Corso Base su Linux

Basato su Fedora 7
Lezione 2

File system

Per file system si intende l'astrazione (metodi e protocolli) con cui si organizzano i file su un supporto fisico ad accesso casuale (floppy, cdrom, hard-disk, ecc)

Le informazioni riguardanti un oggetto (file o directory) di un file system sono contenute in un **inode**, che viene identificato da un numero progressivo e descrive le caratteristiche base di un oggetto: permessi, data di modifica, tipo, posizione ecc.

Un sistema Linux, come ogni sistema Unix, ha una directory principale, chiamata **root** ed indicata con **/** sotto la quale si trovano TUTTE le altre directory e TUTTI gli altri file system eventualmente montati sul sistema (floppy, cdrom, condivisione di rete ecc.).

Il principio è radicalmente diverso da quello presente nel mondo Windows, dove ogni device o risorsa ha un suo nome o lettera identificativa al cui interno si trovano le directory del relativo file system.

Il file system usato in Linux è l' **ext2**
Dal kernel 2.4.x è disponibile un'evoluzione dell'ext2, l'**ext3** che ha il vantaggio di essere un journal file system

Organizzazione del file system

/	Radice (root)
/root	Home dell'utente root
/boot	Contiene tutte le immagini del kernel e file indispensabili al bootstrap
/etc	Contiene i file di configurazione del sistema e dei programmi installati
/home	Contiene le home degli utenti tranne quella di root
/usr	Contiene binari, documentazione, librerie e sorgenti della maggior parte dei programmi (e i sorgenti del kernel)
/var	File contenenti informazioni dinamiche (log, pid file, directory di spool).
/proc	File system virtuale. Contiene, come se fossero file e directory, dati dinamici sul sistema e sui processi
/dev	Contiene i file per la gestione dei dispositivi sia a blocchi (hdd) che a carattere (tty) oltre a file speciali (/dev/null)

Organizzazione del file system

/sbin	Contiene comandi e programmi riservati a root (altri comandi sono in /usr/sbin/)
/bin	Contiene comandi e programm**i** base per tutti gli utenti (altri comandi sono in /usr/bin/)
/lib	File delle *librerie* di sistema utilizzate dai programmi
/tmp	Contiene i file temporanei
/usr/tmp	Altra directory che contiene file temporanei
/usr/doc	Documentazione sul sistema
/mnt	Directory sotto la quale vengono montati altri file system (floppy, cdrom, chiavi USB, oartizioni NTFS, ecc)
/media	Come /mnt si trova in diverse distribuzioni, ma non fa parte della struttura standard (è usata dal demone automount).

Montaggio del file system

Prima di poter utilizzare un file system (es: CDROM, floppy, tape, condivisione di rete windows, directory nfs, partizione fat32 di un hard disk, ecc) questo deve essere formattato e montato in una sotto directory della root (/).

Una volta montato il file system risulta accessibile a programmi e utenti in modo trasparente e diventa parte integrante dell'albero delle directory sotto /

Dopo l'uso il file system può essere smontato (operazione necessaria per espellere un CDROM, un floppy o una chiave USB).

La directory su cui viene montato un file system (mount point) può anche non essere vuota, ma nel momento in cui ci viene montato un file system, i dati ivi contenuti non sono più visibili fino a quando non si smonta il file system.

Per montare un file system si usa il comando ***mount.***

Sintassi:

mount -t [tipo fs] [opzioni] device mount_point

esempio:

$ mount -t ntfs -o r /dev/sda1 /mnt/windows

Mount

mount -t [tipo fs] [opzioni] device directory

[tipo fs] è il tipo di file system da montare.

Linux riconosce un elevato numero di file system:
adfs, affs, autofs, cifs, coda, coherent, cramfs, debugfs, devpts, efs, ext, ext2, ext3, hfs, hpfs, iso9660, jfs, minix, msdos, ncpfs, nfs, nfs4, ntfs, proc, qnx4, ramfs, reiserfs, romfs, smbfs, sysv, tmpfs, udf, ufs, umsdos, usbfs, vfat, xenix, xfs, xiafs.

Il parametro [-t tipo_fs] è un parametro opzionale, se non viene specificato il comando mount tenterà di individuare il file system in automatico.

Qualora il file system non venisse riconosciuto, il comando terminerebbe con un errore e bisogna ripeterlo specificando manualmente il tipo di file system.

Per montare tutti i file system indicati nel file /etc/fstab si usa il prametro **'-a'**

```
$ mount -a
```

Per ulteriori informazioni consultare la pagina del manuale:

```
$ man mount
```

Comandi per i file system

Altri comandi utili nella gestione dei file system sono:

umount [opzioni] device	Smonta un dispositivo (necessario)
df [opzioni] [file]	Verifica lo spazio libero su disco
du [opzioni] [file]	Visualizza lo spazio occupato da file e directory
fsck [opzioni] dispositivo	Verifica l'integrità e ripara il file system
mkfs [opzioni] dispositivo	Crea un file system (formatta)
fdformat device	Formatta a basso livello un floppy

Si ricordi che un file system è individuato con il device corrispondente:

/dev/hda1	Prima partizione del primo disco IDE
/dev/sda2	Seconda partizione del primo disco SCSI
/dev/fd0	Primo floppy disk

FSTAB

Nel file **/etc/fstab** vengono configurate le informazioni sui vari file system (da montare al boot o no) preimpostati sul sistema.

Il suo formato prevede per ogni riga le seguenti informazioni:

1- Dispositivo da montare, si può indicare il device corrispondente (/dev/hda1) o l'etichetta del file system (ext2 o xfs) preceduta dalla keyword LABEL.
2- Mount point sul file system principale
3- file system Type da utilizzare (es: ext2, ext3, iso9660, nfs...)
4- Opzioni specifiche per il mount:
   defaults= monta il dispositivo al boot
   noauto = montaggio solo su richiesta dell'utente
   user = permette il montaggio anche agli utenti
5- Indica se deve essere eseguito il backup usando il comando *dump* (0=NO).
6- Controllo del file system al boot. Uno 0 indica NESSUN CHECK.
   Il controllo viene eseguito in ordine numerico crescente.

```
#/etc/fstab
LABEL=/          /          ext3       defaults      1 1
LABEL=/boot1     /boot      ext3       defaults      1 2
/dev/cdrom       /mnt/cdrom iso9660    noauto, user  0 0
proc             /proc      proc       defaults      0 0
/dev/sda2        none       swap       defaults      0 0
```

LINK

Il comando **ln** crea un link (alias) di un file o directory.

Quando si opera sul link è come se si operasse direttamente sul file tranne che quando si rimuove il link (non si cancella il file).

Hard Link

Per default ln crea un hard link: di fatto un altro nome per un file esistente.
Originale e link sono indistinguibili, condividono lo stesso inode.
Non si possono creare hard link fra file system diversi e su directory.

```
$ ln [opzioni] nome sorgente [nome destinazione]
```

Link symbolico

Detto anche soft link o symlink, si ottiene specificando l'opzione **"-s".**
E' un tipo speciale di file che si riferisce ad un file (inode) diverso
Può essere creato su file system diversi (nfs, cdrom ecc.) e su file che non esistono. Si capisce di quale file sono un link.

```
$ ln -s nome file [nome link]
```

L'uso dei link è completamente trasparente e NON impatta sulle performance del sistema.

Utenti e Gruppi

UTENTI

Si possono creare più account che vengono memorizzati nel file **/etc/passwd**.
Ogni riga del file passwd è del tipo:

username:passwd:UID:GID:user_data:home_dir:shell

UID =	User IDentifier è un valore numerico univoco
GID =	Group Identifier è un valore numerico univoco
user_data =	Nome e Cognome dell'utente, telefono, ecc...
home_dir =	directory di lavoro dell'utente
shell =	shell che utilizzerà l'utente

es: root:passwd:0:0:root:/root:/bin/bash

GRUPPI

Linux gestisce anche gruppi di utenti, per condividere file e per favorire la collaborazione. Ogni utente deve appartenere ad almeno un gruppo.

I gruppi sono definiti nel file **/etc/group**, ogni riga ha la forma del tipo:

group_name:passwd:GID:users_list

es: adm:passwd:4:root, adm, daemon

Attributi di file e directory

Ogni file appartiene ad un utente e a un gruppo.
Per visualizzare utente e gruppo di un file si usa il comando "**ls -l**".

```
$ ls -l /etc/passwd
-rw-r--r-- 1 root adm   77266   Dec 13 17:18     /etc/passwd
```

In questo caso l'utente proprietario è **root**, mentre il gruppo è **adm**.

E' possibile cambiare (avendo i permessi) l'utente ed il gruppo con i comandi:
chown (change owner) e chgrp (change group)

Sintassi: chown nome_utente nome_file
es:

```
$ chown topolino /etc/passwd
-rw-r--r-- 1 topolino adm   77266   Dec 13 17:18     /etc/passwd
```

Sintassi: chgrp nome_gruppo nome_file
es:

```
$ chown pluto /etc/passwd
-rw-r--r-- 1 topolino pluto   77266   Dec 13 17:18     /etc/passwd
```

Al posto del nome utente e del gruppo è possibile specificare i rispettivi UID e GID.

Permessi su file e directory

Ogni file dispone di permessi separati per tre categorie di utenti:

- utente proprietario
- gruppo proprietario
- il resto degli utenti

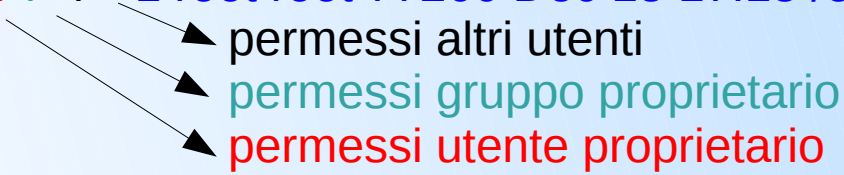
Per ogni categoria esistono tre tipi di permessi:

- lettura definito dal flag r (Read)
- scrittura definito dal flag w (Write)
- esecuzione definito dal flag x (eXecution)

Per le directory questi flag hanno un significato diverso:

- lettura = poter visualizzare i file contenuti
- scrittura = creare, copiare o spostare i file contenuti
- esecuzione = poter usare il nome della directory in un PATH

```
$ ls -l /etc/passwd
-rw-r--r-- 1 root root 77266 Dec 13 17:18 /etc/passwd
```



Modifica dei permessi su file e directory

Per modificare le modalità di accesso ad un file si usa il comando:
chmod (change mode)

Sintassi: chmod operazione nome_file

L'operazione a sua volta è composta da: categoria , azione e permesso

Carattere	Categoria
u	utente proprietario del file
g	gruppo proprietario del file
o	tutti gli altri utenti del sistema
a	tutti gli utenti; equivale a 'ugo'

Operazione	Azione
+	Aggiunge permessi ai permessi esistenti
-	Rimuove permessi dai permessi esistenti
=	Assegna i permessi al file

Permesso: r,w,x hanno il significato visto in precedenza

Uso di chmod

Proteggere un file dalla scrittura

$ chmod go-w cruise

Rendere un file privato

$ chmod go= cruise

Rendere un file pubblico

$ chmod a+rw cruise

Rendere un file pubblico

$ chmod a+x myscript

IMPORTANTE!!
Il superutente, **root**, può sempre accedere a qualsiasi file presente sul sistema, indipendentemente dai suoi permessi di accesso.

NOTA
Di fatto rwx, sono la rappresentazione dello stato di tre bit, partendo dal più significativo si ha: r = 4, w=2, x= 1
pertanto i comandi chmod possono essere impartiti anche in forma numerica:

$ chmod 700 myscript abilita tutti i permessi solo all'owner (file privato)

Comandi di uso frequente

$ touch nome file [INVIO]
aggiorna la data e l'ora di un file, se il file non esiste crea un file vuoto

$ mkdir nome directory [INVIO]
crea una directory

$ mkdir –p work/completed/2001 [INVIO]
crea la directory 2001 e le directory superiori se non esistono.

$ cd nome directory [INVIO]
cambia directory

$ cd [INVIO]
rende come corrente la directory home

$ pwd [INVIO]
visualizza il PATH in cui ci si trova

$ ls -R [INVIO]
visualizza il contenuto della directory corrente e delle sotto directory.

Comandi di uso frequente

$ cp file_sorgente file_destinazione [INVIO]
copia il file_sorgente creando il nuovo file_destinazione

$ mv file_sorgente file_°destinazione [INVIO]
rinomina o sposta il file_sorgente in file_destinazione

$ ls -a
visualizza i file nascosti quelli in cui il primo carattere del nome è un punto (.)

E' possibile utilizzare dei caratteri speciali all'interno del nome dei file:

* Corrisponde ad una serie di nessuno o più caratteri.
* da solo indica tutti i file
*.txt indica i soli file con estensione txt

? Simile ad * ma sostituisce un generico carattere
cas? equivale a tutti i file di 4 caratteri la aventi le prime tre uguali a cas

~ rappresenta la home directory dell'utente
~ = /home/pippo
cd ~/prova equivale al comando: cd /home/pippo/prova

Comandi di uso frequente

$ find percorso -name nome_file [INVIO]
visualizza tutti i file che si trovano sotto “percorso” aventi “nome_file”.

$ find /tmp -size +10000k [INVIO]
visualizza i file memorizzati sotto /tmp aventi dimensione maggiore di 10000k

$ find /home -user topolino [INVIO]
visualizza i file memorizzati sotto /home di proprietà di topolino

$ which programma [INVIO]
visualizza il percorso completo di dove si trova il comando

$ grep nome [INVIO]
filtra le righe che contengono la parola “nome”

$ more programma [INVIO]
interrompe la visualizzazione quando si riempie lo schermo ed attende la pressione di un tasto per proseguire

$ man comando [INVIO]
visualizza la pagina del manuale relativa a “comando”

Il processo di boot

Il processo di boot di una macchina Linux comporta diverse fasi:

1- All'accensione il BIOS (Basic Input Output System) memorizzato su memoria ROM non volatile, definisce l'ordine dei device da usare per effettuare il boot.

2- Il BOOT SECTOR del primo device di boot contiene il codice (o i riferimenti su dove trovarlo) del ***loader*** che esegue il bootstrap del sistema operativo.

3- Il loader lancia il caricamento del kernel di Linux, che *si copia in memoria* ed esegue i controlli e il riconoscimento dell'hardware presente.

4- A fine caricamento il kernel esegue il processo init, padre di tutti i processi, che gestisce il caricamento di tutti gli altri programmi da eseguire per completare il boot

Il BIOS

Ogni sistema Intel ha sulla motherboard un BIOS su ROM (ormai su memorie FLASH) con cui gestire l'hardware del sistema.

All'avvio di un computer non c'è nulla in RAM e nessun programma predefinito da caricare.
Le istruzioni su come procedere sono nella memoria non volatile del BIOS, in cui, fra le impostazioni definibili dall'utente, c'e' la sequenza dei dispositivi da usare per il boot.

Nei BIOS più recenti è possibile effettuare il boot da floppy, cdrom, dvd, hard disk (potendo scegliere se dare precedenza a HD IDE, SATA o SCSI) e altri dispositivi quali chiavi USB, Zip o scheda di rete.

Il BIOS cerca, nell'ordine configurato, il boot sector sui diversi dispositivi di boot previsti.

Gli hard disk hanno un boot sector per ogni partizione e un unico Master Boot Record (MBR) che è il boot sector dell'intero hard disk.
Se si esegue il boot da un hard disk, è il codice contenuto nel MBR che viene eseguito

I Linux Loader

Esistono diversi loader che eseguono il bootstrap del sistema operativo Linux:

Su sistemi Intel Based: LILO, GRUB, LOADLIN, SYSLINUX, BOOTLIN;
su sistemi Alpha si usa MILO, mentre su sistemi Sparc SILO.

Tutti svolgono la stessa funzione. Loadlin e Syslinux sono programmi DOS che eseguono il kernel caricandolo da una partizione DOS, altri sono adattamenti di LILO che riguardano sistemi non basati su processori Intel.

Windows usa come loader VBR (Volume Boot Record) che si occupa di caricare l'NTLDR ed utilizza il file boot.ini per la configurazione.

Kernel

Il kernel, invocato dal loader, viene caricato in memoria ed inizializza i vari device driver, visualizzando vari messaggi utili per capire e conoscere il proprio sistema.

INIT

L'ultima operazione eseguita dal kernel alla fine del suo caricamento è il lancio del processo ***init***, **il padre di tutti i processi**.
Da questo momento tutto il codice eseguito lavora in user space (in kernel space lavorano solo il kernel e i suoi moduli).

LILO

LILO è uno dei ***LI***nux ***LO***ader più diffusi, permette il boot sia di Linux che di altri sistemi operativi.
Generalmente l'installazione di Linux provvede a creare ed installare LILO (se si è scelto di installare il loader direttamente sull'hard disk e non su floppy) ma in caso di kernel upgrade o aggiunta di un nuovo sistema operativo sulla macchina può essere necessario modificare le impostazioni di LILO manualmente.

Tutte le impostazioni di LILO sono definite nel file ***/etc/lilo.conf*** che contiene una parte di configurazione globale e una o più parti relative alle diverse immagini del kernel o sistemi operativi che si vogliono poter caricare.

Il comando ***/sbin/lilo*** installa sul MBR o sul settore di boot di una partizione il LILO secondo le indicazioni date in /etc/lilo.conf.
Al momento del boot LILO inoltre presenta la possibilità di dare comandi vari e di scegliere quale sistema operativo o versione di kernel lanciare.

ATTENZIONE: Operare maldestramente con LILO e il MBR può impedire il boot del sistema operativo, si suggerisce sempre di avere un dischetto di boot disponibile da utilizzare in caso di danni o problemi con l'MBR.

Configurare il LILO

Il file /etc/lilo.conf è diviso in diverse sezioni, analizziamole in dettaglio:

Global Options

prompt	Visualizza un prompt per scegliere il Sistema operativo di cui eseguire il boot e passare dei parametri al kernel.
timeout=50	Definisce il tempo in decimi di secondo che lilo aspetta prima di caricare l'immagine di default.
default=linux	Indica quale sistema operativo caricare di default.
boot=/dev/hda	Indica il device dove installare LILO. /dev/hda1 = boot record della prima partizione
map=/boot/map	Indica la posizione del ***map*** file, che contiene la posizione fisica del kernel in un formato interpretabile da LILO.
install=/boot/boot.b	Definisce il file che il comando /sbin/lilo installa sul settore di boot. E' il binario vero e proprio di LILO.
message=/boot/message	Definisce un file contenente del testo da visualizzare al boot.
linear	Genera indirizzi di settore lineari, invece di indirizzi di settore/head/cilinder, che possono dare problemi nel caso di dischi di grandi capacità. E' consigliabile metterlo.

Configurare il LILO

Label Options per Linux

image=/boot/vmlinuz-2.4.7-10	Indica dove risiede l'immagine del kernel da caricare.
label=linux	Definisce il nome associato a questa immagine, è quello che si può scrivere nel LILO command prompt all'avvio per caricare questa immagine.
initrd=/boot/initrd-2.4.7-10.img	Definisce la posizione dell'immagine dell'initial ram disk. Il kernel usa questo file system come RAM disk, ne esegue il file linuxrc e poi monta il vero file system su Hard Disk. Serve, ad esempio, per installare un kernel modulare su Hard Disk SCSI, per i quali i driver sono disponibili come moduli e non possono essere caricati direttamente.
read-only	Indica al kernel di montare la root, sotto definita, in modo read-only durante la fase di boot. Così facendo è possibile eseguire un file system check durante il boot senza rischi. Prima di passare il controllo delle operazioni al processo init, il kernel monta nuovamente la root in read-write.
root=/dev/hda3	Definisce in quale partizione si trova la root del sistema.

Configurare il LILO

Label Options per Windows

label=Windows_2000	Definisce il nome della label associata a questo OS
other=/dev/hda1	Indica la partizione in cui risiede un sistema operativo diverso da Linux.
optional	Omette l'immagine del kernel se non è disponibile al momento della creazione del map file.

Opzioni aggiuntive

Al boot quando appare la scritta **LILO:**, premendo il tasto [TAB] vengono visualizzate le label disponibili per il boot.
Per passare dei parametri al kernel dal LILO prompt, si usa il comando ***append***
Per una scheda di rete, ad esempio, che usi l'IRQ 10 e l'indirizzo di I/O 210 (potrebbe essere necessario per schede di rete ISA o EISA non PnP):

```
append = "ether=10,0x210,eth0"
```

Opzioni importanti per la sicurezza (utilizzabili sia nell'area global che label) sono:

password=pippo	Indica che è necessaria una password (pippo) per poter dare comandi al prompt di LILO
restricted	Indica che la password serve solo quando si devono passare dei parametri al boot, ma non per scegliere la label da caricare.

GRUB

GRUB è un boot loader multipiattaforma estremamente flessibile e potente.

Ha un propria CLI in cui inserire a mano i parametri di boot o può presentare un'interfaccia a menu configurabile tramite il file /etc/grub.conf.

Per installare grub sul settore di avvio basta dare il comando:

grub-install /dev/hda

Differentemente da LILO non c'è bisogno di ridare il comando ogni volta che si cambia la configurazione.

E' sufficiente modificare il file di configurazione e al successivo riavvio del computer verranno presentate le nuove opzioni.

E' più sicuro poiché si evita di riscrivere il MBR e di avere un boot loader inconsistente a causa di una opzione non corretta.

Utilizzando la CLI di GRUB, disponibile durante la fase di boot, si possono variare i parametri di boot esistenti o specificare un comando di boot completamente nuovo.

Configurazione di GRUB

Segue un esempio di /etc/grub.conf che esegue le stesse operazioni del file /etc/lilo.conf mostrato in precedenza.

default=0	Imposta come default la prima "label" sotto indicata
timeout=10	Imposta a 10 secondi il tempo di attesa prima di caricare automaticamente l'entry di default.
splashimage=(hd0,2)/boot/grub/splash.xpm.gz	Indica dove trovare l'immagine da visualizza nella schermata di boot
password --md5 $1$6ðòüZßXÈ$bXTLL8IbDhnwmjyaNNcPG.	Imposta una password, criptata, da fornire per poter accedere al menu o alla command-line.
title Red Hat Linux (2.4.7-10)	Titolo della prima scelta del menu ("label")
root (hd0,2)	Hard disk (primary master) e partizione (terza) del device di root.
kernel /boot/vmlinuz-2.4.7-10 ro root=/dev/hda3	Path nel quale si trova il kernel
initrd /boot/initrd-2.4.7-10.img	Path del file system da montare su Ram Disk al boot.
title DOS	Il titolo della seconda scelta del menu
rootnoverify (hd0,0)	Hard disk (primary master) e partizione (prima) del device di root. Non tenta di montare la partizione.
chainloader +1	Prova a caricare in cascata il primo blocco della partizione sopra definita.

Kernel

Quando il boot loader esegue il kernel, questo prosegue con il riconoscimento e l'inizializzazione dell'hardware presente.

Per ridurre l'occupazione di memoria su disco il kernel normalmente è compresso, pertanto la prima operazione da eseguire è quella di decomprimerlo in memoria.

Durante il caricamento presenta a video una serie di informazioni, a volte fin troppo dettagliate, sull'hardware trovato.

Per vedere i messaggi del kernel creati durante l'ultimo boot basta digitare il comando dmesg, che visualizza esattamente quanto viene visualizzato dal kernel durante il boot (a volte in tempi troppo rapidi per essere leggibile).

A fine caricamento il kernel lancia il processo ***init*** il padre di tutti i processi.

Messaggi del Kernel

Segue un esempio di un dmesg. Alcune parti variano a seconda dell'hardware presente sul sistema, altre sono sostanzialmente uguali su tutti i sistemi Linux.

Linux version 2.4.13 (root@llocalhost) (gcc version 2.96 20000731 (Red Hat Linux 7.1 2.96-81)) #5 Fri Nov 9 16:36:50 CET 2001
Mostra le versioni del kernel (2.4.13), del compilatore interno (gcc) e del sistema operativo

Calibrating delay loop... 666.82 BogoMIPS
Test per verificare la velocità del processore. Valori più alti indicano prestazioni migliori.

Memory: 62272k/65536k available (1091k kernel code, 2880k reserved, 315k data, 212k init, 0k highmem)
Rilevazione della memoria fisica disponibile.

CPU: Intel Pentium II (Deschutes) stepping 01
Identificazione del processore

Serial driver version 5.05c (2001-07-08) with MANY_PORTS SHARE_IRQ SERIAL_PCI enabled
ttyS00 at 0x03f8 (irq = 4) is a 16550A
ttyS01 at 0x02f8 (irq = 3) is a 16550A
Inizializzazione delle porte seriali

hda: IBM-DTTA-351010, ATA DISK drive
Identificazione dell'hard-disk

Partition check:
hda: hda1 hda2 < hda5 hda6 hda7 hda8 >
Verifica dell'integrità delle partizioni da montare.

INIT

Init è il padre di tutti i processi, il suo ruolo consiste nel gestire il lancio di tutti i programmi necessari per rendere il sistema attivo creando i processi a partire da uno script nel file ***/etc/inittab***.

Nell'inittab vengono definiti:
- le directory con gli script di avvio per i diversi runlevel (/etc/rc.d/rc.0);
- il runlevel di default;
- altri script e comandi che vengono eseguiti al boot o in condizioni particolari.

Il primo script lanciato da inittab è (su Fedora 7) /etc/rc.d/rc.sysinit.

Tra le varie operazioni di init viene eseguito il controllo sulla consistenza dei file system, se si trovano errori che non possono essere riparati automaticamente è possibile che il processo di boot si blocchi.

In questo caso viene richiesta la password di root e viene eseguita una shell dalla quale si può eseguire un file system check manuale.
Dopo aver verificato su quale partizione il kernel si è fermato, si può dare il comando: **fsck.ext2**

Es: fsck.ext2 /dev/hda5

/etc/inittab

Il primo script lanciato da inittab è (su Fedora 7) /etc/rc.d/rc.sysinit, che esegue varie operazioni tra cui:

- Impostazioni di alcuni path generali nella variabile $PATH;
- Configurazione dell'ambiente di rete;
- Avvio della partizione di swap per la memoria virtuale;
- Impostazione del nome dell'host;
- Check del file system root;
- Check delle quote di spazio assegnate agli utenti, se previste;
- Mount del file system root in modalità scrittura/lettura;
- Preparazione del sistema per caricamento dei moduli;
- Check delle dipendenze dei moduli;
- Check di tutti i file system ed eventuali riparazioni;
- Mount di tutti i file system;
- Pulizia di file di supporto al boot e di processi non più attivi;
- Umount dell'initrd;
- Impostazione dell'orologio;
- Inizializzazione delle porte seriali;
- Caricamento Moduli;
- Attivazione dei servizi del runlevel.

Runlevel

Init viene configurato in modo da potere portare il sistema in diversi livelli operativi ognuno dei quali presenta caratteristiche diverse (programmi e servizi in esecuzione), detti ***runlevel***.

In genere su Linux sono utilizzati i seguenti livelli:

Runlevel 0 :	/etc/rc.d/rc0.d	Avvia la sequenza di arresto del sistema (shutdown)
Runlevel 1:	/etc/rc.d/rc1.d	Modalità singolo utente, nessun altro utente può collegarsi, il servizio di rete è disabilitato.
Runlevel 2:	/etc/rc.d/rc2.d	Multiutente, il servizio rete è attivo ma è disabilitato il file sharing.
Runlevel 3:	/etc/rc.d/rc3.d	Multiutente, predefinito quando si opera in modalità testuale, tutti i servizi sono attivi.
Runlevel 4:	/etc/rc.d/rc4.d	Inutilizzato. Può essere dedicato ad usi personali
Runlevel 5:	/etc/rc.d/rc5.d	E' il runlevel predefinito quando si vuole avviare Linux in modalità grafica (viene eseguito il server X)
Runlevel 6:	/etc/rc.d/rc6.d	Il runlevel 6 è quello di reboot.

Lo script /etc/rc.d/rc gestisce quali processi far partire a seconda del runlevel, andando ad analizzare le singole directory /etc/rc.d/rc#.d.

Runlevel

Gli script per la gestione dei processi di tutti i runlevel si trovano nella directory ***/etc/rc.d/init.d***.

Nelle directory dei vari runlevel, invece, esistono una serie di symlink con nomi del tipo S12syslog o K65identd che puntano agli script contenuti in /etc/rc.d/init.d.

I nomi dei link presentano la seguente nomenclatura:

<azione><ordine><nome>

azione — può essere **S** (start) o **K** (kill).
S fa partire il servizio associato
K ferma il servizio corrispondente.

ordine — indica in l'ordine di esecuzione degli script (dal più basso al più alto).

nome — è lo stesso del file presente sotto init.d

Lo script /etc/rc.d/rc è lo script “padre”, quello che a secondo del runlevel richiesto si occupa di eseguire gli script contenuti nella directory corrispondente.

Runlevel

Esempi:

Se abbiamo il file /etc/rc.d/rc3.d/S85httpd (link a /etc/rc.d/init.d/httpd), si avrà un server web avviato quando la macchina si troverà al runlevel 3.

Se vogliamo, invece, che venga avviato il server web in questo runlevel, basterà rinominare il file, sostituendo la lettera K alla S:

mv /etc/rc.d/rc3.d/S85httpd /etc/rc.d/rc3.d/K85httpd

Gli script possono anche essere utilizzati direttamente dall'utente per gestire i singoli servizi:

/etc/rc.d/init.d/httpd [start | stop | restart] fa partire, ferma, riavvia il server Web

Sotto Fedora i servizi possono essere attivati anche con il comando:

service nome_servizio (start | stop | restart)

Gestione dei servizi

La configurazione dei servizi per i vari runlevel mostrata finora permette una grande versatilità, ma non è facilmente gestibile.

Per aiutare l'utente nella configurazione dei servizi disponibili al boot si possono usare dei tool testuali o grafici.

I più comuni sono:

serviceconf	permette di configurare i servizi per i singoli runlevel (grafico)
ntsysv	per configurare i servizi del runlevel corrente (testuale con interfaccia semigrafica)
chkconfig	per tutti i runlevel. (testuale a riga di comando)

E' consigliabile disattivare tutti i servizi che non si utilizzano per due OTTIMI motivi:

- minore consumo di risorse: soprattutto CPU e RAM
- sicurezza: un servizio attivo è passibile di attacchi da parte di hacker

Installazione dei programmi

Inizialmente in Unix l'installazione dei programmi doveva essere eseguita manualmente e richiedeva la compilazione dei sorgenti in genere scritti in C/C++.

Ancora oggi questo metodo è utilizzato anche se è stato affiancato da soluzioni più sofisticate e più facili da utilizzare.

Per installare un programma partendo dal codice sorgente è necessario:

- scaricare i file sorgente normalmente raccolti in un file di tipo tar.gz
- decomprimere il file di archivio: tar xvzf programma.tar.gz
- posizionarsi nella cartella ottenuta: cd programma
- se presente il file configure digitare: ./configure
  verrà creato il file con le istruzioni per la compilazione, il Makefile
- digitare il comando: make
- digitare il comando (opzionale): make install

Per disinstallare un programma se presente si usa il comando: make unistall
Diversamente vanno cancellati a mano tutti i file installati.

La personalizzazione del kernel di Linux deve essere eseguita in questo modo.
Nella cartella che contiene i sorgenti del kernel digitare:

make menuconfig	per selezionare le opzioni necessarie
make bZimage	per creare l'immagine compressa del kernel

Vantaggi nell'uso dei sorgenti

Lo svantaggio dovuto ad una maggiore complessità e necessità di una conoscenza più approfondita delle fasi di compilazione, viene ampiamente ripagato dai vantaggi forniti dall'uso dei sorgenti.

Poter installare software e compilare lo stesso kernel tramite i sorgenti ha dei grandi vantaggi che l'installazione diretta di binari non offre.

- Aumento delle performance tramite ottimizzazioni e razionalizzazioni,
- Possibilità di customizzazione,
- Possibilità di correzione di bugs tramite patch
- Possibilità di eseguire aggiornamenti in modo rapido, appena sul sito del produttore appaiono i sorgenti di una nuova versione.

In particolare per il kernel la necessità di compilazione è quasi d'obbligo su sistemi in produzione per i quali non si vogliano mantenere kernel generici o con il supporto di innumerevoli moduli come quelli comunemente forniti nelle distribuzioni standard.
Si ricordi che un kernel monolitico, ad esempio, ha una maggiore velocità di esecuzione, una minore occupazione di memoria ed una maggiore stabilità.

RPM

Red Hat ha creato diversi anni fa una utility, **RPM** (Red Hat Package Manager), che facilità tutte le operazioni necessarie alla gestione dei programmi.

RPM, serve per installare, disinstallare, aggiornare, interrogare, verificare e costruire pacchetti software (packages).

RPM gestisce le dipendenze solo di primo livello (non è recursivo).
Se per l'installazione di un package come prerequisito è necessario un altro package, RPM indica il package mancante, ma non se a sua volta “dipende” da altri.

Analogamente se si prova a disinstallare un package da cui ne dipendono altri si viene avvisati di questa evenienza e l'operazione viene interrotta.

Gli RPM automaticamente copiano i file di un pacchetto nelle directory giuste (logs in /var/log, file di configurazione in /etc/, binari in /usr/bin o /usr/sbin, script di startup in /etc/rc.d/init.d/ ecc.) e verificano la presenza di conflitti o installazioni più recenti.
Un RPM non cancella mai nulla che non abbia installato.
Se deve sostituire o cancellare un file di configurazione, per esempio, mantiene il file esistente aggiungendo il suffisso **.rpmsave**.

Uso di RPM

Comandi principali

rpm -i [opzioni] [pacchetti]	Installazione pacchetti RPM
rpm -U [opzioni] [pacchetti]	Aggiornamento di pacchetti RPM
rpm -e [opzioni] [pacchetti]	Disinstallazione di pacchetti RPM
rpm -q [opzioni] [pacchetti]	Interrogazione di pacchetti RPM
rpm -V [pacchetto]	Verifica del pacchetto

esempi:

$ rpm -qa | grep smb — Visualizza i pacchetti il cui nome contiene “smb”

```
libsmbios-libs-0.13.10-1.fc7
libsmbclient-3.0.26a-0.fc7
smb4k-0.8.4-1.fc7
```

$ rpm -ql smb4k | more — Elenca file contenuti in un pacchetto installato

$ rpm -qlp smb4k.rpm | more — Elenca file contenuti in un pacchetto rpm

rpm usa un database nel quale memorizza i pacchetti installati.

Fedora durante la fase di installazione crea il file /root/install.log che contiene l'elenco di tutti i pacchetti installati.
Per avere un elenco aggiornato di tutti i pacchetti installati sul sistema, si può impartire il comando: rpm -qa | sort > /root/install.log

RPM

Convenzione dei nomi dei package

nome-versione-release.arch.rpm

esempio:

efax-0.8a-11.i386.rpm	ottimizzato per cpu Intel 386
efax-0.8a-11.i686.rpm	ottimizzato per cpu Intel P4 e successive
efax-0.8a-11.x86_64.rpm	ottimizzato per S.O. a 64 bit e cpu Intel a 64 bit

Per la ricerca dei pacchetti precompilati si può usare il comando rpmfind o il sito:
http://rpmfind.net/linux/RPM/

Interfacce grafiche per RPM

Per un uso più comodo si possono gestire gli rpm usando delle interfacce grafiche.
Le più comuni sono: gnorpm, kpackage, xrp, rpmdrake, pkgview

Svantaggi

- I file sono già compilati, se si intende customizzare alcuni flag di compilazione si deve sempre farlo manualmente partendo da un tar.gz;
- Non sempre nuovi RPM vengono resi disponibili in tempi brevi quando esce una nuova versione di un programma.
- La gestione delle dipendenze non prevede la recursione.

Ulteriori informazioni sono disponibili su: http://www.rpm.org

YUM Yellow dog Updater, Modified

Yum (http://linux.duke.edu/projects/yum/) è un gestore di package automatico per sistemi che usano il formato rpm.
Yum trova le dipendenze e mostra quali altri package devono essere installati oltre quello indicato per una corretta installazione.

Rispetto RPM presenta le seguenti caratteristiche:
- Uso di Repositori multipli
- File di configurazione semplice
- Calcola correttamente le dipendenze (anche recursivamente)
- Uso del formato rpm
- Interfaccia utente semplificata.

L'uso è molto semplice:

```
yum [install | update | remove] package_1 [package_2 .... package_n]
```

Per configurare yum si usa il file /etc/yum.conf

Per la ricerca dei package yum si appoggia a dei Repository

Anche per yum esistono dei frontend grafici, fedora usa **pirut**
Si può installare un altro frontend che presenta delle funzioni avanzate: **yumex**

```
$ yum install yumex
```


YUM Repository

Fedora si appoggia ai suoi Repository ufficiali, ma è conveniente aggiungerne altri al fine di poter scegliere tra un più ampio ventaglio di software già precompilato per la propria distribuzione.
Per aggiungere dei repository è sufficiente creare un nuovo file con estensione .repo all'interno della cartella /etc/yum/repos.d

```
$ cat /etc/yum.repos.d/livna.repo
[livna]
name=Livna for Fedora Core $releasever - $basearch - Base
baseurl= http://rpm.livna.org/fedora/$releasever/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-livna

$ cat /etc/yum.repos.d/dries.repo
[dries]
name=Extra Fedora rpms dries - $releasever - $basearch
baseurl=http://ftp.belnet.be/packages/dries.ulyssis.org/fedora/fc$releasever/
    $basearch/RPMS.dries/
gpgkey=http://dries.ulyssis.org/rpm/RPM-GPG-KEY.dries.txt
gpgcheck=1
enabled=1
```

Altri package manager

Quanto detto finora per la gestione dei pacchetti rpm, vale non solo per Fedora, ma per tutte le distribuzioni che utilizzano l'rpm come package manger.

Altri package manager evoluti tipo yum sono:

YaSt (**Y**et **A**nother **S**etup **T**ool) utilizzato da SUSE

urpmi utilizzato da Mandriva

DPKG

Le distribuzioni debian e quelle su essa basate usano **dpkg** (al posto di rpm) e **apt** (equivalente a yum)

Le funzionalità sono simili a quelle di rmp ed yum, cambia la sintassi dei comandi.
Per ulteriori approfondimenti si rimanda alla pagina del manuale:

man dpkg

man apt

e al link: http://it.wikipedia.org/wiki/Advanced_Packaging_Tool

Anche in questo caso esistono dei frontend grafici che ne facilitano l'utilizzo:

apitude, synaptic, dselect e gnome-apt

Corso Base su Linux

Fine Lezione 2