GNU/Linux Intermedio

Sesta lezione:

KERNEL E DEVICES

Il kernel

- Abbiamo piu volte parlato del “kernel” di Linux, ma non abbiamo mai realmente approfondito il discorso.
- Il kernel è il “nucleo” del sistema operativo, come abbiamo detto. E' quel componente software che si occupa di far dialogare gli applicativi con l'hardware, oppure tra di loro.
- Il kernel di Linux è monolitico, ovvero composto di un solo file.
- In realtà per questioni di comodità, scalabilità e flessibilità, il kernel si avvale di “componenti esterne” chiamate “moduli”, tra i quali sono inclusi i “device drivers”.
- I moduli vengono “caricati” e “scaricati” a “runtime”, durante l'esecuzione, a volte anche automaticamente.

Compilare i moduli

- Quasi tutte le funzionalità del kernel sono “modularizzabili”.
- Quando si configura il kernel (vedremo passo passo), per ogni funzionalità [che lo prevede] è possibile scegliere se:
  - compilarla come “modulo”, e quindi fare in modo che questa venga “caricata” solo quando necessario
  - compilarla “staticamente”, includendola di fatto nel codice binario del kernel.
- Spesso ci si trova di fronte alla convinzione che compilare quanto piu possibile staticamente sia meglio.
- Questo è sicuramente vero in un ambito server, dove la possibilità di caricare moduli può rappresentare un pericolo nel momento in cui il sistema viene violato, e quindi si disabilita questa funzionalità.

Statico o modulare?

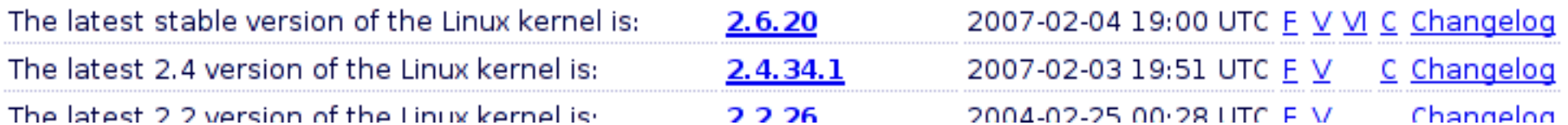
- Fatta eccezione per questa particolare esigenza, non c'è una seria ragione per cui scegliere un'opzione piuttosto che un'altra. Ove possibile, si preferiscono i LKM perchè:
  - sono veloci tanto quanto i componenti compilati (l'operazione di caricamento è molto veloce e viene fatta una volta sola)
  - possono essere scaricati, liberando spazio in RAM
  - vengono caricati solo una volta che il kernel è pronto, e quindi consentono di identificare un problema grave su una certa funzionalità con piu facilità
  - Ricompilarli (perchè dobbiamo aggiungerne uno che non avevamo previsto) è meno oneroso che ricompilare l'intero kernel
- I moduli sono strettamente legati al kernel, e vanno ricompilato ogni volta che lo si cambia.

I moduli

- Possiamo distinguere due diverse “categorie” di moduli:
  - Moduli inclusi nel kernel “vanilla”
    - Sono quelli giudicati importanti da Linus Torvalds e quindi distribuiti insieme al sorgente del kernel
    - Sono necessariamente rilasciati con la stessa licenza del kernel (GPLv2 attualmente)
  - Moduli terzi
    - Si tratta soprattutto di device drivers
    - Servono ad interfacciarsi con hardware non cosi diffusi da giustificare la loro integrazione nel kernel (madwifi)
    - Oppure sono driver proprietari (i moduli video ufficiali di Ati e Nvidia)
    - Se sono distribuiti con il sorgente, possono essere compilati anche loro staticamente nel kernel

Scaricare il kernel

- La scelta della modalità in cui compilare le varie funzionalità del kernel, viene fatta al mento della configurazione.
- Per recupare una copia “vanilla” del kernel di Linux, ci connettiamo al sito web http://www.kernel.org



The latest stable version of the Linux kernel is:	2.6.20	2007-02-04 19:00 UTC	F	V	VI	C	Changelog
The latest 2.4 version of the Linux kernel is:	2.4.34.1	2007-02-03 19:51 UTC	F	V		C	Changelog
The latest 2.2 version of the Linux kernel is:	2.2.26	2004-02-25 00:28 UTC	F	V			Changelog

- La versione “completa”, è accessibile tramite il link “F” (Full).
- Attualmente (2.6.20) la dimensione del file .bz2 che scarichiamo è di 41.4 Mb.
- La versione 2.4.31.1 invece, “pesa” 29.5 Mb.

Estrarre

- Una volta scaricato il pacchetto compresso contenente il kernel, dovremo estrarlo.
- Per motivi di ordine, la regola è quella di tenere copia dei sorgenti del kernel nella cartella `/usr/src/linux`, che solitamente è un link simbolico a `/usr/src/linux-${VER}` dove `${VER}` è il numero di versione del kernel.
- Per rispettare questo ordine, dovremo quindi fare:

```
# Ci spostiamo nella directory...
cd /usr/src
# Estraiamo il sorgente...
tar -xjf /path/to/kernel/package/linux-${VER}.tar.bz2
# Eliminiamo il vecchio link simbolico e lo rifacciamo...
rm linux
ln -s linux-${VER} linux
```

Configurare il kernel

- A questo punto, il sorgente è pronto per essere configurato e successivamente compilato.
- Per la configurazione, che col passare degli anni è diventata piuttosto complessa, gli sviluppatori ci mettono a disposizione 6 diverse modalità:
  - config
    - Domanda dopo domanda, ci viene chiesto di decidere quali funzionalità includere. Puro testo.
  - menuconfig
    - La versione standard del configuratore. Basata su un'interfaccia Ncurses
  - gconfig
    - Interfaccia grafica, basata sulle librerie GTK
  - xconfig

Configurare il kernel

- A questo punto, il sorgente è pronto per essere configurato e successivamente compilato.
- Per la configurazione, che col passare degli anni è diventata piuttosto complessa, gli sviluppatori ci mettono a disposizione 5 diverse modalità:
  - kconfig
    - Interfaccia grafica, basata sulle librerie QT
  - oldconfig
    - Vale come per “config”, ma ci vengono poste solo le domande relative alle funzionalità introdotte con la nuova versione del kernel.
    - Il resto viene ricavato automaticamente dal file .config del precedente kernel (appositamente copiato)
    - Utile quando si aggiorna il kernel ad una nuova versione

Configurare il kernel

- La configurazione del kernel 2.6 si suddivide in 16 sezioni
  - Code maturity level options
    - Contiene una sola opzione di configurazione, che definisce se deve o meno includere il codice “in sviluppo” (EXPERIMENTAL) nelle altre sezioni.
    - Possiamo tranquillamente dirgli di si
  - General setup
    - Contiene tutte quelle voci di configurazione che riguardano il setup generale del kernel, quali il supporto per la swap, o la possibilità di configurarlo per girare su sistemi “embedded”
  - Loadable module support
    - Contiene la configurazione relativa alla gestione dei moduli: se abilitarli, se caricarli automaticamente, introdurre la possibilità di “scaricarli”.

Configurare il kernel

- La configurazione del kernel 2.6 si suddivide in 16 sezioni
  - Block layer
    - Consente di attivare il supporto per device (o files) di oltre 2TB di spazio, e di decidere quali I/O Scheduler attivare.
  - Processor type and features
    - E' la prima sezione davvero interessante.
      - Symmatric multi-processing support
        - Abilita il supporto per piu processori/core (utile anche per gestire HyperThreading)
      - Processor Family
        - Consente di scegliere la “famiglia” del processore.
        - Pentium-4/Celeron(P4-based)/Penium-4 M/Xeon)
      - SMT (Hyperthreading) scheduler support
        - Supporto specifico per l'HT dei Pentium-4

Configurare il kernel

- La configurazione del kernel 2.6 si suddivide in 16 sezioni
  - Processor type and features
    - Preemption Model
      - Consente di definire se il kernel deve essere “Preemtible” (risposta con bassa latenza) o meno. Sui server, non si abilita, sui desktop, si.
    - High Memory Support
      - Abilita il supporto per RAM a 4 o 64GB
  - Power management options
    - Gestisce il “Power management”, nelle due diverse versionie:
      - ACPI
        - Se abilitarlo, con quali sensori
      - APM
        - Se abilitarlo, con quali configurazioni

Configurare il kernel

- La configurazione del kernel 2.6 si suddivide in 16 sezioni
  - Power management options
    - Contiene anche la parte di configurazione del frequency scaling.
      - Consente di abilitarlo, di definire quale “Governor” utilizzare tra:
        - Performance
        - Powersave
        - Userspace
        - Ondemand
        - Conservative
      - Quale usare di default
      - Per quale tipo di processore

Configurare il kernel

- La configurazione del kernel 2.6 si suddivide in 16 sezioni
  - Bus options (PCI, PCMCIA, EISA, MCA, ISA)
    - Abilita il supporto per i diversi tipi di Bus
  - Executable file formats
    - Quali formati eseguibili abilitare: solitamente ELF e basta
    - Includere MISC se si vogliono utilizzare wrapper per i driver di altri sistemi operativi
  - Networking
    - Oltre al supporto per le schede di rete (wireless e wired, quali modelli), contiene anche la configurazione delle reti Bluetooth e IRDA, il routing e tunneling
    - Particolarmente corposa questa sezione, è da compilare con molta attenzione

Configurare il kernel

- La configurazione del kernel 2.6 si suddivide in 16 sezioni
  - Device drivers
    - Quali tipi di device supportare.
    - Che tipi, quali modelli, spaziando dalle device audio, alle device video, ai cdrom, USB e Firewire, ISDN, telefonia... davvero di tutto
  - Filesystems
    - Quali tipologie di filesystem abilitare
    - Abilitare particolari opzioni, come la gestione dei “quota”, o l'automounter del kernel (montaggio di filesystems remoti “on demand”)
  - Instrumentation support
  - Kernel hacking
  - Security options

Configurare il kernel

- La configurazione del kernel 2.6 si suddivide in 16 sezioni
  - Cryptographic options
    - Quali algoritmi di hash e cifratura implementare a livello kernel
  - Library routines
    - CRC16, 32, 32c
- Per compilare staticamente una voce, premere il tasto Y
- Per compilare modulare una volce, premere il tasto M
- Per non includere una funzionalità, premere N
- Il risultato della configurazione è un file nella directory corrente (.config) che contiene tutta la nostra configurazione.
- E' possibile caricare altri .config a partire dal configuratore

Compilare il kernel

- Una volta configurato, possiamo passare alla compilazione vera e propria del kernel.

```
make
```

- Una volta terminata la compilazione dell'immagine del kernel, compiliamo i moduli...

```
make modules
```

- E li installiamo nella loro directory (`/lib/modules/${VER}/`)

```
make modules_install
```

- Infine copiamo l'immagine del kernel nella /boot

```
cp arch/i386/boot/bzImage /boot/vmlinux-${VER}
```

- E' buona norma copiare anche i files `.config` e `System.map` nella boot allo stesso modo, in modo da poter rivedere la configurazione del kernel in futuro.

Ultime modifiche

- A questo punto dovremo fare in modo che il nuovo kernel venga incluso tra quelli avviabili.
- Nel caso in cui non abbiate compilato staticamente il modulo che gestisce il filesystem principale del vostro sistema, dovrete anche rigenerare l'initrd.
- Per fare questo dobbiamo modificare la configurazione del boot-loader e (se usiamo LiLo) rigenerarlo.
- E' una buona idea tenere come “backup” l'ultima configurazione sicuramente funzionante (quella con cui state lavorando) in modo da consentire di riavvire il sistema nel caso in cui qualcosa sia andato storto.
- A questo punto riavviate il sistema e selezionate il nuovo kernel nel menu del boot-loader.

Gestire i moduli

- Una volta che il sistema è avviato e funzionante, parte dei moduli saranno stati automaticamente caricati dal kernel durante la fase di riconoscimento dell'hardware.
- Potete però sempre aggiungere o rimuovere moduli utilizzando alcune applicazioni:
  - `insmod ${nomemodulo}`
    - Inserisce il modulo, senza curarsi delle dipendenze
  - `modprobe ${nomemodulo}`
    - Cerca di inserire il modulo, verificando (ed eventualmente caricando) le relative dipendenze
  - `rmmod ${nomemodulo}`
    - Scarica il modulo.

Gestire i moduli

- Ci sono altri comandi estremamente utili quando lavoriamo con i moduli:
  - `lsmod`
    - Restituisce l'elenco dei moduli caricati
  - `modprobe -l`
    - Restituisce l'elenco dei moduli disponibili
  - `modinfo ${nomemodulo}`
    - Restituisce informazioni relative al modulo (autore, descrizione, dipendenze, configurazioni...)
  - `depmod`
    - Aggiorna l'elenco delle dipendenze tra i vari moduli, in modo da rendere piu agevole il caricamento degli stessi.

Le devices

- Alcuni moduli (quelli detti “device drivers”), hanno il compito di “tradurre” le richieste del sistema operativo in comandi comprensibili all'hardware che gestiscono.
- Alcuni di questi moduli, “registrano” dei files, detti “device nodes” o “special files”, che prendono il nome di “devices”, che verranno visualizzati all'interno della directory `/dev`.
- Esistono 3 diversi tipi di “devices nodes”:
  - Devices a blocchi
    - Sono device che “parlano” in blocchi, e possono quindi ospitare un filesystem.
    - Allocano un buffer di input e output
    - Tutte le le periferiche di massa: cdrom, dvd, pendrive usb, hard-disks...

Le devices

- Alcuni moduli (quelli detti “device drivers”), hanno il compito di “tradurre” le richieste del sistema operativo in comandi comprensibili all'hardware che gestiscono.
- Alcuni di questi moduli, “registrano” dei files, detti “device nodes” o “special files”, che prendono il nome di “devices”, che verranno visualizzati all'interno della directory `/dev`.
- Esistono 3 diversi tipi di “devices nodes”:
  - Devices a caratteri
    - Sono le piu diffuse.
    - “Parlano” ed “ascoltano” a caratteri, come fossero dei files di testo e non hanno buffers. I caratteri vengono letti e scritti direttamente sulla device.
    - Mouse, periferiche video, audio...

Le devices

- Alcuni moduli (quelli detti “device drivers”), hanno il compito di “tradurre” le richieste del sistema operativo in comandi comprensibili all'hardware che gestiscono.
- Alcuni di questi moduli, “registrano” dei files, detti “device nodes” o “special files”, che prendono il nome di “devices”, che verranno visualizzati all'interno della directory `/dev`.
- Esistono 3 diversi tipi di “devices nodes”:
  - Pseudo-device
    - Sono device che non si riferiscono ad alcuna periferica fisica.
    - Offorno funzionalità utili gestite direttamente dal kernel
    - urandom, null, zero...

Udev

- A partire dalla release 2.6 del kernel Linux, la gestione dei “device nodes” è stata un po alla volta migrata ad un nuovo “device manager” chiamato UDEV.
- Udev è il successore di devfs e di hotplug, il che significa che si occupa sia di gestire le device a livello di interfacciamento con il kernel, sia di gestire le modifiche e le assegnazioni che vanno fatte quando una device viene collegata e scollegata.
- L'effetto piu tangibile dell'introduzione di Udev è la “dinamicità” della directory `/dev`.
- Storicamente infatti, i files contenuti nella directory `/dev` erano statici, e veniva loro associata una device quando questa veniva collegata.
- Con udev invece, i files vengono creati quando necessario

Udev

- In realtà le principali innovazioni introdotte da Udev stanno “sotto la carrozzeria”, ed in particolare nel suo file di configurazione
  - Udev supporta infatti il cosi detto “persistent naming” che può rendere i nomi delle device indipendenti dall'ordine con cui vengono inserite.
  - Udev viene inoltre eseguito “come un programma” invece che come un modulo del kernel (come avveniva con devfs) ed il nome delle device può essere assegnato avvalendosi anche di programmi esterni, cosa che non poteva avvenire prima, quando il nome veniva deciso dal kernel stesso.
- Si compone di 1 libreria (namedev, che si occupa di assegnare i nomi alle devices) e di un demone (udevd, che crea i files in `/dev`)

Le device “virtuali”

- Non tutti i device drivers però registrano un “device node” per instaurare una comunicazione tra il kernel ed il sistema.
- Ad esempio le periferiche di rete utilizzano device drivers per le quali non esiste alcun file sotto `/dev`.
- Si tratta di device “virtuali” con le quali ci si interfaccia tramite apposite applicazioni e interfacce del kernel.
- Le regole che devono essere applicate alla creazione delle devices, vengono memorizzate in files nella directory `/etc/udev/rules.d/`.
- La regola qui sotto serve a creare un link “/dev/pilot” al momento della connessione di un palmare Palm

```
SUBSYSTEMS=="usb", KERNEL=="ttyUSB*",
  ATTRS{product}=="Palm Handheld*", SYMLINK+="pilot"
```

Il filesystem Proc

- Alcune informazioni sul sistema, sul suo funzionamento corrente, non sono disponibili tramite “device drivers”.
- Nessun device driver infatti ci fornisce informazioni sulla quantità di memoria utilizzata dal sistema, o sul numero di processi attualmente in esecuzione.
- Queste informazioni vanno chieste direttamente al kernel
- Per fare ciò è stata creata un'apposita interfaccia di “comunicazione”, che assume le sembianze di uno pseudo-filesystem, che viene montato all'avvio sul mountpoint `/proc`.
- Si tratta di un filesystem che in realtà non esiste, ma viene generato dinamicamente dal kernel a fronte delle nostre richieste.

Il filesystem Proc

- Al suo interno troviamo:
  - Una serie di cartelle numerate.
    - Si tratta dei processi, ed il numero che viene assegnato alla directory è il loro PID.
    - All'interno di ogni cartella troviamo una serie di files che ci consentono di accedere alle informazioni che ci possono essere utili riguardo i processi:
      - La riga di comando che ha avviato il processo, comprensiva quindi di tutti i parametri
      - L'environment esistente al momento del suo avvio
      - Le statistiche di utilizzo del processo (tempo di esecuzione, tempo d'attesa...
      - Una cartella contenente l'elenco dei files che ha aperto
      - ...

Il filesystem Proc

- Al suo interno troviamo:
  - La cartella `bus`, che contiene sottocartelle dedicate ai vari bus del sistema (`/proc/bus/usb`, `/proc/bus/pci`, ...)
  - Il file `cmdline` che contiene la “chiamata” del kernel, con tutti i parametri passati (solitamente dal boot-loader)
  - Il file `cpuinfo` che contiene informazioni relative al/ai processori (frequenza, numero, bogomips, cache L2...)
  - Il file `filesystems` che contiene l'elenco dei filesystems supportati dal kernel in esecuzione (anche tramite i moduli)
  - Una cartella `ide` che contiene dati relativi ai dischi (cache, capacità, geometria, supporto “smart”, ...)
  - Il file `interrupts` che contiene i valori ed alcune statistiche relative agli interrupts
  - Il file `iomem` che contiene la mappatura degli indirizzi di I/O

Il filesystem Proc

- Al suo interno troviamo:
  - Il file `ioports` che contiene le porte di I/O
  - Il file `kcore` che ritorna la memoria RAM del kernel (!!!)
  - Il file `meminfo` che contiene informazioni sulla memoria di sistema utilizzata (RAM e swap)
  - Il file `mounts` che contiene informazioni sui filesystems montati
  - Il file `modules` che contiene l'elenco dei moduli montati, il loro utilizzo e le loro dipendenze
  - La directory `net` che contiene informazioni sulla rete (spaziando dalla ARP table alla configurazione delle interfacce di rete ethernet)
  - Il file `swaps` che contiene l'elenco degli swap attivi
  - ...