGP e per il DRI in modo da aumentare le prestazioni della vostra scheda video sotto X.

Multimedia devices

In questo menu troverete il supporto per periferiche di acquisizione video oppure schede radio.

File systems

Da questo menu dovrete abilitare come built-in il filesystem dove è presente la partizione di root del vostro sistema / (ext2 o reiserfs) e potrete abilitare come modulo i filesystem non indispensabili dei quali necessitate il supporto. Da qui potrete abilitare anche il supporto per il devfs, anche se ve lo sconsiglio caldamente per ora essendo ancora in fase sperimentale.

Abilitate invece il "/proc file system support". È bene ribadire che il **filesystem** della vostra partizione Linux dovrebbe non essere compilato come modulo, pena l'impossibilità di fare il boot. Da qui potrete anche scegliere la codifica dei caratteri per la vostra nazione ("Native Language Support"). Per l'Italia la scelta cade su ISO-8859-1.

Console drivers

Da qui potrete il supporto per la vostra scheda VGA e per il Frame-buffer dall'opzione "Frame-buffer support".

Sound

Da questo menu avrete la possibilità di abilitare il supporto alla vostra scheda audio e per l'Open Sound System.

USB support

Tramite questo menu avrete la possibilità di attivare il supporto per l'interfacce USB. C'è anche da dire che il supporto USB sotto Linux è ancora in fase sperimentale, e spesso non c'è modo di far andare molte di esse.

Bluetooth support

Questo è una delle novità del kernel 2.4.6. Permette il supporto per le periferiche wireless che utilizzano questo protocollo. Potete evitare di abilitare anche questa opzione.

Kernel hacking

Questo menu permette di abilitare un'opzione per il debug del kernel. Normalmente non dovrebbe essere acclusa.

A questo punto potete tranquillamente uscire e salvare.

Tra gli **errori** che potreste commettere il più comune è senza dubbio quello di dimenticarvi di abilitare il supporto per qualche parte del sistema essenziale, come il supporto per i dischi IDE (o SCSI se usate

quelli), oppure il supporto al filesystem ext2. È importante anche che il supporto per questi sia messo come built-in, non come modulo, pena un kernel panic dopo il boot. Un'altro dei consigli è quello di **provare** sempre la propria versione del kernel prima di sostituirla. Potete configurare **LILO** in modo da avere sempre una versione di emergenza da caricare. Verrà descritta più avanti questa soluzione.

A questo punto bisogna dare i seguenti comandi per compilare sia i moduli che il kernel.

```
lnxbox1:/usr/src/linux# make dep && make bzImage
lnxbox1:/usr/src/linux# make modules && make modules_install
```

L'operatore && permette di eseguire dei comandi in serie avviando il successivo solo se il precedente è andato a buon fine. A questo punto ci dovremmo ritrovare l'immagine del nuovo kernel nella directory /usr/src/linux/arch/i386/boot sotto il nome di bzImage. Copiamo l'immagine nella directory /boot e rinominiamolo per esempio in vmlinuz-new.

```
lnxbox1:/usr/src/linux#
cp /usr/src/linux/arch/i386/boot/bzImage /boot/vmlinuz-new
```

Copiamo anche il file /usr/src/linux/System.map nella directory /boot

```
lnxbox1:/usr/src/linux# cp /usr/src/linux/System.map /boot
```

A questo punto dovremmo essere in grado di **installare** il nuovo kernel. Per far questo dobbiamo modificare il file /etc/lilo.conf. Se per esempio vogliamo lasciare la possibilità di caricare il vecchio kernel (caldamente consigliato qualora qualcosa andasse storto), possiamo rinominare il vecchio kernel presente in /boot (di solito chiamato **vmlinuz**) in vmlinuz-old, aggiungere una voce al LILO e sostituire il kernel. Per esempio cercate nel file /etc/lilo.conf la voce:

```
image=/boot/vmlinuz
label=Linux
read-only
root=/dev/hda5
```

l'ultima riga cambia a seconda di dove risiede la vostra partizione con Linux. Questa parte del file di configurazione di LILO assegna l'immagine del kernel presente in /boot/vmlinuz assegnandole il nome di Linux. Possiamo rinominare il vecchio kernel in vmlinuz-old, copiare il nuovo kernel in /boot e dandogli il nome di vmlinuz, lasciare invariata la voce precedente e aggiungere sotto una sezione del tipo:

```
image=/boot/vmlinuz-old
label=old
read-only
root=/dev/hda5
```

a questo punto scriviamo:

```
lnxbox1:~# lilo
```

Per rendere **effettive** le modifiche. Se qualcosa fosse andato storto in questo modo abbiamo sempre la possibilità di avviare Linux per correggere gli errori usando il vecchio kernel che richiameremo scrivendo old al prompt di LILO durante il boot.

Compilare il **kernel** non è mai una operazione facile. Una degli ostacoli più grandi è capire cosa abilitare e cosa no. Se proprio non sapete se un'opzione fa o meno al caso vostro il mio consiglio è di consultare l'help in linea, per farsi un'idea a proposito. Se poi i dubbi dovessero rimanere abilitate nel frattempo l'opzione, per poi tornarvi su successivamente e affinare la scelta, quando vi ritroverete un nuovo kernel compilato e funzionante.