`depmod` 220 `insmod` 219 `lsmod` 219 `MAKEDEV` 225 `make-kpkg` 194 `mkinitrd` 230 `mknod` 225 `modinfo` 222 `modprobe` 220 `rmmod` 219 `udevd` 225

Il kernel è il nocciolo del sistema operativo: i programmi utilizzano le funzioni fornite dal kernel e in questa maniera sono sollevati dall'agire direttamente con la CPU. Il kernel Linux è costituito da un file principale, il cui nome può essere '`vmlinuz`', oppure '`zImage`', '`bzImage`' e altri ancora, ma può comprendere anche moduli aggiuntivi per la gestione di componenti hardware specifici che devono poter essere attivati e disattivati durante il funzionamento del sistema.

Quando si fa riferimento a un kernel in cui tutte le funzionalità che servono sono incluse nel file principale, si parla di kernel monolitico, mentre quando parte di queste sono demandate a moduli esterni, si parla di kernel modulare. Il kernel monolitico ha il vantaggio di avere tutto in un file, ma nello stesso modo è rigido e non permette di liberare risorse quando le unità periferiche gestite non servono. Il kernel modulare ha il vantaggio di poter disattivare e riattivare i moduli a seconda delle esigenze, in particolare quando moduli distinti gestiscono in modo diverso lo stesso tipo di unità periferica. Tuttavia, a causa della frammentazione in molti file, l'uso dei moduli può essere fonte di errori.

In generale, chi fa per la prima volta l'esperienza di compilarsi un kernel personalizzato farebbe bene a cominciare incorporando nel file principale tutta la gestione delle memorie di massa, in modo da non dipendere dai moduli per l'innesto del file system principale.

8.1 Ricompilazione del kernel

Le distribuzioni GNU/Linux tendono a fornire agli utilizzatori un kernel modulare per usi generali. Anche se questo tipo di kernel si adatta sicuramente alla maggior parte delle configurazioni, ci sono situazioni particolari dove è preferibile costruire un proprio kernel, monolitico o modulare che sia. Per poter comprendere il procedimento di compilazione del kernel Linux, occorre sapere come si compila e si installa un programma tipico distribuito in forma sorgente, come descritto nella sezione 7.1.

Per poter procedere alla compilazione del kernel è necessario avere installato gli strumenti di sviluppo software, cioè il compilatore e i sorgenti del kernel. In particolare, i sorgenti del kernel possono anche essere reperiti presso vari siti, ma principalmente da *http://www.kernel.org* .

Se i sorgenti sono stati installati come parte di una distribuzione, questi potrebbero trovarsi da qualche parte a partire da '`/usr/src/`', ma la loro compilazione non richiede una collocazione particolare nel file system, tanto che anche un utente comune, senza privilegi, può farlo, collocando i sorgenti nell'ambito della propria directory personale. In particolare, la distribuzione GNU/Linux Debian si limita a piazzare un archivio (un file compresso) nella directory '`/usr/src/`', lasciando all'utente il compito di estrarlo dove meglio crede.

Si parte con l'estrazione dell'archivio che contiene i sorgenti; naturalmente l'utente deve estrarre l'archivio in una posizione in cui gli sia concesso di farlo in base ai permessi che lo riguardano. Va tenuto presente che i sorgenti possono essere compilati da un utente comune, anche se poi questo potrebbe non avere la facoltà di installare effettivamente il kernel prodotto in questo modo.

```
$ tar xjvf linux-versione.tar.bz2 [Invio]
```

Una volta estratti i sorgenti del kernel, si può passare alla configurazione che precede la compilazione. Per questo, ci si posiziona nella directory dei sorgenti; quindi, dopo aver letto il file '`README`', si può procedere.

```
$ cd linux-versione [Invio]
```

La directory corrente deve essere quella a partire dalla quale si diramano i sorgenti del kernel.

```
$ make mrproper [Invio]
```

Il comando '**`make mrproper`**' serve a eliminare file e collegamenti vecchi che potrebbero interferire con una nuova compilazione. In particolare, **elimina la configurazione data in precedenza** alla compilazione del kernel.

```
$ make defconfig [Invio]
```

La prima volta che si procede a definire i contenuti di un proprio kernel, conviene partire da una configurazione predefinita, ottenibile con il comando '**`make defconfig`**', appena mostrato. In pratica si ottiene un file '`.config`' con la selezione delle voci ritenute fondamentali. Successivamente si passa alla fase di configurazione:


```
$ make menuconfig [Invio]
```

Questa è l'operazione più delicata attraverso la quale si definiscono le caratteristiche e i componenti del kernel che si vuole ottenere. Ogni volta che la si esegue, viene riutilizzato il file '`.config`' contenente la configurazione impostata precedentemente, mentre alla fine la nuova configurazione viene salvata nello stesso file. Di conseguenza, ripetendo il procedimento '**`make config`**', le scelte predefinite corrispondono a quelle effettuate precedentemente.[1]

Il comando '**`make mrproper`**' elimina il file '`.config`', quindi si deve fare attenzione a non eseguire tale comando se non è questa l'intenzione.

Figura 8.1. Il menù principale della configurazione del kernel attraverso il comando '**`make menuconfig`**'.

```
.---------- Linux Kernel Configuration ----------.
|  Arrow keys navigate the menu.  <Enter> selects submenus |
|  --->. Highlighted letters are hotkeys.  Pressing <Y>    |
|  includes, <N> excludes, <M> modularizes features.       |
|  Press <Esc><Esc> to exit, <?> for Help, </> for Search. |
|  Legend: [*] built-in  [ ] excluded  <M> module  < >     |
| .------------------------------------------------------. |
| |    General setup  --->                               | |
| |[*] Enable loadable module support  --->              | |
| |-*- Enable the block layer  --->                      | |
| |    Processor type and features  --->                 | |
| |    Power management options (ACPI, APM)  --->        | |
| |    Bus options (PCI, PCMCIA, EISA, MCA, ISA)  --->   | |
| |    Executable file formats  --->                     | |
| |    Networking support --->                           | |
| `v(+)------------------------------------------------' |
|-------------------------------------------------------|
|             <Select>    < Exit >    < Help >          |
`-------------------------------------------------------'
```

Dopo la definizione della configurazione, si può passare alla compilazione del kernel relativo, utilizzando la sequenza di comandi seguente:

```
$ make clean && make [Invio]
```

Si tratta di due operazioni che non richiedono alcun tipo di interazione con l'utente. Al termine della compilazione, se questa ha avuto successo, il nuovo kernel si trova nella directory '`arch/i386/boot/`' con il nome '`bzImage`'[2] (questo vale naturalmente nel caso si utilizzi l'architettura x86).

Naturalmente, per fare in modo che il kernel possa essere utilizzato, questo va collocato dove il sistema che si occupa del suo avvio può trovarlo (capitolo 6). Di solito lo si copia nella directory radice o in '`/boot/`', dandogli il nome '`vmlinuz`' (come di consueto), sistemando poi ciò che serve per il sistema di avvio che si utilizza.

Generalmente, dal momento che si possono utilizzare più kernel alternativi, selezionandoli in fase di avvio, di solito il nome del file è completo della versione del kernel stesso: '`vmlinuz-`***versione***'. Inoltre, si conserva una copia del file '`System.map`', mettendola nella stessa directory in cui si trova il kernel, con un nome del tipo '`System.map-`***versione***'; infine si fa la stessa cosa con il file '`.config`', a cui si dà normalmente il nome '`config-`***versione***'.

Se, in base alla configurazione, ciò che si vuole è ottenere soltanto un kernel monolitico (senza moduli), il procedimento si è concluso; diversamente occorre procedere con la compilazione dei moduli. Ma questi moduli, per poter essere gestiti correttamente, necessitano di programmi di servizio che si occupano della loro attivazione e disattivazione, i quali, a loro volta, potrebbero richiedere un aggiornamento in corrispondenza dell'uso di un nuovo kernel.

I programmi di servizio di gestione dei moduli si trovano normalmente in archivi il cui nome è organizzato in modo simile a quello

dei sorgenti del kernel: '`modules-`***versione***`.tar.gz`'. La struttura della versione rappresentata dai numeri ***versione*** rispecchia lo stesso meccanismo utilizzato per i sorgenti del kernel, però non ne vengono prodotte altrettante versioni, pertanto si deve badare a utilizzare la versione più vicina a quella del kernel che si utilizza. Questo archivio si trova normalmente nella stessa directory del sito dal quale si ottengono i sorgenti del kernel. Anche i programmi contenuti nell'archivio '`modules-`***versione***`.tar.gz`' sono in forma sorgente e prima di poterli utilizzare devono essere compilati e installati.

Se si sta ricompilando il kernel attraverso i sorgenti della distribuzione GNU/Linux che si utilizza, è ragionevole supporre che questi programmi di gestione dei moduli siano già stati installati correttamente.

Per ottenere un kernel modulare, dopo la preparazione del file principale del kernel attraverso il procedimento già descritto, si deve passare alla compilazione dei moduli:

```
$ make modules [Invio]
```

Quindi si installano, ma per questo servono i privilegi dell'utente '**`root`**':

```
# make modules_install [Invio]
```

Quello che si ottiene sono dei file oggetto, i cui nomi hanno un'estensione '`.ko`' (*Kernel object*), raggruppati ordinatamente all'interno di directory discendenti da '`/lib/modules/`***versione***`/`', dove ***versione*** rappresenta il numero della versione dei sorgenti del kernel. La posizione di questi file non deve essere cambiata.

Si osservi che la compilazione dei moduli deve essere ripetuta ogni volta che si ricompila la parte principale del kernel; in altre parole, i moduli non sono compatibili con un kernel differente da quello per il quale sono stati realizzati specificatamente, anche se si tratta della stessa versione.

8.2 Compilazione del kernel in una distribuzione GNU/Linux Debian

Teoricamente, la distribuzione GNU/Linux Debian consente di compilare il kernel e i moduli secondo il procedimento standard; tuttavia è disponibile uno strumento per facilitare la compilazione del kernel, passando per la creazione di un pacchetto Debian vero e proprio, che poi può essere installato secondo la procedura comune della distribuzione stessa. Il pacchetto in questione è denominato '**`kernel-package`**' e per questo scopo può essere usato direttamente senza bisogno di alcuna configurazione. È sufficiente procedere nel modo seguente:

1. `cd` ***directory_iniziale_dei_sorgenti***
   ci si sposta nella directory iniziale dei sorgenti del kernel;
2. `make {config|menuconfig|xconfig|…}`
   si procede con la configurazione del kernel che si vuole ottenere;
3. `make-kpkg clean`
   ci si prepara alla compilazione;
4. `make-kpkg --revision=`***versione*** `kernel_image`
   si esegue la compilazione generando l'archivio Debian corrispondente, nella directory precedente (la directory genitrice), completo di moduli se previsti.

Questo ultimo passaggio richiede però i privilegi dell'utente '**`root`**' per arrivare alla creazione del pacchetto Debian.

L'esempio seguente si riferisce alla compilazione di un kernel 2.6.21.6 (compresi i moduli eventuali) collocato nella directory '`~/linux-2.6.21.6/`'.

Questa collocazione è volutamente differente da quella standard per la distribuzione GNU/Linux Debian, la quale invece dovrebbe essere '`~/kernel-source-2.6.21.6/`', proprio per mostrare che ciò non influisce in questo contesto.

```
$ cd ~/linux-2.6.21.6 [Invio]

$ make-kpkg clean [Invio]

$ su root [Invio]

# make-kpkg --revision=custom.1.0 kernel_image [Invio]
```

Si può osservare che la versione è stata definita dalla stringa '**`custom.1.0`**'. Questo è ciò che viene suggerito nella documentazione originale. In particolare, il numero «1.0» va incrementato ogni volta che si predispone una versione successiva.

Al termine si ottiene l'archivio '`kernel-image-2.6.21.6_custom.1.0_i386.deb`', collocato nella directory precedente a quella dei sorgenti da cui è stato ottenuto; per installarlo basta procedere come segue, usando naturalmente i privilegi dell'utente '**`root`**':

```
# dpkg -i ../kernel-image-2.6.21.6_custom.1.0_i386.deb [Invio]
```

8.3 Elementi della configurazione

Gli elementi richiesti per la configurazione del kernel prima della sua compilazione, dipendono molto dalla versione che si possiede. In particolare, può capitare che alcune voci vengano spostate da una versione all'altra del kernel.

Le varie opzioni sono raggruppate in alcuni gruppi principali, i quali dovrebbero guidare intuitivamente nella configurazione prima della compilazione del kernel:

- *General setup*, caratteristiche generali, sia fisiche, sia logiche del sistema;
- *Enable loadable module support*, gestione dei moduli del kernel;
- *Enable the block layer*, gestione dei dispositivi a blocchi che può essere omessa in certi kernel particolari (ma nell'architettura x86 non può mancare);
- *Processor type and features*, caratteristiche del microprocessore o dei microprocessori;
- *Power management options (ACPI, APM)*, controllo del sistema di alimentazione e del risparmio energetico;
- *Bus options (PCI, PCMCIA, EISA, MCA, ISA)*, caratteristiche di vari tipi di bus;
- *Executable file formats*, formati dei file eseguibili;
- *Networking*, protocolli di rete;
- *Device Drivers*, controllo dei dispositivi di vario genere;
  - *Generic Driver Options*, opzioni generali riferite alla gestione dei dispositivi;
  - *Connector - unified userspace <:-> kernelspace linker*
  - *Memory Technology Devices (MTD)*, gestione di memorie MTD, ovvero memoria speciale che ha la proprietà di non essere volatile come la RAM comune;
  - *Parallel port support*, gestione delle porte parallele;
  - *Plug and Play support*, gestione del Plug & Play;

– *Block devices*, gestione dei dispositivi a blocchi;
– *Misc devices*, gestione di dispositivi vari;
– *ATA/ATAPI/MFM/RLL support*, gestione di dischi PATA/ATAPI e simili;
– *SCSI device support*, gestione di unità SCSI;
– *Serial ATA and Paralel ATA drivers*, gestione di dischi SATA e gestione alternativa di dischi PATA;
– *Multi-device support (RAID and LVM)*, gestione software di unità multiple di memorizzazione, come nel caso dei dischi RAID;
– *Fusion MPT device support*, gestione di dispositivi Fusion MPT;
– *IEEE 1394 (FireWire) support*, gestione di un bus IEEE 1394, noto anche con il nome FireWire;
– *I2O device support*, gestione di dispositivi periferici speciali denominati «I2O»;
– *Macintosh device drivers*, gestione di dispositivi specifici per elaboratori Macintosh;
– *Network device support*, gestione di dispositivi di rete;
– *ISDN subsystem*, gestione di alcune schede speciali per la connessione a una rete ISDN (non è necessario utilizzare queste opzioni se si dispone di un «modem» ISDN esterno);
– *Telephony Support*, gestione di hardware speciale per la telefonia digitale su IP;
– *Input device support*, gestione di hardware per l'inserimento dati (tastiere, mouse ecc.);
– *Character devices*, gestione dei dispositivi a caratteri (terminali, porte seriali, porte parallele, mouse, ecc.);
– *I2C support*, gestione dei bus seriali I2C e SMBus;
– *SPI support*, gestione del protocollo SPI (*Serial peripheral interface*);
– *Dallas's 1-wire bus*, gestione del bus denominato come la voce stessa;
– *Power supply class support*, gestione dell'alimentatore;
– *Hardware Monitoring support*, gestione di componenti tipici delle schede madri per il controllo dello stato di funzionamento dell'hardware, soprattutto per quanto riguarda la temperatura;
– *Multifunction device drivers*, gestione di dispositivi multifunzionali;
– *Multimedia devices*, gestione di dispositivi multimediali;
– *Graphics support*, gestione della console;
– *Sound*, gestione dell'audio;
– *HID Devices*, gestione generale dei dispositivi di interazione (*Human interface device*);
– *USB support*, gestione del bus USB e delle unità periferiche relative;
– *MMC/SD cart support*, gestione del protocollo relativo ai bus MMC (*Multimedia card*);
– *LED devices*, gestione di unità che visualizzano certe condizioni attraverso luci led (non si tratta di quelli della tastiera);
– *InfiniBand support*, gestione di unità InfiniBand e dei protocolli di comunicazione relativi;
– *EDAC - error detection and reporting (RAS)*, gestione di funzionalità di individuazione di errori nell'hardware;
– *Real Time Clock*, gestione di funzionalità «RTC»;
– *DMA Engine support*, gestione avanzata della copia in memoria di dati, senza la richiesta di intervento da parte della CPU;
– *Auxiliary Display support*, gestione di unità di visualizzazione ausiliarie;

– *Virtualization*, gestione delle funzionalità di virtualizzazione hardware;
– *Userspace I/O*
– *Linux hypervisor example code*

• *File systems*, gestione di file system, con le problematiche relative;
• *Instrumentation support*
• *Kernel hacking*, configurazione particolare per chi vuole lavorare attivamente allo sviluppo del kernel;
• *Security options*, configurazione dei sistemi di sicurezza attuati dal kernel;
• *Cryptographic options*, funzionalità crittografiche;
• *Library routines*, librerie.

Nelle sezioni seguenti vengono descritti in parte solo alcuni di questi gruppi di configurazione, mostrando qualche esempio che comunque non può esaurire il problema.

8.3.1 General setup

Questa sezione raccoglie delle opzioni di importanza generale che non hanno trovato una collocazione specifica in un'altra posizione della procedura di configurazione.

```
[*] Prompt for development and/or incomplete code/drivers
    () Cross-compiler tool prefix (NEW)
      () Local version - append to kernel release
[ ] Automatically append version information to the version string
      Kernel compression mode (Gzip) --->
((none)) Default hostname (NEW)
[*] Support for paging of anonymous memory (swap)
[*] System V IPC
[*] POSIX Message Queues
[*] BSD Process Accounting
      [*] BSD Process Accounting version 3 file format
[ ] open by fhandle syscalls (NEW)
-*- Export task/process statistics through netlink (EXPERIMENTAL)
      -*- Enable per-task delay accounting (EXPERIMENTAL)
[ ] Enable extended accounting over taskstats (EXPERIMENTAL) (NEW)
[*] Auditing support
[ ] Enable system-call auditing support
      IRQ subsystem --->
      RCU Subsystem --->
<M> Kernel .config support
      [*] Enable access to .config through /proc/config.gz
(16) Kernel log buffer size (16 => 64KB, 17 => 128KB)
[*] Control Group support --->
-*- Namespaces support --->
[ ] Automatic process group scheduling (NEW)
[ ] Enable deprecated sysfs features to support old userspace tools (NEW)
[*] Kernel->user space relay support (formerly relayfs)
[*] Initial RAM filesystem and RAM disk (initramfs/initrd) support
      () Initramfs source file(s)
[*] Optimize for size
[ ] Configure standard kernel features (expert users) (NEW) --->
[ ] Embedded system
      Kernel Performance Events And Counters --->
[*] Disable heap randomization
      Choose SLAB allocator (SLAB) --->
[ ] Profiling support
[ ] Kprobes
[ ] Optimize trace point call sites (NEW)
      GCOV-based kernel profiling --->
```

8.3.2 Enable loadable module support

Questa sezione della procedura di configurazione permette di attivare il sistema di gestione dei moduli. I moduli sono blocchetti di kernel separati che possono essere attivati e disattivati durante il funzionamento del sistema. Solo alcune parti del kernel possono essere gestite in forma di modulo.

Se si intende creare un kernel modulare, è evidente la necessità di attivare questa gestione all'interno della parte principale del kernel stesso.

--- Enable loadable module support
[ ] Forced module loading
[*] Module unloading
[ ] Forced module unloading
[*] Module versioning support
[ ] Source checksum for all modules

8.3.3 Processor type and features

Questa sezione serve a definire il tipo di microprocessore utilizzato. In generale, se si utilizza un'architettura di tipo x86, la selezione del tipo di microprocessore '**386**' garantisce la creazione di un kernel compatibile nella maggior parte delle situazioni, a discapito però delle prestazioni.

Sempre nel caso di architettura di tipo x86, è possibile abilitare l'emulazione per il coprocessore matematico (i387), che in alcuni elaboratori molto vecchi non è incluso. Di solito, l'inclusione del codice di emulazione non crea problemi di conflitti, perché viene individuata automaticamente la presenza dell'hardware relativo e l'emulazione non viene attivata se non quando necessario. In tal modo, includendo questa funzionalità si genera un kernel più compatibile.

L'esempio che si vede sotto si riferisce a una compatibilità limitata a microprocessori i686 o superiori:

[*] Tickless System (Dynamic Ticks)
[*] High Resolution Timer Support
[*] Symmetric multi-processing support
-- Enable MPS table*
[*] Support for big SMP systems with more than 8 CPUs
[*] Support for extended (non-PC) x86 platforms
[ ] Intel MID platform support (NEW)
[ ] RDC R-321x SoC
[*] Support non-standard 32-bit SMP architectures
[*] NUMAQ (IBM/Sequent)
[*] Summit/EXA (IBM x440)
[*] Unisys ES7000 IA32 series
< > Eurobraille/Iris poweroff module (NEW)
[*] Single-depth WCHAN output
[*] Paravirtualized guest support --->
[*] Memtest
Processor family (Pentium-4/Celeron(P4-based)/Pentium-4 M/older Xeon) --->
[*] Generic x86 support
[*] HPET Timer Support
(8) Maximum number of CPUs
[*] SMT (Hyperthreading) scheduler support
[*] Multi-core scheduler support
[ ] Fine granularity task level IRQ time accounting (NEW)
Preemption Model (Preemptible Kernel (Low-Latency Desktop)) --->
[*] Reroute for broken boot IRQs
[*] Machine Check / overheating reporting
[*] Intel MCE features
[*] AMD MCE features
[ ] Support for old Pentium 5 / WinChip machine checks
<M> Machine check injector support
<M> Toshiba Laptop support
<M> Dell laptop support

[*] Enable X86 board specific fixups for reboot
<M> /dev/cpu/microcode - microcode support
[*] Intel microcode patch loading support
[*] AMD microcode patch loading support
<M> /dev/cpu//msr - Model-specific register support*
<M> /dev/cpu//cpuid - CPU information support*
High Memory Support (64GB) --->
-- PAE (Physical Address Extension) Support*
-- Numa Memory Allocation and Scheduler Support*
[ ] NUMA emulation (NEW)
(4) Maximum NUMA Nodes (as a power of 2)
Memory model (Discontiguous Memory) --->
[ ] Allow for memory compaction (NEW)
[*] Page migration
[*] Enable KSM for page merging
(32768) Low address space to protect from user allocation
[ ] Transparent Hugepage Support (NEW)
[ ] Enable cleancache driver to cache clean pages if tmem is present (NEW)
[*] Allocate 3rd-level pagetables from highmem
[*] Check for low memory corruption
[*] Set the default setting of memory_corruption_check
(64) Amount of low memory, in kilobytes, to reserve for the BIOS (NEW)
[ ] Math emulation
-- MTRR (Memory Type Range Register) support*
[*] MTRR cleanup support
(0) MTRR cleanup enable value (0-1)
(1) MTRR cleanup spare reg num (0-7)
[*] EFI runtime service support
[*] Enable seccomp to safely compute untrusted bytecode
[ ] Enable -fstack-protector buffer overflow detection (EXPERIMENTAL)
Timer frequency (1000 HZ) --->
[*] kexec system call
[ ] kernel crash dumps
[*] Build a relocatable kernel
(0x100000) Alignment value to which kernel should be aligned
-- Support for hot-pluggable CPUs*
[*] Compat VDSO support
[ ] Built-in kernel command line

8.3.4 Power management and ACPI options

Questa sezione permette di accedere alle funzionalità legate al controllo dell'alimentazione e al risparmio energetico. In generale si tratta di opzioni delicate che dipendono molto dalle caratteristiche fisiche dell'elaboratore.

[*] Suspend to RAM and standby
[ ] Hibernation (aka 'suspend to disk')
[ ] Run-time PM core functionality
[ ] Power Management Debug Support
[*] ACPI (Advanced Configuration and Power Interface) Support --->
[ ] SFI (Simple Firmware Interface) Support --->
<> APM (Advanced Power Management) BIOS support --->*
CPU Frequency scaling --->
-- CPU idle PM support*
[ ] Cpuidle Driver for Intel Processors (NEW)

L'esempio mostra l'attivazione complessiva delle funzionalità, le quali devono però essere specificate all'interno di altri menù. In generale, dovrebbe essere possibile attivare le funzioni che consentono di spegnere l'elaboratore via software, senza problemi particolari di incompatibilità, selezionando la voce *APM (Advanced Power Management) BIOS Support* e compilando la maschera relativa nel modo seguente:

--- APM (Advanced Power Management) BIOS support

[ ] Ignore USER SUSPEND
[ ] Enable PM at boot time
[*] Make CPU Idle calls when idle
[ ] Enable console blanking using APM
[ ] Allow interrupts during APM BIOS calls

Si osservi che l'attivazione della voce *Use real mode APM BIOS call to power off* può essere controproducente con alcune schede madri che dispongono del BIOS AMI.

8.3.5 Bus options

Questa sezione permette di selezionare la gestione di alcuni tipi di bus. Generalmente è necessario abilitare l'uso di bus ISA e PCI:

[*] PCI support
PCI access mode (Any) --->
[*] PCI Express support
<M> PCI Express Hotplug driver
[*] Root Port Advanced Error Reporting support
[ ] PCI Express ECRC settings control
<M> PCIe AER error injector support
-- PCI Express ASPM control*
[ ] Debug PCI Express ASPM (NEW)
[*] Message Signaled Interrupts (MSI and MSI-X)
< > PCI Stub driver
[*] Interrupts on hypertransport devices
[ ] PCI IOV support
[*] ISA support
[ ] EISA support
[ ] MCA support
<M> NatSemi SCx200 support
<M> NatSemi SCx200 27MHz High-Resolution Timer Support
<M> PCCard (PCMCIA/CardBus) support --->
<M> Support for PCI Hotplug --->
[ ] RapidIO support (NEW)

Se si utilizzano ancora le schede PCMCIA, è necessario attivare la gestione dei vari tipi di integrati che sono in grado di gestire un bus di questo tipo. Si accede a queste voci da *PCCARD (PCMCIA/CardBus) support*:

--- PCCard (PCMCIA/CardBus) support
<M> 16-bit PCMCIA support
[*] Load CIS updates from userspace (EXPERIMENTAL)
-- 32-bit CardBus support*
**** PC-card bridges ****
<M> CardBus yenta-compatible bridge support
<M> Cirrus PD6729 compatible bridge support
<M> i82092 compatible bridge support
<M> i82365 compatible bridge support
<M> Databook TCIC host bridge support

8.3.6 Executable file formats

Il kernel deve essere predisposto per il tipo di eseguibili che si vogliono usare. In generale, se possibile, conviene abilitare tutti i tipi disponibili:

[*] Kernel support for ELF binaries
[ ] Write ELF core dumps with partial segments
<M> Kernel support for a.out and ECOFF binaries
<> Kernel support for MISC binaries*

8.3.7 Networking

La configurazione delle funzionalità di rete è importante anche se il proprio elaboratore è isolato. Quando è attiva la gestione della rete, il kernel fornisce implicitamente le funzionalità di inoltro dei pacchetti, consentendo in pratica il funzionamento come router. Tuttavia, l'attivazione di ciò dipende dall'inclusione della gestione del file system '`/proc/`' (8.3.9) e dell'interfaccia '**`sysctl`**' (8.3.1). Inoltre, durante il funzionamento del sistema è necessario attivare

espressamente l'inoltro IPv4 attraverso un comando simile a quello seguente:

```
# echo '1' > /proc/sys/net/ipv4/ip_forward [ Invio ]
```

Esiste anche un comando analogo per IPv6, che però va usato soltanto quando serve veramente, perché con la funzionalità di inoltro attiva, il nodo di rete non prende in considerazione la configurazione automatica di Radvd (sezione 32.15.6):

```
# echo 1 > /proc/sys/net/ipv6/conf/interfaccia/forwarding [ Invio ]
```

--- Networking support
Networking options --->
[*] Amateur Radio support --->
<M> CAN bus subsystem support --->
<M> IrDA (infrared) subsystem support --->
<M> Bluetooth subsystem support --->
{M} RxRPC session sockets
[ ] RxRPC dynamic debugging
<M> RxRPC Kerberos security
-- Wireless --->*
<M> WiMAX Wireless Broadband support --->
<M> RF switch subsystem support --->
<M> Plan 9 Resource Sharing Support (9P2000) --->
< > CAIF support (NEW) --->
< > Ceph core library (EXPERIMENTAL) (NEW)
< > NFC subsystem support (EXPERIMENTAL) (NEW) --->

Le opzioni di funzionamento della rete sono raccolte in un menù a cui si accede dalla voce *Networking options*:

<> Packet socket*
<> Unix domain sockets*
<M> Transformation user configuration interface
[ ] Transformation sub policy support (EXPERIMENTAL)
[ ] Transformation migrate database (EXPERIMENTAL)
[ ] Transformation statistics (EXPERIMENTAL)
<M> PF_KEY sockets
[ ] PF_KEY MIGRATE (EXPERIMENTAL)
[*] TCP/IP networking
[*] IP: multicasting
[*] IP: advanced router
[ ] FIB TRIE statistics (NEW)
[*] IP: policy routing
[*] IP: equal cost multipath
[*] IP: verbose route monitoring
[ ] IP: kernel level autoconfiguration
<M> IP: tunneling
< > IP: GRE demultiplexer (NEW)
[*] IP: multicast routing
[ ] IP: multicast policy routing (NEW)
[*] IP: PIM-SM version 1 support
[*] IP: PIM-SM version 2 support
[ ] IP: ARP daemon support
[*] IP: TCP syncookie support
<M> IP: AH transformation
<M> IP: ESP transformation
<M> IP: IPComp transformation
<M> IP: IPsec transport mode
<M> IP: IPsec tunnel mode
<M> IP: IPsec BEET mode
{} Large Receive Offload (ipv4/tcp)*
<M> INET: socket monitoring interface
[*] TCP: advanced congestion control --->
[*] TCP: MD5 Signature Option support (RFC2385) (EXPERIMENTAL)
<M> The IPv6 protocol --->
[*] Security Marking
[ ] Timestamping in PHY devices (NEW)
[*] Network packet filtering framework (Netfilter) --->

```
<M> The DCCP Protocol (EXPERIMENTAL) --->
-M- The SCTP Protocol (EXPERIMENTAL) --->
< > The RDS Protocol (EXPERIMENTAL)
<M> The TIPC Protocol (EXPERIMENTAL) --->
<M> Asynchronous Transfer Mode (ATM)
      <M> Classical IP over ATM
[ ] Do NOT send ICMP if no neighbour
      <M> LAN Emulation (LANE) support
      <M> Multi-Protocol Over ATM (MPOA) support
      <M> RFC1483/2684 Bridged protocols
[ ] Per-VC IP filter kludge
< > Layer Two Tunneling Protocol (L2TP) (NEW) --->
<M> 802.1d Ethernet Bridging
      [*] IGMP/MLD snooping
[*] Distributed Switch Architecture support --->
<M> 802.1Q VLAN Support
[ ] GVRP (GARP VLAN Registration Protocol) support
<M> DECnet Support
      [*] DECnet: router support (EXPERIMENTAL)
<M> ANSI/IEEE 802.2 LLC type 2 Support
<M> The IPX protocol
      [*] IPX: Full internal IPX network
<M> Appletalk protocol support
      <M> Appletalk interfaces support
      <M> Apple/Farallon LocalTalk PC support
      <M> COPS LocalTalk PC support
      [*] Dayna firmware support
      [*] Tangent firmware support
      <M> Appletalk-IP driver support
      [*] IP to Appletalk-IP Encapsulation support
      [*] Appletalk-IP to IP Decapsulation support
<M> CCITT X.25 Packet Layer (EXPERIMENTAL)
<M> LAPB Data Link Driver (EXPERIMENTAL)
<M> Acorn Econet/AUN protocols (EXPERIMENTAL)
      [*] AUN over UDP
      [*] Native Econet
<M> WAN router
<M> Phonet protocols family
<M> IEEE Std 802.15.4 Low-Rate Wireless Personal Area Networks
support (EXPERIMENTAL)
[*] QoS and/or fair queueing --->
[ ] Data Center Bridging support
-*- DNS Resolver support
< > B.A.T.M.A.N. Advanced Meshing Protocol (NEW)
      Network testing --->
```

La gestione relativa a IPTables, diventa accessibile in un menù separato, dopo aver attivato la voce *Network packet filtering framework*:

```
--- Network packet filtering framework (Netfilter)
[ ] Network packet filtering debugging
      [*] Advanced netfilter configuration
      [*] Bridged IP/ARP packets filtering
      Core Netfilter Configuration --->
< > IP set support (NEW) --->
      <M> IP virtual server support --->
      IP: Netfilter Configuration --->
      IPv6: Netfilter Configuration --->
      DECnet: Netfilter Configuration --->
      <M> Ethernet Bridge tables (ebtables) support --->
```

Per accedere alla gestione di filtro IPv4, occorre selezionare la voce *IP: Netfilter Configuration*:

```
<M> IPv4 connection tracking support (required for NAT)
      [*] proc/sysctl compatibility with old connection tracking
<M> IP Userspace queueing via NETLINK (OBSOLETE)
<M> IP tables support (required for filtering/masq/NAT)
      <M> "ah" match support
      <M> "ecn" match support
      <M> "ttl" match support
      <M> Packet filtering
      <M> REJECT target support
      <M> LOG target support
      <M> ULOG target support
      <M> Full NAT
      <M> MASQUERADE target support
      <M> NETMAP target support
      <M> REDIRECT target support
      <M> Packet mangling
      <M> CLUSTERIP target support (EXPERIMENTAL)
      <M> ECN target support
      <M> "TTL" target support
      <M> raw table support (required for NOTRACK/TRACE)
<M> ARP tables support
      <M> ARP packet filtering
      <M> ARP payload mangling
```

Successivamente, dal menù *Device drivers*, *Network device support*, si procede all'inserzione del codice necessario alla gestione delle varie interfacce di rete utilizzate effettivamente.

8.3.8 Device drivers

Questa sezione raccoglie un menù di tipi di dispositivo, ognuno dei quali conduce poi a un sottomenù con le voci dettagliate dei dispositivi gestibili.

```
Generic Driver Options --->
<M> Connector - unified userspace <-> kernelspace linker --->
<M> Memory Technology Device (MTD) support --->
<M> Parallel port support --->
-*- Plug and Play support --->
[*] Block devices --->
-*- Misc devices --->
<M> ATA/ATAPI/MFM/RLL support (DEPRECATED) --->
      SCSI device support --->
<M> Serial ATA and Parallel ATA drivers --->
[*] Multiple devices driver support (RAID and LVM) --->
< > Generic Target Core Mod (TCM) and ConfigFS Infrastructure
(NEW) --->
[*] Fusion MPT device support --->
      IEEE 1394 (FireWire) support --->
<M> I2O device support --->
[*] Macintosh device drivers --->
-*- Network device support --->
[ ] ISDN support --->
<M> Telephony support --->
      Input device support --->
      Character devices --->
      {M} I2C support --->
[*] SPI support --->
      PPS support --->
      PTP clock support --->
-*- GPIO Support --->
<M> Dallas's 1-wire support --->
      {*} Power supply class support --->
      {M} Hardware Monitoring support --->
      {*} Generic Thermal sysfs driver --->
[*] Watchdog Timer Support --->
      Sonics Silicon Backplane --->
      Broadcom specific AMBA --->
[*] Multifunction device drivers (NEW) --->
[*] Voltage and Current Regulator Support --->
<M> Multimedia support --->
```

Graphics support --->
<M> Sound card support --->
[*] HID Devices --->
[*] USB support --->
{M} Ultra Wideband devices (EXPERIMENTAL) --->
<M> MMC/SD/SDIO card support --->
<M> Sony MemoryStick card support (EXPERIMENTAL) --->
-- LED Support --->*
[*] Accessibility support --->
<M> InfiniBand support --->
[*] EDAC (Error Detection And Correction) reporting --->
[ ] Real Time Clock (NEW) --->
[*] DMA Engine support --->
[*] Auxiliary Display support --->
{M} Userspace I/O drivers --->
Virtio drivers --->
[ ] Staging drivers --->
[*] X86 Platform Specific Device Drivers --->
[*] IOMMU Hardware Support (NEW) --->
[ ] Virtualization drivers (NEW) --->

8.3.8.1 Device drivers, parallel port support

Se esiste ancora, la gestione della porta parallela non riguarda solo la stampa, dal momento che consentirebbe anche l'uso di altri tipi di unità periferiche. In questo gruppo di opzioni è possibile abilitare l'uso delle porte parallele, stabilendo eventualmente il grado di compatibilità di queste.

--- Parallel port support
<M> PC-style hardware
<M> Multi-IO cards (parallel and serial)
[*] Use FIFO/DMA if available (EXPERIMENTAL)
[ ] SuperIO chipset support (EXPERIMENTAL)
<M> Support for PCMCIA management for PC-style ports
<M> AX88796 Parallel Port
[*] IEEE 1284 transfer modes

8.3.8.2 Device drivers, plug and play support

La gestione del Plug & Play permette al kernel di configurare automaticamente alcuni dispositivi che aderiscono a queste specifiche.

--- Plug and Play support
[*] PNP debugging messages
**** Protocols ****
[*] ISA Plug and Play support
[*] Plug and Play BIOS support (EXPERIMENTAL)
[*] Plug and Play BIOS /proc interface

8.3.8.3 Device drivers, block devices

Un dispositivo a blocchi è quello che utilizza una comunicazione a blocchi di byte di dimensione fissa, contrapponendosi al dispositivo a caratteri con cui la comunicazione avviene byte per byte. Il dispositivo a blocchi tipico è costituito da un'unità di memorizzazione di massa.

Merita attenzione particolare anche il dispositivo definito *loopback*,[3] il quale rappresenta in pratica un file contenente l'immagine di una memoria di massa che viene letta come se fosse un'unità di memorizzazione reale. Questa possibilità, tra le altre cose, consente di gestire direttamente i file che contengono la riproduzione esatta di dischi o altre memorie di massa.

Infine, è qui che si può abilitare e configurare la gestione dei dischi RAM, ovvero di dischi che vengono rappresentati nella memoria RAM.

--- Block devices
<> Normal floppy disk support*
< > XT hard disk support
< > Parallel port IDE device support
<M> Compaq SMART2 support
<M> Compaq Smart Array 5xxx support
[*] SCSI tape drive support for Smart Array 5xxx
<M> Mylex DAC960/DAC1100 PCI RAID Controller support
<M> Micro Memory MM5415 Battery Backed RAM support (EXPERIMENTAL)
<M> Loopback device support
(8) Number of loop devices to pre-create at init time (NEW)
<M> Cryptoloop Support
<M> DRBD Distributed Replicated Block Device support
[ ] DRBD fault injection
<M> Network block device support
<M> OSD object-as-blkdev support
<M> Promise SATA SX8 support
< > Low Performance USB Block driver
<> RAM block device support*
(16) Default number of RAM disks
(24576) Default RAM disk size (kbytes)
[ ] Support XIP filesystems on RAM block device
<M> Packet writing on CD/DVD media
(8) Free buffers for data gathering
[ ] Enable write caching (EXPERIMENTAL)
<M> ATA over Ethernet support
<M> Virtio block driver (EXPERIMENTAL)
[ ] Very old hard disk (MFM/RLL/IDE) driver
< > Rados block device (RBD) (NEW)

8.3.8.4 Device drivers, SCSI support

Questa sezione riguarda la gestione del kernel delle unità SCSI. Le interfacce SCSI non hanno uno standard comune come avviene nel caso di quelle ATA e derivate, per cui è indispensabile includere il codice specifico per tutte le interfacce che si intendono utilizzare.

In certe situazioni può essere necessario abilitare la «gestione generica» SCSI. In particolare, questo serve nel caso si preveda l'uso di un masterizzatore SCSI.

-M- RAID Transport Class
{M} SCSI device support
-M- SCSI target support
[*] legacy /proc/scsi/ support
**** SCSI support type (disk, tape, CD-ROM) ****
<M> SCSI disk support
<M> SCSI tape support
<M> SCSI OnStream SC-x0 tape support
<M> SCSI CDROM support
[*] Enable vendor-specific extensions (for SCSI CDROM)
<M> SCSI generic support
<M> SCSI media changer support
<M> SCSI Enclosure Support
[*] Probe all LUNs on each SCSI device
[*] Verbose SCSI error reporting (kernel size +=12K)
[*] SCSI logging facility
[ ] Asynchronous SCSI scanning
SCSI Transports --->
[*] SCSI low-level drivers --->
[*] PCMCIA SCSI adapter support --->
<M> SCSI Device Handlers --->
<M> OSD-Initiator library
<M> OSD Upper Level driver
(1) (0-2) When sense is returned, DEBUG print all sense descriptors
[ ] Compile All OSD modules with lots of DEBUG prints

Per definire l'uso di questo o di quell'adattatore SCSI, occorre selezionare la voce *SCSI low-level drivers*.

8.3.8.5 Device drivers, serial ATA and paralel ATA drivers

Questa sezione riguarda la gestione complessiva delle unità ATA, sia seriali, sia parallele. Come per le unità SCSI è necessario includere il codice specifico per tutte le interfacce che si intendono utilizzare.

--- Serial ATA and Parallel ATA drivers
[*] Verbose ATA error reporting
[*] ATA ACPI Support
[*] SATA Port Multiplier support
**** Controllers with non-SFF native interface ****
<M> AHCI SATA support
< > Platform AHCI SATA support (NEW)
<M> Initio 162x SATA support
< > ACard AHCI variant (ATP 8620) (NEW)
<M> Silicon Image 3124/3132 SATA support
[*] ATA SFF support
**** SFF controllers with custom DMA interface ****
<M> Pacific Digital ADMA support
<M> Pacific Digital SATA QStor support
<M> Promise SATA SX4 support (Experimental)
[*] ATA BMDMA support (NEW)
**** SATA SFF controllers with BMDMA ****
<M> Intel ESB, ICH, PIIX3, PIIX4 PATA/SATA support
<M> Marvell SATA support
<M> NVIDIA SATA support
<M> Promise SATA TX2/TX4 support
<M> Silicon Image SATA support
<M> SiS 964/965/966/180 SATA support
<M> ServerWorks Frodo / Apple K2 SATA support
<M> ULi Electronics SATA support
<M> VIA SATA support
<M> VITESSE VSC-7174 / INTEL 31244 SATA support
**** PATA SFF controllers with BMDMA ****
<M> ALi PATA support
<M> AMD/NVidia PATA support
< > ARASAN CompactFlash PATA Controller Support (NEW)
<M> ARTOP 6210/6260 PATA support
<M> ATI PATA support
<M> ARTOP/Acard ATP867X PATA support
<M> CMD64x PATA support
<M> CS5510/5520 PATA support
<M> CS5530 PATA support
<M> CS5535 PATA support (Experimental)
<M> CS5536 PATA support
<M> Cypress CY82C693 PATA support (Very Experimental)
<M> EFAR SLC90E66 support
<M> HPT 366/368 PATA support
<M> HPT 370/370A/371/372/374/302 PATA support
<M> HPT 371N/372N/302N PATA support
<M> HPT 343/363 PATA support
[*] HPT 343/363 DMA support
<M> IT8213 PATA support (Experimental)
<M> IT8211/2 PATA support
<M> JMicron PATA support
<M> Marvell PATA support via legacy mode
<M> NETCELL Revolution RAID support
<M> Ninja32/Delkin Cardbus ATA support
<M> Nat Semi NS87415 PATA support
<M> Intel PATA old PIIX support
<M> OPTI FireStar PATA support (Very Experimental)
<M> Promise PATA 2027x support
<M> Older Promise PATA controller support
<M> RADISYS 82600 PATA support (Experimental)
<M> RDC PATA support
<M> SC1200 PATA support
<M> Intel SCH PATA support
<M> SERVERWORKS OSB4/CSB5/CSB6/HT1000 PATA support

<M> CMD / Silicon Image 680 PATA support
-M- SiS PATA support
< > Toshiba Piccolo support (Experimental)
<M> Compaq Triflex PATA support
<M> VIA PATA support
<M> Winbond SL82C105 PATA support
**** PIO-only SFF controllers ****
<M> CMD640 PCI PATA support (Experimental)
<M> ISA Plug and Play PATA support
<M> Intel PATA MPIIX support
<M> Nat Semi NS87410 PATA support
<M> OPTI621/6215 PATA support (Very Experimental)
<M> PCMCIA PATA support
<M> QDI VLB PATA support
<M> PC Tech RZ1000 PATA support
<M> Winbond W83759A VLB PATA support (Experimental)
**** Generic fallback / legacy drivers ****
<M> ACPI firmware driver for PATA
<M> Generic ATA support
-M- Legacy ISA PATA support (Experimental)

8.3.8.6 Device drivers, multi-device support

La gestione software di più unità di memorizzazione in modo combinato richiede la selezione di un gruppo speciale di opzioni. Può trattarsi di dischi o partizioni ridondanti come forma di protezione dalle perdite di dati, oppure può essere un modo per fondere assieme più partizioni in una partizione logica più grande. Se c'è la possibilità, conviene attivare almeno le funzionalità necessarie a gestire il livello RAID 1. Si veda eventualmente la sezione 19.10.

--- Multiple devices driver support (RAID and LVM)
<> RAID support*
[*] Autodetect RAID arrays during kernel boot
<M> Linear (append) mode
<M> RAID-0 (striping) mode
<M> RAID-1 (mirroring) mode
<M> RAID-10 (mirrored striping) mode
<M> RAID-4/RAID-5/RAID-6 mode
[ ] RAID-4/RAID-5/RAID-6 Multicore processing (EXPERIMENTAL)
<M> Multipath I/O support
<M> Faulty test module for MD
<M> Device mapper support
[ ] Device mapper debugging support
<M> Crypt target support
<M> Snapshot target
<M> Mirror target
< > RAID 1/4/5/6 target (EXPERIMENTAL) (NEW)
< > Mirror userspace logging (EXPERIMENTAL)
<M> Zero target
<M> Multipath target
<M> I/O Path Selector based on the number of in-flight I/Os
<M> I/O Path Selector based on the service time
<M> I/O delaying target (EXPERIMENTAL)
[ ] DM uevents (EXPERIMENTAL)
< > Flakey target (EXPERIMENTAL) (NEW)

8.3.8.7 Device drivers, network device support

L'utilizzo delle funzionalità di rete richiede la selezione della gestione delle interfacce di rete relative.

--- Network device support
<M> Intermediate Functional Block support
<> Dummy net driver support*
<M> Bonding driver support
<M> MAC-VLAN support (EXPERIMENTAL)
< > MAC-VLAN based tap driver (EXPERIMENTAL)

<M> EQL (serial line load balancing) support
<M> Universal TUN/TAP device driver support
<M> Virtual ethernet pair device
<M> General Instruments Surfboard 1000
<M> ARCnet support --->
{M} Generic Media Independent Interface device support
-*- PHY Device support and infrastructure --->
[*] Ethernet (10 or 100Mbit) --->
-*- Ethernet (1000 Mbit) --->
-*- Ethernet (10000 Mbit) --->
<*> Token Ring driver support --->
[*] Wireless LAN --->
WiMAX Wireless Broadband devices --->
USB Network Adapters --->
[*] PCMCIA network device support --->
[*] Wan interfaces support --->
[*] ATM drivers --->
<M> IEEE 802.15.4 drivers --->
*** CAIF transport drivers ***
<*> FDDI driver support
<M> Digital DEFTA/DEFEA/DEFPA adapter support
[ ] Use MMIO instead of PIO
<M> SysKonnect FDDI PCI support
[*] HIPPI driver support (EXPERIMENTAL)
<M> Essential RoadRunner HIPPI PCI adapter support (EXPERIMENTAL)
[ ] Use large TX/RX rings (EXPERIMENTAL)
<M> PLIP (parallel port) support
{M} PPP (point-to-point protocol) support
[*] PPP multilink support (EXPERIMENTAL)
[*] PPP filtering
<M> PPP support for async serial ports
<M> PPP support for sync tty ports
<M> PPP Deflate compression
<M> PPP BSD-Compress compression
<M> PPP MPPE compression (encryption) (EXPERIMENTAL)
<M> PPP over Ethernet (EXPERIMENTAL)
<M> PPP over ATM
<M> SLIP (serial line) support
[*] CSLIP compressed headers
[*] Keepalive and linefill
[*] Six bit SLIP encapsulation
[*] Fibre Channel driver support
<M> Network console logging support
[ ] Dynamic reconfiguration of logging targets
[ ] Netpoll traffic trapping
<M> Virtio network driver (EXPERIMENTAL)
<M> VMware VMXNET3 ethernet driver

8.3.8.8 Device drivers, input device support

I dispositivi per l'inserimento dati richiedono la selezione di una serie di opzioni che fanno capo a questo menù:

-*- Generic input layer (needed for keyboard, mouse, ...)
{M} Support for memoryless force-feedback devices
{M} Polled input device skeleton
{M} Sparse keymap support library
*** Userland interfaces ***
-*- Mouse interface
[*] Provide legacy /dev/psaux device
(1024) Horizontal screen resolution
(768) Vertical screen resolution
<M> Joystick interface
<M> Event interface
< > Event debugging
*** Input Device Drivers ***
-*- Keyboards --->
[*] Mice --->
[*] Joysticks/Gamepads --->
[*] Tablets --->
[*] Touchscreens --->
[*] Miscellaneous devices --->
Hardware I/O ports --->

8.3.8.9 Device drivers, character devices

Un dispositivo a caratteri è quello che utilizza una comunicazione byte per byte e si contrappone a quello a blocchi con cui la comunicazione avviene attraverso l'uso di blocchi di byte di dimensione fissa.

-*- Virtual terminal
[*] Support for binding and unbinding console drivers
-*- Unix98 PTY support
[ ] Support multiple instances of devpts
[ ] Legacy (BSD) PTY support
[ ] Non-standard serial port support
<M> HSDPA Broadband Wireless Data Card - Globe Trotter
< > GSM MUX line discipline support (EXPERIMENTAL) (NEW)
< > Trace data sink for MIPI P1149.7 cJTAG standard (NEW)
[*] /dev/kmem virtual device support
Serial drivers --->
<M> Parallel printer support
[*] Support for console on line printer
<M> Support for user-space parallel port device drivers
<M> Virtio console
<M> IPMI top-level message handler --->
{*} Hardware Random Number Generator Core support
<M> Timer IOMEM HW Random Number Generator support
<M> Intel HW Random Number Generator support
<M> AMD HW Random Number Generator support
<M> AMD Geode HW Random Number Generator support
<M> VIA HW Random Number Generator support
<M> VirtIO Random Number Generator support
{M} /dev/nvram support
< > Enhanced Real Time Clock Support (legacy PC RTC driver) (NEW)
< > Generic /dev/rtc emulation (NEW)
<M> Double Talk PC internal speech card support
<M> Siemens R3964 line discipline
<M> Applicom intelligent fieldbus card support
<M> Sony Vaio Programmable I/O Control Device support (EXPERIMENTAL)
PCMCIA character devices --->
<M> ACP Modem (Mwave) support
<M> NatSemi SCx200 GPIO Support
<M> NatSemi PC8736x GPIO Support
{M} NatSemi Base GPIO Support
<M> RAW driver (/dev/raw/rawN)
(256) Maximum number of RAW devices to support (1-65536)
[*] HPET - High Precision Event Timer
[*] Allow mmap of HPET
<M> Hangcheck timer
<M> TPM Hardware Support --->
<M> Telecom clock driver for ATCA SBC
< > Log panic/oops to a RAM buffer (NEW)

8.3.8.10 Device drivers, graphic support

Questa sezione riguarda la gestione della console, dal punto di vista grafico:

<M> /dev/agpgart (AGP Support) --->
-*- VGA Arbitration
(16) Maximum number of GPUs
[ ] Laptop Hybrid Graphics - GPU switching support
<M> Direct Rendering Manager (XFree86 4.1.0 and higher DRI

support) --->
< > Intel GMA500 Stub Driver (NEW)
{M} Lowlevel video output switch controls
{} Support for frame buffer devices --->*
-- Backlight & LCD device support --->*
Display device support --->
Console display driver support --->
[*] Bootup logo --->

Può essere importante verificare la configurazione contenuta nel menù a cui si accede dalla voce *Console display driver support*:

-- VGA text console*
[*] Enable Scrollback Buffer in System RAM
(64) Scrollback Buffer Size (in KB)
<M> MDA text console (dual-headed) (EXPERIMENTAL)
{} Framebuffer Console support*
-- Map the console to the primary display device*
[*] Framebuffer Console Rotation
[ ] Select compiled-in fonts

8.3.8.11 Device drivers, sound card support

«

Questo gruppo di opzioni consente di gestire le funzionalità audio, specificando l'uso di una scheda audio particolare.

--- Sound card support
[*] Preclaim OSS device numbers
<M> Advanced Linux Sound Architecture --->
< > Open Sound System (DEPRECATED) --->

Attraverso la voce *Advanced Linux Sound Architecture* si raggiunge un menù specifico; l'altro menù, riferito alla voce *Open Sound System*, non è più consigliato.

--- Advanced Linux Sound Architecture
<M> Sequencer support
<M> Sequencer dummy client
<M> OSS Mixer API
<M> OSS PCM (digital audio) API
[*] OSS PCM (digital audio) API - Include plugin system
[*] OSS Sequencer API
<M> HR-timer backend support
[*] Use HR-timer as default sequencer timer
-- Dynamic device file minor numbers*
[*] Support old ALSA API
[*] Verbose procfs contents
[ ] Verbose printk
[ ] Debug
[*] Generic sound devices --->
-- ISA sound devices --->*
[*] PCI sound devices --->
[*] SPI sound devices --->
[*] USB sound devices --->
[*] FireWire sound devices (NEW) --->
[*] PCMCIA sound devices --->
<M> ALSA for SoC audio support --->

8.3.8.12 Device drivers, USB support

«

Questo gruppo di opzioni consente la gestione di adattatori USB e delle unità periferiche relative. In generale conviene prevedere l'uso di tutti i tipi di adattatori USB, assieme all'accesso a memorie di massa (dischi esterni), che sono componenti abbastanza comuni.

--- USB support
{M} Support for Host-side USB
[ ] USB verbose debug messages
[*] USB announce new devices
**** Miscellaneous USB options ****
[*] USB device filesystem (DEPRECATED)
[*] USB device class-devices (DEPRECATED)
[ ] Dynamic USB minor allocation

<M> USB Monitor
-M- Enable Wireless USB extensions (EXPERIMENTAL)
<M> Support WUSB Cable Based Association (CBA)
[ ] Enable CBA debug messages
**** USB Host Controller Drivers ****
<M> Cypress C67x00 HCD support
< > xHCI HCD (USB 3.0) support (EXPERIMENTAL)
<M> EHCI HCD (USB 2.0) support
[*] Root Hub Transaction Translators
[ ] Improved Transaction Translator scheduling
<M> OXU210HP HCD support
<M> ISP116X HCD support
<M> ISP 1760 HCD support
<M> ISP1362 HCD support
<M> OHCI HCD support
[ ] OHCI support for Broadcom SSB OHCI core
<M> UHCI HCD (most Intel and VIA) support
<M> Elan U132 Adapter Host Controller
<M> SL811HS HCD support
[ ] partial ISO support (NEW)
<M> CF/PCMCIA support for SL811HS HCD
<M> R8A66597 HCD support
<M> Wireless USB Host Controller Interface (WHCI) driver (EXPERIMENTAL)
<M> Host Wire Adapter (HWA) driver (EXPERIMENTAL)
**** USB Device Class drivers ****
<M> USB Modem (CDC ACM) support
<M> USB Printer support
<M> USB Wireless Device Management support
<M> USB Test and Measurement Class support
**** NOTE: USB_STORAGE depends on SCSI but BLK_DEV_SD may ****
**** also be needed; see USB_STORAGE Help for more info ****
<M> USB Mass Storage support
[ ] USB Mass Storage verbose debug
< > Realtek Card Reader support (NEW)
<M> Datafab Compact Flash Reader support
<M> Freecom USB/ATAPI Bridge support
<M> ISD-200 USB/ATA Bridge support
<M> USBAT/USBAT02-based storage support
<M> SanDisk SDDR-09 (and other SmartMedia, including DPCM) support
<M> SanDisk SDDR-55 SmartMedia support
<M> Lexar Jumpshot Compact Flash Reader
<M> Olympus MAUSB-10/Fuji DPC-R1 support
<M> Support OneTouch Button on Maxtor Hard Drives
<M> Support for Rio Karma music player
<M> SAT emulation on Cypress USB/ATA Bridge with ATACB
< > USB ENE card reader support (NEW)
< > USB Attached SCSI (NEW)
[ ] The shared table of common (or usual) storage devices
**** USB Imaging devices ****
<M> USB Mustek MDC800 Digital Camera support
<M> Microtek X6USB scanner support
**** USB port drivers ****
<M> USS720 parport driver
<M> USB Serial Converter support --->
**** USB Miscellaneous drivers ****
<M> EMI 6|2m USB Audio interface support
<M> EMI 2|6 USB Audio interface support
<M> ADU devices from Ontrak Control Systems
<M> USB 7-Segment LED Display
<M> USB Diamond Rio500 support
<M> USB Lego Infrared Tower support
<M> USB LCD driver support
<M> USB LED driver support

<M> Cypress CY7C63xxx USB driver support
<M> Cypress USB thermometer driver support
<M> Siemens ID USB Mouse Fingerprint sensor support
<M> Elan PCMCIA CardBus Adapter USB Client
<M> Apple Cinema Display support
<M> USB 2.0 SVGA dongle support (Net2280/SiS315)
[*] Text console and mode switching support
<M> USB LD driver
<M> PlayStation 2 Trance Vibrator driver support
<M> IO Warrior driver support
<M> USB testing driver
<M> iSight firmware loading support
< > USB YUREX driver support (NEW)
<M> USB DSL modem support --->
<M> USB Gadget Support --->
**** OTG and related infrastructure ****
<M> GPIO based peripheral-only VBUS sensing 'transceiver'
<M> NOP USB Transceiver Driver

8.3.9 Filesystems

Attraverso questa sezione si definiscono i tipi di file system che si vogliono gestire. In particolare, anche i file system virtuali, come '`/proc/`' e '`/dev/pty/`', vengono definiti qui.

Il file system standard dei sistemi GNU/Linux è il tipo Second-extended, ovvero Ext2, Ext3 o Ext4. La gestione di questo tipo di file system deve essere inclusa nel kernel, a meno che si stia cercando di produrre del codice specifico per un'applicazione particolare.

In questo gruppo di opzioni trovano posto anche quelle necessarie alla condivisione attraverso la rete, per esempio con il protocollo NFS.

È interessante osservare che è necessario specificare anche i sistemi di partizionamento dei dischi. In generale è indispensabile la gestione delle partizioni tipiche dei sistemi Dos.

Infine, è importante anche tenere in considerazione il tipo di codifica che si vuole poter utilizzare nell'ambito del file system. La codifica in questione riguarda il modo di rappresentare i nomi dei file, che potrebbe richiedere estensioni particolari. In generale viene abilitata la codifica UTF-8.

<M> Second extended fs support
[*] Ext2 extended attributes
[*] Ext2 POSIX Access Control Lists
[*] Ext2 Security Labels
[ ] Ext2 execute in place support
<M> Ext3 journalling file system support
[*] Default to 'data=ordered' in ext3
[*] Ext3 extended attributes
[*] Ext3 POSIX Access Control Lists
[*] Ext3 Security Labels
<M> The Extended 4 (ext4) filesystem
[*] Ext4 extended attributes
[*] Ext4 POSIX Access Control Lists
[*] Ext4 Security Labels
[ ] EXT4 debugging support
[ ] JBD (ext3) debugging support
[ ] JBD2 (ext4) debugging support
<M> Reiserfs support
[ ] Enable reiserfs debug mode
[ ] Stats in /proc/fs/reiserfs
[*] ReiserFS extended attributes
[*] ReiserFS POSIX Access Control Lists
[*] ReiserFS Security Labels
<M> JFS filesystem support

[*] JFS POSIX Access Control Lists
[*] JFS Security Labels
[ ] JFS debugging
[ ] JFS statistics
<M> XFS filesystem support
[*] XFS Quota support
[*] XFS POSIX ACL support
[*] XFS Realtime subvolume support
[ ] XFS Debugging support (EXPERIMENTAL)
<M> GFS2 file system support
[*] GFS2 DLM locking
<M> OCFS2 file system support
<M> O2CB Kernelspace Clustering
<M> OCFS2 Userspace Clustering
[*] OCFS2 statistics
[*] OCFS2 logging support
[ ] OCFS2 expensive checks
<M> Btrfs filesystem (EXPERIMENTAL) Unstable disk format
[*] Btrfs POSIX Access Control Lists
<M> NILFS2 file system support (EXPERIMENTAL)
[*] Dnotify support
[*] Inotify support for userspace
[ ] Filesystem wide access notification (NEW)
-- Quota support*
[*] Report quota messages through netlink interface
[ ] Print quota warnings to console (OBSOLETE)
[ ] Additional quota sanity checks
<M> Old quota format support
<M> Quota format vfsv0 and vfsv1 support
<M> Kernel automounter version 4 support (also supports v3)
<M> FUSE (Filesystem in Userspace) support
<M> Character device in Userspace support
Caches --->
CD-ROM/DVD Filesystems --->
DOS/FAT/NT Filesystems --->
Pseudo filesystems --->
[*] Miscellaneous filesystems --->
[*] Network File Systems --->
Partition Types --->
-- Native language support --->*
-M- Distributed Lock Manager (DLM) --->

La voce *Pseudo filesystems* consente di accedere alle funzioni relative a file system virtuali:

-- /proc file system support*
[*] /proc/kcore support
-- Virtual memory file system support (former shm fs)*
[*] Tmpfs POSIX Access Control Lists
-- Tmpfs extended attributes*
[*] HugeTLB file system support
{M} Userspace-driven configuration filesystem

8.3.10 Kernel hacking

Attraverso questa sezione si possono attivare alcune funzionalità, utili generalmente per chi partecipa allo sviluppo del kernel Linux. C'è comunque almeno una voce che può interessare alla maggior parte degli utenti.

[ ] Show timing information on printks
(4) Default message log level (1-7) (NEW)
[ ] Enable __deprecated logic
[ ] Enable __must_check logic
(1024) Warn for stack frames larger than (needs gcc 4.4)
[*] Magic SysRq key
[*] Strip assembler-generated symbols during link
[ ] Enable unused/obsolete exported symbols
[*] Debug Filesystem
[ ] Run 'make headers_check' when building vmlinux

[ ] Enable full Section mismatch analysis (NEW)
[ ] Kernel debugging
[ ] RCU debugging: sparse-based checks for pointer usage (NEW)
[*] Compile the kernel with frame pointers
(60) RCU CPU stall timeout in seconds (NEW)
[*] Print additional per-task information for RCU_CPU_STALL_DETECTOR (NEW)
< > Linux Kernel Dump Test Tool Module
[*] Sysctl checks
[ ] Tracers --->
[ ] Remote debugging over FireWire early on boot
[ ] Remote debugging over FireWire with firewire-ohci
[ ] Enable dynamic printk() support
[ ] Enable debugging of DMA-API usage
[ ] Perform an atomic64_t self-test at boot (NEW)
< > Self test for hardware accelerated raid6 recovery
[ ] Sample kernel code --->
< > Test kstrto() family of functions at runtime (NEW)*
[*] Filter access to /dev/mem
[*] Enable verbose x86 bootup info messages
-- Early printk*
[ ] Early printk via EHCI debug port
[ ] Set loadable kernel module data as NX and text as RO (NEW)
[ ] Enable IOMMU stress-test mode
IO delay type (port 0x80 based port-IO delay [recommended])
--->
[ ] Allow gcc to uninline functions marked 'inline'

La voce *Magic SysRq key*, se attivata, consente di disporre sempre di un accesso privilegiato ad alcune funzioni attraverso una console, indipendentemente dal fatto che su questa sia attiva o meno una sessione di lavoro. Queste funzioni si ottengono con delle combinazioni di tasti, costituite da [*R_Sist AltGr* ***x***], dove ***x*** è una lettera che cambia in funzione dell'operazione che si vuole compiere. Il tasto [*R_Sist*] corrisponde a [*SysRq*] nelle tastiere per la lingua inglese, da cui viene il nome della voce di menù. Si osservi anche che il tasto [*R_Sist*] o [*SysRq*], coincide con il tasto [*Stampa*] o [*Print_Screen*].

8.4 Come fare per configurare correttamente il kernel che si vuole compilare

Il kernel Linux è molto dinamico e il suo sviluppo prende spesso delle strade imprevedibili. Questa vitalità è molto importante per il futuro del software libero; senza di essa non ci sarebbe modo di usare domani le nuove tecnologie proposte. In questo senso, diventa difficile dare delle indicazioni precise e durature sul modo corretto di configurare il kernel prima della compilazione.

L'unica documentazione sicura sotto questo aspetto è quella che si può consultare in modo contestuale quando si utilizza il comando '**`make menuconfig`**' (o altro equivalente). Eventualmente, può essere utile sapere che le informazioni che si leggono lì si trovano in vari file '`Kconfig`' collocati nelle directory contenenti il codice relativo al contesto a cui si riferiscono. Segue un estratto di uno di questi file '`Kconfig`', precisamente si tratta di '`drivers/usb/Kconfig`':

```
#
# USB device configuration
#

menu "USB support"

# ARM SA1111 chips have a non-PCI based "OHCI-compatible"
# USB host interface.
config USB
        tristate "Support for USB"
        depends on PCI || SA1111
        ---help---
          Universal Serial Bus (USB) is a specification for
          a serial bus subsystem which offers higher speeds
          and more features than the traditional PC serial
          port.  The bus supplies power to peripherals and
          …
```



```
          Say Y here if your computer has a USB port and you
          want to use USB devices.  You then need to say Y
          to at least one of "UHCI HCD support" or "OHCI HCD
          …
```

Quando si parte da zero, è sufficiente accertarsi di eliminare il file '`.config`', che comunque viene eliminato con il comando '**`make mrproper`**'. In questo modo, il programma che guida alla configurazione del kernel offre già le risposte più ovvie alle domande che fa. Naturalmente è sempre necessario leggere le prime volte il testo delle spiegazioni disponibili, fino a che si raggiunge una dimestichezza adeguata al tipo di esigenze che si hanno. Eventualmente, si può generare una prima configurazione predefinita con l'aiuto di '**`make defconfig`**', da modificare poi sulla base delle proprie esigenze.

Come la documentazione interna suggerisce spesso, nella directory '`Documentation/`' sono contenuti tanti file di testo con spiegazioni particolareggiate rispetto a problemi specifici della configurazione. A questo punto dovrebbe essere evidente che non si può configurare e compilare un kernel se non si conosce minimamente la lingua inglese.

Questo tipo di lavoro passa poi necessariamente per una lunga serie di tentativi falliti (avendo cura di conservare i file '`.config`', per poter ripartire almeno dall'ultima configurazione tentata). Tuttavia, il principiante non deve pensare di essersi messo involontariamente nei guai, perché queste difficoltà riguardano tutti, anche gli esperti, proprio perché la dinamicità nello sviluppo del kernel Linux porta continue novità.

8.5 Parametri di avvio del kernel Linux

Il kernel è in grado di ricevere opzioni in fase di avvio che possono servire per scopi differenti. In particolare, per gestire alcuni dispositivi è necessario informare il kernel sulle loro caratteristiche (tipicamente l'indirizzo di I/O e il livello di IRQ). Nel capitolo 6 è descritto il meccanismo attraverso cui si avvia il sistema e i vari modi di passare al kernel queste istruzioni particolari. Qui vengono mostrati solo alcuni dei parametri di avvio che possono essere utilizzati.

È importante tenere presente che gli indirizzi di I/O vanno espressi in esadecimale, nella forma 0x***nnn***, il livello di IRQ viene indicato in modo decimale e l'indirizzo di memoria condivisa viene espresso in esadecimale. Se non viene indicato diversamente, gli indirizzi di I/O e di memoria condivisa sono quelli di partenza.

8.5.1 File system principale (root)

Nel momento dell'avvio, il kernel deve conoscere alcune informazioni essenziali sul file system principale. Per prima cosa deve sapere dove si trova (quale disco e quale partizione), quindi deve sapere in che modo va innestato inizialmente (in sola lettura o anche in scrittura).

- Selezione del file system principale.

  ```
  root=dispositivo
  ```

  Permette di specificare un dispositivo differente da quello predefinito per innestare il file system principale. Il dispositivo può essere rappresentato nel modo consueto, per esempio '**`root=/dev/sda2`**', anche se in questa fase di avvio non esiste ancora un file system all'interno del quale cercare un tale file di dispositivo.

- Accesso iniziale in sola lettura.

  ```
  ro
  ```

  Permette di definire un accesso iniziale al file system principale in sola lettura. Questa è la condizione necessaria per poter eseguire un controllo dell'integrità del file system prima di passare alla gestione normale.

- Accesso iniziale in lettura e scrittura.

```
rw
```

Permette di definire un accesso iniziale al file system principale in lettura e scrittura.

8.5.2 Memoria

- Memoria RAM disponibile.

```
mem=dimensione
```

In caso di necessità, permette di definire la dimensione di memoria RAM disponibile effettivamente. Si può indicare un numero esadecimale nella forma 0x..., oppure un numero decimale normale, seguito eventualmente dalla lettera '**k**', che sta a indicare kibibyte (simbolo: «Kibyte»), oppure dalla lettera '**M**', che sta a indicare mebibyte (simbolo: «Mibyte»). Quando ci si trova nella necessità di indicare la memoria esistente occorre fare attenzione a non esagerare, soprattutto occorre controllare se una parte di questa, verso la fine, viene utilizzata dal firmware BIOS, perché in tal caso va specificata una dimensione inferiore.

- Caricamento del file system principale come disco RAM.

```
load_ramdisk={0|1}
```

Permette di definire se si vuole caricare il file system principale come disco RAM. '**load_ramdisk=1**' significa che si intende attivare un disco RAM, mentre assegnando il valore zero ciò non accade. Il valore predefinito è '**load_ramdisk=0**'.

8.5.3 Varie

- Programma Init.

```
init=programma_iniziale
```

Permette di definire il nome, completo di percorso, del programma che deve svolgere la funzione di «processo iniziale» (Init). Normalmente, il kernel provvede da solo a cercare '`/sbin/init`' e in alternativa '`/etc/init`'. Come ultima risorsa tenta di avviare '`/bin/sh`'. Se per qualunque motivo non funziona il programma Init standard, si può tentare di avviare il sistema facendo partire la shell al suo posto.

- Riavvio automatico quando si manifesta un *kernel panic*.

```
panic=secondi
```

Quando il kernel incontra un problema grave, si dice scherzosamente che si è verificato un *kernel panic*. In situazioni normali, il sistema si blocca in attesa di un intervento umano. Attraverso questa istruzione è possibile indicare al kernel di riavviare il sistema dopo un intervallo di tempo espresso in secondi. Il valore predefinito è zero, corrispondente a una durata infinita.

- Esclusione degli indirizzi di I/O.

```
reserve=indirizzo_i/o,estensione[,indirizzo_i/o,estensione]...
```

Permette di isolare una o più zone di indirizzi di I/O, all'interno delle quali il kernel non deve eseguire alcun tentativo di identificazione di componenti. Di solito, dopo un'istruzione del genere, si inseriscono le dichiarazioni esplicite dei dispositivi che ci sono effettivamente. Il primo valore, quello che esprime l'indirizzo, viene espresso attraverso una notazione esadecimale del tipo consueto (0x...), mentre il secondo è un numero decimale. L'esempio seguente mostra come riservare gli indirizzi di I/O da 300_{16} a 340_{16}:

```
reserve=0x300,64
```


- Disabilitazione del supporto APM.

```
apm=off
```

Quando il supporto per l'APM (*Advanced power management*) è inserito nel kernel e la sua gestione è incompatibile con l'hardware disponibile, per evitare il blocco del sistema è meglio disabilitare questa gestione.

- Controllo delle funzioni ACPI.

```
acpi=off|strict
```

Quando il supporto per l'ACPI (*Advanced configuration and power interface*) è inserito nel kernel e la sua gestione è incompatibile con l'hardware disponibile, per evitare il blocco del sistema è possibile disattivarlo con '**acpi=off**', oppure limitarlo in modo che sia attivo solo se l'hardware è perfettamente compatibile con '**acpi=strict**'. Se la CPU è composta da più nuclei, oppure se sono presenti più CPU, le funzionalità ACPI non possono essere disattivate del tutto.

- Controllo del bus PCI.

```
pci=noacpi
```

Quando il supporto per l'ACPI (*Advanced configuration and power interface*) è inserito nel kernel e la sua gestione è incompatibile con l'hardware disponibile, per evitare il blocco del sistema è possibile disattivarlo, per ciò che riguarda il bus PCI, con '**pci=noacpi**'. Esistono comunque altre opzioni per il controllo del bus PCI.

8.5.4 Interfacce di rete Ethernet

Le istruzioni di avvio riferite alle interfacce di rete Ethernet sono un po' strane e iniziano sempre con il parametro '**ether=**'. La sintassi generale è la seguente:

```
ether=livello_irq,indirizzo_i/o[,extra1,[extra2...]],nome
```

In pratica, gli argomenti che identificano il livello di IRQ, l'indirizzo di I/O e il nome simbolico dell'interfaccia di rete sono sempre parte della sintassi, mentre altri argomenti nella parte centrale possono essere presenti in funzione del tipo di scheda.

- Adattatori compatibili con NE1000 e NE2000.

```
ether=livello_irq,indirizzo_i/o,nome
```

I valori che si vuole siano determinati automaticamente vanno lasciati a zero.

- Adattatore 3Com 3c503.

```
ether=livello_irq,indirizzo_i/o,0,ricetrasmettitore,nome
```

L'argomento indicato come ricetrasmettitore è un numero che rappresenta il tipo di connessione fisica scelta: 0=interno (BNC), 1=esterno (AUI).

Segue la descrizione di alcuni esempi.

- `ether=11,0x300,eth0`

  Scheda NE2000: indirizzo I/O 300_{16}, livello di IRQ 11.

- `ether=11,0x300,eth0 ether=10,320,eth1`

  Due schede NE2000: la prima configurata con indirizzo I/O 300_{16} e livello di IRQ 11, la seconda con indirizzo di I/O 320_{16} e livello di IRQ 10.

- `ether=0,0,eth1`

  Due schede NE2000 configurate in modo differente e non conflittuale, in grado di essere riconosciute se installate singolarmente. Si vuole ottenere l'autorilevamento della seconda scheda.

- `ether=9,0x300,0,1,eth0`

  Scheda 3c503: indirizzo di I/O 300_{16}, livello di IRQ 9, ricetrasmettitore esterno.

- `ether=3,0,0,0,eth0`

  Scheda 3c503: si vuole che venga rilevato automaticamente l'indirizzo di I/O, mentre il livello di IRQ è 3. Si utilizza il ricetrasmettitore interno.

8.6 Gestione dei moduli

I moduli del kernel sono porzioni di questo che possono essere caricate in memoria quando se ne presenta la necessità e scaricate subito dopo. I moduli del kernel Linux sono spesso quello che in altri sistemi viene definito *driver*. In generale, se si dispone di una distribuzione GNU/Linux organizzata con un kernel modulare, è consigliabile sfruttare quel kernel già predisposto, assieme ai suoi moduli.

Il minimo indispensabile per attivare e disattivare i moduli è costituito da due programmi di servizio specifici: '**insmod**' e '**rmmod**'. Il primo serve per caricare i moduli, il secondo per scaricarli. L'operazione di caricamento dei moduli deve essere fatta tenendo presente le eventuali dipendenze che ci possono essere. Per esempio, se il modulo «C» richiede la presenza del modulo «B», il quale a sua volta richiede la presenza del modulo «A», occorre caricare ordinatamente i moduli «A», «B» e «C». Nello stesso modo, lo scarico dei moduli può essere fatto solo se si rispettano le dipendenze. Nel caso appena descritto, per scaricare il modulo «A» occorre prima scaricare «C» e «B».

8.6.1 Dipendenza tra i moduli

I moduli sono generalmente file che terminano con l'estensione '`.ko`' e si collocano al di sotto della directory '`/lib/modules/`***versione***`/`', dove la versione si riferisce al kernel per il quale sono stati predisposti. Per esempio, '`/lib/modules/2.6.23.1/`', si riferisce ai moduli del kernel 2.6.23.1. Per facilitare l'individuazione e il caricamento dei moduli, viene creato generalmente un file, '`modules.dep`', nella directory iniziale di questi, attraverso il programma '**depmod**':

```
# depmod -a [Invio]
```

Il file contiene l'elenco dei moduli presenti, con l'indicazione precisa delle dipendenze. L'esempio seguente mostra il caso del modulo della scheda di rete NE2000, '`ne.ko`', il quale dipende dal modulo '`8390.ko`'.

```
kernel/drivers/net/ne.ko: kernel/drivers/net/8390p.ko
```

Invece di caricare i moduli con il programma '**insmod**', cosa che richiede attenzione nella sequenza di caricamento a causa delle dipendenze, si può utilizzare '**modprobe**' che si avvale del file '`modules.dep`' e si arrangia a caricare tutto quello che serve nel modo corretto. Per esempio, utilizzando il comando seguente si ottiene prima il caricamento del modulo '`8390.ko`' e successivamente di '`ne.ko`':

```
# modprobe ne [Invio]
```

8.6.2 Parametri

Come accade già con la porzione principale del kernel, alcuni dispositivi possono essere individuati e gestiti correttamente solo se si forniscono delle informazioni aggiuntive. Per questo, alcuni moduli richiedono l'indicazione di parametri composti dalla sintassi seguente:

simbolo [***=valore***]

Quando si caricano i moduli di questo tipo con '**insmod**' è necessario fornire anche i parametri, nella parte finale della riga di comando. La stessa cosa vale per '**modprobe**', solo che in questo caso si può realizzare un file di configurazione, '`/etc/modprobe.conf`' oppure '`/etc/modprobe.d/`*...*`.conf`', contenente le informazioni sui parametri dei moduli utilizzati e altre indicazioni eventuali.

Per esempio, attraverso la riga seguente del file '`/etc/modprobe.conf`' o in un altro file della directory '`/etc/modprobe.d<`/', si vuole specificare che l'indirizzo di I/O del dispositivo relativo al modulo '`ne.ko`' è 300_{16}:

```
options ne io=0x0300
```

8.6.3 Utilizzo di «insmod»

Il programma '**insmod**' permette di caricare un modulo nel kernel. Il nome del modulo può essere indicato specificando il nome del file completo di estensione ed eventualmente di percorso (*path*), oppure specificando semplicemente il nome del file del modulo senza l'estensione: in questo ultimo caso, '**insmod**' cerca il file (con la sua estensione naturale) all'interno delle directory standard per i moduli.

`insmod` [***opzioni***] ***file_oggetto*** [***simbolo=valore***...]

Quando nel kernel è attivata la gestione del *kernel daemon* e il demone '**kerneld**' è in funzione, non dovrebbe essere necessario l'utilizzo di '**insmod**' per caricare i moduli. Segue la descrizione di alcuni esempi.

- `# insmod /lib/modules/2.6.23.1/kernel/drivers/net/plip.ko` [*Invio*]

  Attiva il modulo '**plip**' rappresentato dal file '`/lib/modules/2.6.23.1/kernel/drivers/net/plip.ko`'.

- `# insmod plip` [*Invio*]

  Come nell'esempio precedente, ma si lascia a '**insmod**' il compito di cercare il file.

8.6.4 Utilizzo di «rmmod»

Il programma '**rmmod**' permette di scaricare uno o più moduli dal kernel, sempre che questi non siano in uso e non ci siano altri moduli caricati che vi fanno riferimento.

`rmmod` [***opzioni***] ***modulo***...

Nella riga di comando vengono indicati i nomi dei moduli e non i nomi dei file dei moduli. Se vengono indicati più moduli, questi vengono scaricati nell'ordine in cui appaiono. Se viene usata l'opzione '**-a**', senza indicare nomi di moduli, vengono scaricati tutti i moduli che risultano essere inattivi. Segue la descrizione di alcuni esempi.

- `# rmmod plip` [*Invio*]

  Scarica il modulo '**plip**'.

- `# rmmod -a` [*Invio*]

  Scarica tutti i moduli inutilizzati.

8.6.5 Utilizzo di «lsmod»

Il programma '**lsmod**' permette di visualizzare la situazione sull'utilizzo dei moduli. Le stesse informazioni ottenibili da '**lsmod**' si possono avere dal contenuto del file '`/proc/modules`'. Utilizzando '**lsmod**' si ottiene una tabellina di tre colonne:

'**Module**'	rappresenta il nome del modulo;
'**Pages**'	rappresenta il numero di pagine di memoria utilizzate (una pagina è un blocco di 4 Kibyte);
'**Used by**'	rappresenta l'utilizzo da parte di altri moduli (lo zero indica che non è utilizzato).

Ecco cosa potrebbe apparire:

```
# lsmod [Invio]

Module                  Size  Used by
```

```
nfsd                  205680  13
exportfs                6528  1 nfsd
…
parport_pc             33956  1
parport                34888  2 lp,parport_pc
psmouse                35976  0
…
atl1                   33676  0
mii                     6272  1 atl1
nfs                   225132  0
lockd                  59528  3 nfsd,nfs
nfs_acl                 4480  2 nfsd,nfs
sunrpc                156508  10 nfsd,nfs,lockd,nfs_acl
msdos                  10112  0
vfat                   13056  0
fat                    49564  2 msdos,vfat
…
intel_agp              23964  1
agpgart                33228  1 intel_agp
```

8.6.6 Utilizzo di «depmod»

Il programma '**depmod**' serve a generare un file di dipendenze tra i moduli, che poi viene utilizzato da '**modprobe**' per caricarli rispettando le dipendenze. Precisamente, viene creato il file '`/lib/modules/`***versione***`/modules.dep`'.

```
depmod [opzioni]
```

Tabella 8.13. Alcune opzioni.

Opzione	Descrizione
`-a` [*versione*] `--all` [*versione*]	Scandisce tutti i moduli della versione del kernel in funzione. '**depmod**' viene utilizzato generalmente con questa opzione per creare il file delle dipendenze. Se si desidera creare il file delle dipendenze per i moduli di un'altra versione di kernel, si può specificare espressamente tale versione.
`-s` `--system-log`	Invia le segnalazioni di errore al registro del sistema.

Segue la descrizione di alcuni esempi.

- `#` **`depmod -a`** [*Invio*]

  Genera il file '`/lib/modules/`***versione***`/modules.dep`'.

- `#` **`depmod -a 2.6.23.1`** [*Invio*]

  Genera il file '`/lib/modules/2.6.23.1/modules.dep`', riferito appunto ai moduli del kernel della versione 2.6.23.1.

```
if [ -x /sbin/depmod ]
then
    echo "Analisi delle dipendenze tra i moduli"
    /sbin/depmod -a
fi
```

  Si tratta di un pezzo di uno degli script della procedura di inizializzazione del sistema, in cui si avvia la generazione del file delle dipendenze tra i moduli solo se il programma esiste.

8.6.7 Utilizzo di «modprobe»

Il programma '**modprobe**' è fatto per agevolare il caricamento dei moduli del kernel.

```
modprobe [opzioni] file_oggetto [simbolo=valore…]
```

Quando viene usato senza l'indicazione di alcuna opzione, cioè solo con il nome del modulo e l'eventuale aggiunta dei parametri, '**modprobe**' carica prima i moduli necessari a soddisfare le dipendenze, quindi provvede al caricamento del modulo richiesto. Se l'operazione fallisce, tutti i moduli superflui vengono scaricati nuovamente.

Tra le altre cose, '**modprobe**' permette di tentare il caricamento del modulo «giusto» a partire da un gruppo, quando non si conosce bene quale sia il modulo adatto a un certo tipo di dispositivo o di servizio. Per farlo è necessario indicare il tipo di modulo e il modello. Il tipo è rappresentato dalla directory che lo contiene ('`fs/`', '`misc/`', '`net/`', '`scsi/`', ecc.) e il modello si esprime utilizzando i consueti metacaratteri ('**`?`**' e '**`*`**').

Il programma '**modprobe**' fa uso di uno o più file di configurazione, attraverso cui è possibile modificare le sue impostazioni predefinite e in particolare si possono definire i parametri normali necessari ad alcuni tipi di moduli. Il file in questione è '`/etc/modprobe.conf`', oppure l'insieme dei file contenuti nella directory '`/etc/modprobe.d/`'.

Tabella 8.15. Alcune opzioni.

Opzione	Descrizione
`-a` `--all`	Carica tutti i moduli (generalmente non viene utilizzata questa opzione).
`-c` `--show-conf`	Emette la configurazione attuale per la gestione dei moduli; ciò comprende sia la parte predefinita, sia il contenuto dei file di configurazione ('`/etc/modules.conf`' oppure la directory '`/etc/modprobe.d/`').
`-l` `--list`	Elenca i moduli disponibili.
`-r` `--remove`	Scarica i moduli dal kernel, eliminando anche quelli che sono stati caricati per soddisfare le dipendenze, sempre che ciò sia possibile.
`-t` ***tipo modello*** `--type` ***tipo modello***	Permette di definire il tipo di modulo, attraverso il nome usato per la directory che lo contiene ('`fs/`', '`misc/`', '`net/`', '`scsi/`',...) e attraverso un modello espresso con dei metacaratteri. Utilizzando questa opzione, occorre fare attenzione a proteggere i metacaratteri dall'interpretazione da parte della shell, per esempio con l'uso di apici singoli o doppi.

Segue la descrizione di alcuni esempi.

- `#` **`modprobe -l`** [*Invio*]

  Elenca tutti i moduli disponibili.

- `#` **`modprobe -l -t net`** [*Invio*]

  Elenca tutti i moduli di tipo '**`net`**', cioè quelli contenuti nella directory omonima.

- `#` **`modprobe -l -t net '3c*'`** [*Invio*]

  Elenca i moduli il cui nome inizia per '**`3c`**', di tipo '**`net`**'; in pratica elenca i moduli delle schede di rete 3Com.

- `#` **`modprobe -c`** [*Invio*]

  Emette la configurazione attuale della gestione dei moduli di '**modprobe**'.

- `#` **`modprobe plip`** [*Invio*]

  Carica il modulo '`/lib/modules/`***versione***`/kernel/drivers/net/plip.ko`'.

- `#` **`modprobe -t net 'p*'`** [*Invio*]

  Tenta di caricare un modulo che inizi con la lettera '**`p`**', dalla directory '`/lib/modules/`***versione***`/kernel/drivers/net/`'.

8.6.8 Informazione sui moduli

I moduli sono provvisti di una breve descrizione sul loro scopo e di altre informazioni. Per ottenere questi dati, è possibile usare il programma '**modinfo**':

```
modinfo [opzioni] nome_modulo|nome_file
```

In pratica, è sufficiente indicare al programma il nome del modulo o il nome del file che costituisce il modulo, per ottenere le informazioni disponibili:

```
$ modinfo whci [Invio]

filename:     /lib/modules/2.6.34.1/kernel/drivers/uwb/whci.ko
license:      GPL
author:       Cambridge Silicon Radio Ltd.
description:  WHCI UWB Multi-interface Controller enumerator
alias:        pci:v*d*sv*sd*bc0Dsc10i10*
depends:      umc
vermagic:     2.6.34.1 SMP preempt mod_unload modversions
```

8.7 Configurazione dei moduli

Il file '`/etc/modprobe.conf`', ovvero l'insieme di file contenuto nella directory '`/etc/modprobe.d/`', permette di configurare il comportamento di '**modprobe**'. Le righe vuote e quanto preceduto dal simbolo '`#`' viene ignorato. Le righe possono essere continuate utilizzando la barra obliqua inversa ('\') alla fine, subito prima del codice di interruzione di riga.

Le righe di questo file vengono interpretate attraverso una shell, permettendo così di utilizzare le tecniche di sostituzione fornite comunemente da queste, come i metacaratteri e con la sostituzione di comando.

Questo file di configurazione può contenere diversi tipi di direttive; nelle sezioni seguenti se ne mostrano solo alcune. Per la descrizione completa si veda la pagina di manuale *depmod(1)*.

In linea di massima, si possono accumulare più direttive dello stesso tipo.

8.7.1 Direttiva alias

La direttiva '**alias**' permette di indicare un nome alternativo a un nome di un modulo reale:

```
alias alias modulo_reale
```

Ciò può essere utile a vario titolo e in ogni caso sono stabiliti molti alias già in modo predefinito. Lo si può osservare con il comando seguente:

```
# modprobe -c [Invio]

…
alias block-major-2 floppy
alias block-major-3 ide-probe-mod
…
alias char-major-4 serial
alias char-major-5 serial
alias char-major-6 lp
…
alias dos msdos
…
alias iso9660 isofs
…
alias plip0 plip
alias plip1 plip
alias ppp0 ppp
alias ppp1 ppp
…
```

In questo caso, si può osservare che è possibile fare riferimento al modulo '**isofs**' anche attraverso il nome '**iso9660**'. Tuttavia, gli alias non sono semplicemente di aiuto agli «smemorati», ma anche una necessità. Si osservi la configurazione seguente tratta da un ipotetico file '`/etc/modprobe.conf`'.


```
alias eth0 ne
…
```

L'alias '**eth0**' (ovvero la prima interfaccia Ethernet) permette di fare in modo che, quando si configura l'interfaccia di rete con '**ifconfig**' venga avviato automaticamente il modulo corretto: in questo caso '**ne**'.

Ogni modulo ha le sue particolarità, quindi deve essere valutata caso per caso l'opportunità di utilizzare un alias adatto a qualche scopo.

8.7.2 Direttiva options

La direttiva '**options**' permette di definire i parametri di utilizzo di un modulo, identificato attraverso il suo nome reale, oppure un alias:

```
options nome simbolo=valore…
```

L'esempio seguente definisce che il modulo '**ne**' (Ethernet NE2000) deve essere utilizzato per un dispositivo che si raggiunge con il canale di I/O 300_{16} e l'IRQ 11:

```
alias eth0 ne
options ne io=0x300 irq=11
```

Attraverso questa direttiva si indicano solo le opzioni che non possono essere determinate altrimenti dal sistema. Questo significa che **non** è necessaria una riga '**options**' per tutti i dispositivi che si intende utilizzare attraverso i moduli.

8.8 Firmware

Alcuni componenti hardware hanno la necessità di caricare del codice per poter diventare operativi ed essere così in grado di comunicare correttamente con il kernel. Questo codice speciale, noto come «firmware», viene caricato nell'hardware, attraverso il kernel, quando se ne presenta la necessità, partendo da file che si collocano generalmente all'interno della directory '`/lib/firmware/`***versione***`/`', dove ***versione*** rappresenta il numero della versione dei sorgenti del kernel per il quale sono realizzati.

I sorgenti del kernel includono il codice necessario a produrre il firmware disponibile con licenze compatibili rispetto al codice complessivo; purtroppo, però, esiste dell'hardware che richiede firmware diffuso in modo binario, il cui codice originale rimane segreto. In questi casi, il firmware deve essere raccolto separatamente e collocato sempre a partire dalla directory '`/lib/firmware/`…', ma la posizione effettiva dipende poi da ciò che si aspetta il gestore del dispositivo (del kernel) che si occupa del suo caricamento.

Per fare riferimento all'azione con cui il kernel carica il firmware, nella documentazione originale si parla anche di *hotplug*. Per questa ragione, dal momento che il kernel si avvale comunque di un programma o di uno script per procedere con il caricamento del firmware necessario, questo dovrebbe corrispondere a '`/sbin/hotplug`'. In realtà, il percorso effettivo di questo programma dovrebbe essere specificato nel file virtuale '`/proc/sys/kernel/hotplug`', ma quello predefinito è proprio '`/sbin/hotplug`'.

Tra i sorgenti del kernel, precisamente nella directory '`Documentation/firmware_class/`', si trova un esempio di come potrebbe essere realizzato uno script del genere. Gli esempi successivi riguardano due versioni abbastanza recenti del kernel Linux.

Listato 8.20. Script '`/sbin/hotplug`' per un kernel 2.6.34.

```
#!/bin/sh
#
# Both $DEVPATH and $FIRMWARE are already provided in the
# environment.
#
HOTPLUG_FW_DIR=/lib/firmware/

echo 1 > /sys/$DEVPATH/loading
cat $HOTPLUG_FW_DIR/$FIRMWARE > /sys/$DEVPATH/data
echo 0 > /sys/$DEVPATH/loading
```

Listato 8.21. Script ‘/sbin/hotplug’ per un kernel 2.6.36.

```
#!/bin/sh
#
# Both $DEVPATH and $FIRMWARE are already provided in the
# environment.
#
HOTPLUG_FW_DIR=/lib/firmware/

if [ "$SUBSYSTEM" == "firmware" -a "$ACTION" == "add" ]; then
  if [ -f $HOTPLUG_FW_DIR/$FIRMWARE ]; then
    echo 1 > /sys/$DEVPATH/loading
    cat $HOTPLUG_FW_DIR/$FIRMWARE > /sys/$DEVPATH/data
    echo 0 > /sys/$DEVPATH/loading
  else
    echo -1 > /sys/$DEVPATH/loading
  fi
fi
```

Nei sistemi basati su kernel Linux che si avvalgono di uDev per la gestione automatica dei file di dispositivo, è uDev stesso che assolve anche il compito di caricare il firmware, secondo la modalità che descrivono i due script di esempio mostrati. In quei casi, di solito, nel sistema è assente del tutto il file ‘/sbin/hotplug’. Si veda eventualmente la sezione 8.9.2 che tratta di uDev.

8.9 File di dispositivo

Nei sistemi Unix, come GNU/Linux, il kernel permette alle applicazioni di comunicare con le unità fisiche, ovvero i dispositivi, attraverso un’astrazione costituita dai ***file di dispositivo***. Questi file sono di tipo speciale e tradizionalmente sono contenuti all’interno della directory ‘/dev/’ (si veda anche la sezione 20.16). La particolarità di questi file sta nella definizione di due numeri che, in pratica, costituiscono il canale di comunicazione con il kernel stesso. Si tratta del numero ***primario*** e del numero ***secondario*** (oppure *major* e *minor*, secondo la terminologia originale inglese), dove il primo rappresenta il tipo di dispositivo e il secondo serve a identificare esattamente un dispositivo particolare. Questi numeri dipendono dal kernel e di conseguenza possono variare da un sistema operativo Unix all’altro. Anche i nomi che si danno a questi file possono variare da un sistema Unix all’altro; in certi casi ci sono piccole differenze anche tra le stesse distribuzioni GNU/Linux. In ogni caso, il documento di riferimento per ciò che riguarda GNU/Linux, è il file ‘***sorgenti_linux***/Documentation/devices.txt’, corrispondente a *Linux allocated devices*, aggiornato da H. Peter Anvin.

Dal momento che questi file servono solo in quanto contengono i numeri primario e secondario di un certo dispositivo, potrebbero funzionare anche collocati al di fuori della loro directory tradizionale, utilizzando eventualmente nomi differenti. Questa possibilità viene sfruttata da alcune distribuzioni GNU/Linux, nella fase di installazione, quando nel sistema di avvio vengono creati al volo i file di dispositivo necessari a completare l’operazione, utilizzando eventualmente la directory temporanea per questo scopo.

I file di dispositivo si distinguono in due categorie, in base al fatto che l’hardware a cui corrispondono sia in grado di gestire un flusso di caratteri (ma forse è più corretto usare il termine byte in questo caso), presi ognuno singolarmente, oppure richieda che i dati siano raggruppati in blocchi di una dimensione determinata. Nel primo caso si parla di dispositivo a caratteri, mentre nel secondo di dispositivo a blocchi.

Dato che i dispositivi fisici sono gestiti attraverso questi file di dispositivo, l’accesso all’hardware viene controllato con i permessi che vengono dati a questi file. La gestione di questi permessi è molto importante nell’impostazione che viene data al sistema ed è uno dei punti su cui si trovano le differenze significative tra le varie distribuzioni GNU/Linux. Inoltre, l’esistenza di utenti e gruppi fittizi, con nomi come ‘**floppy**’, ‘**sys**’, ‘**daemon**’ e altri, dipende spesso da questa esigenza di controllo dell’accesso ai dispositivi.

8.9.1 Creazione dei file di dispositivo

Quando si ricompila il kernel per includere la gestione di funzionalità particolari, per accedere a queste, o per accedere ai componenti fisici per i quali è stata stabilita la gestione, può essere necessario intervenire nella directory ‘/dev/’ allo scopo di creare o modificare qualche file di dispositivo. In generale, le distribuzioni GNU/Linux tendono a prevedere tutti i file necessari, ma la stessa evoluzione del kernel introduce esigenze nuove e spesso la necessità di provvedere da soli a questi file. Inoltre, è difficile che siano disponibili dal principio tutti i file di dispositivo possibili e immaginabili.

I file di dispositivo si creano in particolare con il programma di servizio ‘**mknod**’:

```
mknod [-m modalità_dei_permessi] file {b|c|u} ↩
↪     [n_primario n_secondario]
```

Con la lettera «b» si crea un file di dispositivo a blocchi, mentre con la lettera «c», si crea un file di dispositivo a caratteri. Il caso particolare della lettera «u», riguarda un dispositivo a caratteri senza *buffer*.

Si osservino gli esempi seguenti:

- `# mknod -m 0600 /dev/tty9 c 4 9` [*Invio*]

  crea il file di dispositivo a caratteri ‘/dev/tty9’, concedendo soltanto i permessi di lettura e di scrittura al proprietario;

- `# mknod -m 0660 /dev/hda1 b 3 1` [*Invio*]

  crea il file di dispositivo a blocchi ‘/dev/hda1’, concedendo i permessi di lettura e scrittura all’utente proprietario e al gruppo.

Anche se ‘**mknod**’ è tutto quello che serve per creare i file di dispositivo necessari, non è sempre il mezzo più comodo per provvedere a questo problema. Infatti, occorre considerare anche le convenzioni della propria distribuzione GNU/Linux, anche per ciò che riguarda i permessi e l’appartenenza di questi file; inoltre non è sempre detto che si possano ricordare esattamente le caratteristiche dei file di dispositivo di cui si ha bisogno. Per questo viene in aiuto lo script ‘**MAKEDEV**’, che tradizionalmente si deve trovare proprio nella directory ‘/dev/’. Questo script non è standard, ma il suo scopo lo è: facilitare la creazione dei file di dispositivo.

```
/dev/MAKEDEV {dispositivo... | gruppo}
```

Generalmente si possono indicare come argomento uno o più nomi di file di dispositivo, senza il percorso. Questi dovrebbero essere creati nella directory corrente. L’esempio seguente crea il file di dispositivo corrispondente alla prima console virtuale, assegnandogli tutti gli altri attributi corretti:

```
# /dev/MAKEDEV tty1 [Invio]
```

L’esempio successivo crea il file di dispositivo corrispondente al primo disco fisso ATA, assegnandogli tutti gli altri attributi corretti:

```
# /dev/MAKEDEV hda [Invio]
```

È probabile che, al posto dei nomi dei file di dispositivo, si possano usare nomi di gruppi di questi. Per esempio, lo script ‘**MAKEDEV**’ della distribuzione GNU/Linux Debian prevede il gruppo ‘**generic**’ per fare riferimento ai file di dispositivo più comuni:

```
# /dev/MAKEDEV generic [Invio]
```

8.9.2 kernel Linux e uDev

uDev è un sistema, esterno al kernel Linux, che consente di generare automaticamente i file di dispositivo nella directory ‘/dev/’, nel momento in cui i dispositivi reali si rendono disponibili. Il sistema uDev si avvale del demone ‘**udevd**’, il quale utilizza le informazioni fornite dai file system virtuali ‘/proc/’ e ‘/sys/’. Perché il file system virtuale ‘/sys/’ venga innestato correttamente, occorre verifi-

care che il file '`/etc/fstab`' contenga la dichiarazione appropriata, molto simile a quella del file system virtuale '`/proc/`':

```
…
proc    /proc            proc    defaults        0   0
none    /proc/bus/usb    usbfs   defaults        0   0
sys     /sys             sysfs   defaults        0   0
…
```

L'esempio mostra anche gli altri file system virtuali comuni.

uDev utilizza alcuni file di configurazione che normalmente si collocano nella directory '`/etc/udev/`'. Il primo file da prendere in considerazione è '`/etc/udev/udev.conf`', il quale, di solito, viene distribuito già configurato correttamente. Ecco un esempio del suo contenuto:

```
# udev.conf

# The initial syslog(3) priority: "err", "info", "debug" or
# its numerical equivalent. For runtime debugging, the
# daemons internal state can be changed with:
# "udevcontrol log_priority=<value>".
udev_log="err"

# maximum size of the /dev tmpfs
tmpfs_size="10M"
```

Per definire le caratteristiche dei file di dispositivo da creare, inclusi i loro permessi di accesso, si utilizzano i file contenuti nella directory '`/etc/udev/rules.d/`', i quali vengono «eseguiti» secondo l'ordine lessicografico del loro nome. Per questo, di norma si tratta di collegamenti simbolici a file contenuti nella directory '`/etc/udev/`', che però hanno nomi che non tengono conto dell'ordine di esecuzione.

La sintassi con cui sono definite le direttive all'interno dei file a cui si fa riferimento nella directory '`/etc/udev/rules.d/`' è piuttosto oscura, anche se si può intuire il senso di ciò che rappresenta. Per intervenire in qualche modo nei file di dispositivo, attraverso la gestione di uDev, bisognerebbe creare un proprio file, scegliendo un nome appropriato alla posizione in cui deve trovarsi nella sequenza di esecuzione. Tuttavia, ci sarebbe anche la possibilità di usare il file '`/etc/udev/links.conf`' per cose molto semplici: creazione di collegamenti simbolici, directory e file di dispositivo particolari. L'uso di questo file viene sconsigliato, ma è bene sapere come si interpreta, se dovesse essercene uno predefinito.

```
# This file does not exist.
# Please do not ask the debian maintainer about it.
# You may use it to do strange and
# wonderful things, at your risk.

L fd             /proc/self/fd
L stdin          /proc/self/fd/0
L stdout         /proc/self/fd/1
L stderr         /proc/self/fd/2
L core           /proc/kcore
L sndstat        /proc/asound/oss/sndstat

D pts
D shm

M null           c   1 3
M console        c   5 1

# Hic sunt leones.
M ppp            c 108 0
D loop
M loop/0         b   7 0
D net
M net/tun        c  10 200
```

L'esempio appena mostrato dovrebbe essere abbastanza intuitivo: le righe che iniziano con la lettera «L» richiedono la creazione di un collegamento simbolico (*link*); le righe che iniziano con la lettera «D» richiedono la creazione di una directory; le righe con la lettera «M» richiedono la creazione di un file di dispositivo.

Per quanto riguarda l'avvio e l'arresto del demone '**`udevd`**', è auspicabile che la propria distribuzione GNU/Linux sia già organizzata per questo. A ogni modo, si intuisce che questo demone deve essere avviato molto presto nell'ambito della procedura di avvio del sistema e deve anche essere fermato solo alla fine della procedura di arresto.

Dal momento che uDev è un sistema esterno al kernel, durante la fase di avvio, c'è sicuramente un momento in cui il kernel ha innestato il file system principale, ma uDev non è ancora in funzione. Per questa ragione, è indispensabile che la directory '`/dev/`', contenga inizialmente i file di dispositivo comuni, altrimenti diventa praticamente impossibile avviare il sistema.
Successivamente, una volta attivato uDev, il contenuto della directory '`/dev/`' originale diventa accessibile a partire da '`/dev/.static/dev/`', dove è possibile intervenire per modificare i file di dispositivo statici iniziali.

I file contenuti nella directory '`/etc/udev/rules.d/`' consentono di associare un'azione agli eventi di uDev. Per esempio, alla comparsa di un file di dispositivo è possibile associargli la modifica dei permessi, ma si arriva anche a poter associare l'avvio di un processo che abbia qualcosa a che fare con questo.

A titolo di esempio, si supponga di voler attribuire i permessi di lettura e scrittura, per tutti gli utenti, ai file di dispositivo che hanno a che fare con la gestione dell'audio, ai file relativi a dischetti e dischi ottici. Si comincia esplorando la situazione della directory '`/etc/udev/rules.d/`':

```
# ls /dev/udev/rules.d [Invio]

…
020_permissions.rules
025_libgphoto2.rules
025_libsane.rules
85-pcmcia.rules
udev.rules
z20_persistent-input.rules
z20_persistent.rules
z25_persistent-net.rules
z45_persistent-net-generator.rules
z50_run.rules
z55_hotplug.rules
z60_hdparm.rules
z60_xserver-xorg-input-wacom.rules
…
```

Tra le direttive di questi file, ci potrebbero essere già quelle che si prendono cura dei file di dispositivo a cui si è interessati. Ma per questo, conviene aggiungere un altro file, in modo che appaia per ultimo, secondo l'ordine lessicografico. Per esempio: '`z90_permessi-a-tutti.rules`'. Il file in questione potrebbe essere realizzato nel modo seguente:

```
ACTION=="add", SUBSYSTEM=="sound", MODE="0666"
ACTION=="add", SUBSYSTEM=="block", KERNEL=="fd[0-9]*", ↩
↪MODE="0666"
ACTION=="add", SUBSYSTEM=="block", KERNEL=="hd[a-z]", ↩
↪DRIVERS=="ide-cdrom", MODE="0666"
```

Intuitivamente si comprende che le direttive di questi file sono composte da un elenco di attributi che servono a identificare i file di dispositivo a cui si è interessati e l'evento considerato, seguito dall'azione da compiere. In questo caso, l'azione da compiere è rappresentata dalla voce '**`MODE="0666"`**', che attribuisce i permessi di lettura e scrittura a tutti gli utenti.

Per vedere come si associa l'esecuzione di un programma a un evento dei file di dispositivo, si pensi a una stampante collegata attraverso una porta USB: quando si collega la stampante viene creato automaticamente il file di dispositivo '`/dev/usb/lp0`', mentre quando la si stacca, questo file viene eliminato. Se si usa un demone per il servizio di stampa che non è preparato per queste sorprese, occorre

aggiungere un altro file con delle direttive appropriate. Per esempio potrebbe trattarsi del file ‘`z90_stampante-usb.rules`’:

```
# Reload "lpd" when adding a printer.
ACTION=="add", SUBSYSTEMS=="usb", KERNEL=="lp[0-9]*", ↩
↪RUN+="/etc/init.d/lpd restart"

# Reload "lpd" when removing a printer.
ACTION=="remove", SUBSYSTEMS=="usb", KERNEL=="lp[0-9]*", ↩
↪RUN+="/etc/init.d/lpd restart"
#
```

In pratica, in questo esempio si fa in modo che venga eseguito il comando ‘**`/etc/init.d/lpd restart`**’ quando ci sono novità sui file di dispositivo per la stampa.

Si osservi che le direttive riferite a eventi di creazione dei file di dispositivo (‘**`ACTION="add"`**’) devono precedere quelle riferite a eventi di eliminazione degli stessi (‘**`ACTION="remove"`**’).

È importante osservare che i programmi o gli script che si avviano attraverso la gestione degli eventi di uDev non possono essere troppo impegnativi, altrimenti c’è il rischio di bloccare il sistema.

8.9.3 Messaggio: «unable to open an initial console»

Quando si utilizzano dei meccanismi automatici per la gestione dei file di dispositivo, può succedere che qualcosa non funzioni nel modo previsto. In particolare, può capitare che all’avvio il kernel non riesca a trovare il file di dispositivo ‘`/dev/console`’, cosa che si traduce nel messaggio di avvertimento seguente:

```
…
Warning: unable to open an initial console
…
```

Per poter comprendere cosa accade, è necessario avviare il sistema attraverso un’unità esterna (CD/DVD autoavviabili, memorie solide o simili), con il quale si può indagare sullo stato iniziale della directory ‘`/dev/`’ (si intende la directory ‘`/dev/`’ del file system che è in funzione quando si tenta di avviare il sistema operativo senza successo). Se questa è vuota, come può succedere se ci si affida a uDev o a Devfs, conviene predisporre manualmente dei file di dispositivo al suo interno, che vengono utilizzati in mancanza del funzionamento (tempestivo) dei sistemi automatici.

Se si dispone dello script ‘**`MAKEDEV`**’, dovrebbe essere sufficiente il comando seguente, da eseguire quando la directory corrente corrisponde alla directory ‘`dev/`’ che si intende sistemare:

```
# ./MAKEDEV generic [Invio]
```

Naturalmente, se lo script ‘**`MAKEDEV`**’ non si trova nella stessa directory, ma altrove, si deve specificare il percorso di avvio appropriato.

8.10 Disco RAM iniziale: Initrd

Un disco RAM iniziale, noto con il nome Initrd, è un file system speciale che viene realizzato in modo da essere innestato inizialmente dal kernel Linux, allo scopo di avviare un sistema minimo, con cui eseguire alcune operazioni preliminari. Al termine di queste operazioni, normalmente il sistema contenuto nel disco RAM iniziale innesta il file system standard e passa il controllo al programma Init.

La tecnica del disco RAM iniziale viene usata solitamente per caricare dei moduli prima di innestare il file system definitivo, per esempio quando il kernel richiede un modulo speciale per accedere a tale file system.

Si osservi che la realizzazione di un disco RAM iniziale è un’operazione complessa, dove occorre avere la capacità di realizzare un sistema GNU/Linux elementare, ma perfettamente autonomo.

La definizione «disco RAM» ovvero «RAM disk» deriva dalla presunzione che la memoria di massa sia costituita da un’unità a disco. La tecnologia si evolve e, oltre ai dischi, esistono altre forme di memorizzazione basate su componenti allo stato solido; tuttavia, spesso la terminologia rimane ancorata alla tecnologia del passato. Pertanto, pur chiamandosi disco RAM, si intende un file system caricato in memoria centrale, come se fosse un’unità di memorizzazione di massa.

8.10.1 Teoria e visione generale

La gestione del disco RAM iniziale deve essere abilitata nel kernel (sezione 8.3.8.3). Il file system del disco RAM che si realizza deve contenere il programma o lo script ‘**`linuxrc`**’, collocato nella directory radice. Questo programma deve svolgere tutte le funzioni che si desiderano, quindi deve innestare il file system principale previsto, spostando la directory radice all’inizio di questo, avviando lì il programma Init (‘`/sbin/init`’).

Il kernel Linux deve essere avviato specificando il file system principale, il nome del programma Init, fornendo anche l’indicazione della collocazione del file-immagine contenente il disco RAM iniziale. Le prime due informazioni si possono dare facilmente tra le opzioni di avvio del kernel e dovrebbero corrispondere alle stringhe seguenti:

```
root=/dev/ram0 init=/linuxrc rw
```

Si osservi che il disco RAM iniziale deve essere accessibile inizialmente anche in scrittura, così come suggerisce il modello mostrato sopra con l’opzione ‘**`rw`**’.

Teoricamente, l’indicazione della collocazione del disco RAM iniziale potrebbe essere data «manualmente» tra le opzioni di avvio del kernel, però in pratica diventa difficile esprimere un percorso in un disco che è ancora da innestare, pertanto si usa normalmente una direttiva separata attraverso il sistema di avvio prescelto. Vengono mostrati alcuni esempi della configurazione di sistemi di avvio comuni.

Listato 8.29. GRUB 1: il file-immagine ‘`initrd.gz`’ si trova nella directory ‘`/boot/`’ della seconda partizione del primo disco.

```
title linux
kernel (hd0,1)/boot/vmlinuz root=/dev/ram0 init=/linuxrc rw
initrd (hd0,1)/boot/initrd.gz
```

Listato 8.30. SYSLINUX: il file-immagine ‘`initrd.gz`’ si trova nella directory principale della partizione in cui si trovano i file di avvio.

```
LABEL linux
    KERNEL vmlinuz
    APPEND root=/dev/ram0 init=/linuxrc rw initrd=initrd.gz
```

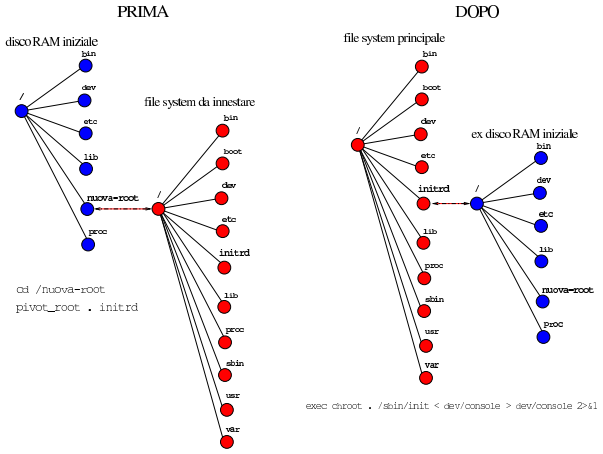
Listato 8.31. ISOLINUX: il file-immagine ‘`initrd.gz`’ si trova nella directory ‘`/isolinux/`’ del CD-ROM.

```
LABEL linux
    KERNEL vmlinuz
    APPEND root=/dev/ram0 init=/linuxrc rw initrd=initrd.gz
```

Si può vedere il disco RAM iniziale, come un file system contenente fondamentalmente un programma Init, che convenzionalmente corrisponde a ‘`/linuxrc`’. La difficoltà sta nel ridurre al minimo il sistema di questo disco RAM; eventualmente il file ‘`/linuxrc`’ potrebbe essere un programma realizzato appositamente, senza bisogno di altro.

Una volta che il programma o lo script ‘`/linuxrc`’ ha compiuto il suo lavoro, questo deve innestare il file system che deve in seguito diventare quello principale, in una directory, quindi deve eseguire la funzione ***pivot_root()*** (si veda la pagina di manuale *pivot_root(2)*) per scambiare i ruoli.

Figura 8.32. Scambio della directory principale e avvio del programma Init.



Convenzionalmente, la directory '`/initrd/`' è quella che viene usata per lo scambio delle directory principali, in modo da ospitare il file system del disco RAM iniziale. Al termine, teoricamente, è possibile eseguire anche il distacco del disco RAM iniziale e liberare la memoria che lo contiene:

```
# umount /initrd [Invio]

# blockdev --flushbufs /dev/ram0 [Invio]
```

In mancanza della possibilità di realizzare un programma '**`linuxrc`**' fatto su misura, la funzione ***pivot_root()*** può essere richiamata attraverso il programma '**`pivot_root`**',[4] che fa parte del pacchetto Util-linux; anche il programma '**`blockdev`**'[5] fa parte dello stesso pacchetto.

8.10.2 Creazione automatica di un disco RAM iniziale

Esiste il pacchetto Initrd tools[6] che dovrebbe consentire di realizzare automaticamente un file-immagine da usare come disco RAM iniziale. Teoricamente, il comando seguente dovrebbe essere sufficiente per ottenere il risultato; si veda comunque la pagina di manuale *mkinitrd(8)* per studiare il suo utilizzo.

```
mkinitrd [opzioni]
```

Il pacchetto Initrd tools può essere molto utile se risulta già predisposto correttamente secondo l'organizzazione della propria distribuzione GNU/Linux. Diversamente, richiede la preparazione di una configurazione e probabilmente diventa più conveniente produrre a mano il proprio disco RAM iniziale.

8.10.3 Esempio di un disco RAM iniziale

Per comprendere il funzionamento di un disco RAM iniziale è necessario mostrare un esempio completo, che in parte si adegua al contenuto della figura 8.32. Si comincia preparando la struttura del file system del disco RAM a partire da una directory di lavoro:

```
# cd [Invio]

# mkdir radice-initrd [Invio]

# cd radice-initrd [Invio]

# mkdir bin dev etc lib proc nuova-root [Invio]
```

Si copiano i programmi che si ritiene siano indispensabili:

```
# cp /bin/dash /bin/mkdir /bin/mount ./bin [Invio]

# cp /sbin/insmod /sbin/pivot_root ./bin [Invio]

# cp /usr/sbin/chroot ./bin [Invio]
```


In questo caso la shell è Dash e si preparano alcuni collegamenti simbolici:

```
# cd ./bin [Invio]

# ln -s dash ash [Invio]

# ln -s dash sh [Invio]

# cd .. [Invio]
```

Con l'aiuto di '**`ldd`**' si individuano i file delle librerie che richiedono i programmi copiati e si copiano a loro volta le librerie necessarie:

```
# cp /lib/libc.so.6 /lib/ld-linux.so.2 ./lib [Invio]
```

Si crea un file '`/etc/fstab`' minimo:

```
# echo "/dev/ram0 / auto defaults 0 0" > etc/fstab [Invio]
```

Si copiano i file dei moduli necessari (viene omesso il passaggio), quindi si creano i file di dispositivo che possono servire; qui viene usato lo script '**`MAKEDEV`**':

```
# cd ./dev [Invio]

# /dev/MAKEDEV console [Invio]

# /dev/MAKEDEV null [Invio]

# /dev/MAKEDEV zero [Invio]

# /dev/MAKEDEV ram [Invio]

# /dev/MAKEDEV hd [Invio]

# /dev/MAKEDEV sd [Invio]

# cd .. [Invio]
```

Alla fine, si comincia la realizzazione dello script '**`linuxrc`**', da collocare all'inizio della gerarchia ('`/linuxrc`'). Inizialmente, per verificare il funzionamento di massima del sistema, conviene fare una cosa molto semplice, come potrebbe essere questa:

```
#!/bin/sh

echo linuxrc di prova; premi [Invio] per terminare
read x
```

Una volta creato il file, occorre ricordare di dargli i permessi necessari all'esecuzione:

```
# chmod 755 linuxrc [Invio]
```

Per la verifica, basta usare '**`chroot`**':

```
# chroot . /linuxrc [Invio]

  linuxrc di prova; premi [Invio] per terminare
# [Invio]
```

Seguendo l'esempio, si ottiene il messaggio che chiede di premere [*Invio*], quindi lo script termina di funzionare e si torna alla situazione normale. A questo punto si può modificare lo script in modo da produrre gli effetti che si desiderano. In questo caso, si vuole dare il tempo al kernel di accorgersi della presenza di un disco USB ('`/dev/sda2`'), che viene innestato per l'avvio vero e proprio del sistema:

```
#!/bin/sh
echo Quando il disco USB viene riconosciuto, premere [Invio]
read var

/bin/mount -t proc proc proc

/bin/mount -o ro /dev/sda2 /nuova-root

cd /nuova-root

/bin/mkdir initrd
/bin/pivot_root . initrd

exec chroot . /sbin/init < dev/console > dev/console 2>&1
```

A questo punto si può creare il file-immagine. Questo file può essere «normale», oppure compresso con Gzip, o con l'uso del file system Cramfs (purché il kernel sia poi in grado di leggerlo così). In questo caso si segue la strada della compressione con Gzip.

```
# cd [Invio]

# dd if=/dev/zero of=file-immagine bs=1024k count=2 [Invio]

# mkfs.ext2 -F file-immagine [Invio]

# mkdir /tmp/immagine [Invio]

# mount -o loop -t ext2 file-immagine /tmp/immagine [Invio]

# cd radice-initrd [Invio]

# cp -dpR * /tmp/immagine [Invio]

# umount /tmp/immagine [Invio]

# gzip -9 < file-immagine > initrd.gz [Invio]
```

Si ottiene così il file '`initrd.gz`'. Qui si omette di proposito di completare l'esempio con la spiegazione della procedura necessaria per fare in modo che il file preparato venga usato correttamente. Nella sezione teorica ci sono diversi esempi, tuttavia, questo esempio particolare avrebbe lo scopo di avviare un disco USB.

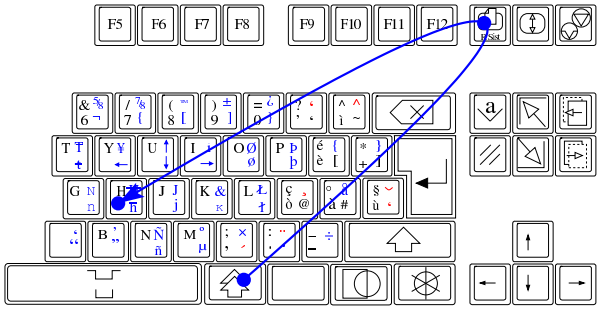
8.11 Combinazioni magiche con «R sist», ovvero «SysRq»

Se si attiva la voce *Magic SysRq key* del menù *Kernel hacking* si dispone sempre di un accesso privilegiato ad alcune funzioni attraverso una console, indipendentemente dal fatto che su questa sia attiva o meno una sessione di lavoro.

- *Kernel hacking*
  - *Magic SysRq key*

Queste funzioni si ottengono con delle combinazioni di tasti, costituite da [*R_Sist AltGr* ***x***], dove ***x*** è una lettera che cambia in funzione dell'operazione che si vuole compiere. Il tasto [*R_Sist*] corrisponde a [*SysRq*] nelle tastiere per la lingua inglese, da cui viene il nome della voce di menù. Si osservi anche che il tasto [*R_Sist*] o [*SysRq*], coincide con il tasto [*Stampa*] o [*Print_Screen*].

Figura 8.36. Combinazione di tasti [*R_Sist AltGr h*].



Ammesso che si intendano usare queste funzioni, di tutte le combinazioni va ricordata sicuramente [*R_Sist AltGr h*], come richiamato dalla figura, con cui si ottiene un menù delle altre combinazioni disponibili.

Tabella 8.37. Combinazioni di tasti più comuni per comunicare direttamente con il kernel.

Combinazione	Descrizione
[*R Sist AltGr h*]	Richiama una guida rapida a tutti i comandi disponibili.
[*R Sist AltGr r*]	*raw* Richiede di disattivare la modalità *raw*, che di norma viene attivata quando si utilizzano programmi grafici. Se si dà questo comando quando tale modalità non è attiva, non si provocano inconvenienti.
[*R Sist AltGr s*]	*sync* Richiede di completare le operazioni di scrittura.
[*R Sist AltGr u*]	*unmount* Richiede di reinnestare il file system in sola lettura.
[*R Sist AltGr b*]	*reboot* Richiede di riavviare il sistema.
[*R Sist AltGr i*]	Richiede di eliminare tutti i processi, a esclusione di '**`init`**'.
[*R Sist AltGr k*]	*kill* Richiede di eliminare tutti i processi attivi sulla console da cui si dà il comando.
[*R Sist AltGr f*]	Richiede di eliminare il processo elaborativo che risulta essere quello che utilizza più risorse.

La combinazione di tasti [*R Sist AltGr r*] serve per riprendere il controllo di una console quando si è creato un problema con un programma grafico che utilizza la tastiera in modo diretto (*raw*). Per riavviare il sistema, conviene procedere con i comandi: [*R Sist AltGr s*], [*R Sist AltGr u*] e [*R Sist AltGr b*]. In pratica, in questo modo si completano le operazioni di scrittura nelle unità di memorizzazione; si passa a un utilizzo in sola lettura e quindi si riavvia il sistema.

Riquadro 8.38. Errori comuni segnalati dal kernel Linux

Errore	Annotazioni
`Kernel panic: No init found. Try passing init= option to kernel.`	Il kernel non è in grado di avviare il programma '**`/sbin/init`**' o qualunque altro programma indicato con il parametro '**`init=`**…'. Per risolvere il problema si può provare a utilizzare il parametro '**`init=`**…' (eventualmente indicando un programma differente, come '**`/bin/sh`**'), oppure, nel caso il programma esista ci si deve accertare del motivo per cui questo non viene avviato. Il mancato funzionamento del programma che svolge il ruolo di Init può dipendere dalle librerie che non sono corrette o che non sono state collegate correttamente con il file '`/etc/ld.so.cache`'; in modo particolare, è necessario che le librerie siano accessibili in lettura e anche eseguibili.
`Warning: unable to open an initial console.`	Il kernel non è in grado di aprire il file di dispositivo '`/dev/console`'. Probabilmente la directory '`/dev/`' è vuota e si prevede l'uso di un sistema come uDev, che però non ha ancora potuto creare i file di dispositivo. Per risolvere il problema occorre predisporre dei file di dispositivo di uso generale, che poi possono essere rimpiazzati da eventuali sistemi automatici per la loro gestione.

8.12 Riferimenti

- *The Linux kernel archive*, http://www.kernel.org/
- Linus Torvalds, */usr/src/linux symlink*, http://linuxmafia.com/faq/Kernel/usr-src-linux-symlink.html
- The Answer Gang, */usr/src/linux symlink considered harmful*, Linuxgazette, n. 62, http://linuxgazette.net/issue62/tag/4.html
- Sorgenti del kernel: le informazioni più aggiornate sull'uso dei parametri di avvio si possono trovare all'interno degli stessi sorgenti del kernel, nelle directory successive a '***sorgenti_linux***`/drivers/`' e all'interno di '***sorgenti_linux***`/Documentation/`' (può trattarsi dei commenti iniziali ai file dei sorgenti, o file di testo separati).
- Pagina di manuale *bootparam(7)*

- *grml - Linux for texttool-users and sysadmins, GrmlWiki [[hardware]],* http://wiki.grml.org/doku.php?id=hardware
- Werner Almesberger, Hans Lermen, *Using the initial RAM disk (initrd)*, '***sorgenti_linux***`/Documentation/initrd.txt`'

[1] Al posto di '**`make menuconfig`**' si possono usare dei comandi alternativi; eventualmente si può avere un elenco completo delle funzionalità del file-make con il comando '**`make help`**'.

[2] La sigla '**`bzImage`**' sta per *big zImage*, a indicare che si tratta di un'estensione del formato «zImage».

[3] Il termine *loop device* usato qui, non deve essere confuso con *loopback device* usato nella configurazione dei servizi di rete.

[4] **util-linux: pivot_root** GNU GPL

[5] **util-linux: blockdev** GNU GPL

[6] **Initrd tools** GNU GPL