Linux Base - Capitolo n. 4

Edizioni ByteMan (08-11-2005)

revisione: 25/01/2008

Riepilogo

Come era logico aspettarsi, dato il lungo intervallo intercorso tra la fine del primo modulo e l'inizio del secondo, si è reso necessario un minimo di riepilogo di quanto è stato fatto in precedenza.

Su suggerimento di alcuni colleghi abbiamo inserito in ciascun seminario una sezione **Download** contenente la versione stampabile compressa (.zip) del materiale sviluppato nel corso degli incontri in modo da poter disporre anche di un efficace supporto cartaceo. Per motivi di praticità è stato scelto il formato **.doc** leggibile sia con MS-Office sia con OpenOffice.

Questi i punti salienti toccati nel corso dell'incontro:

- Data la grande quantità di distribuzioni Linux, che all'inizio confondono il neofita, abbiamo voluto semplificare individuando 3 grandi filoni principali: **Debian**, **RedHat**, **Slackware**. Ciascuna di queste ha poi tutta una serie di distribuzioni derivate. Noi abbiamo scelto, per questo corso, Debian ed, in particolare, una sua derivata Live: **Knoppix 3.8.1** (vedi maggiori dettagli nel seminario 3).
- Abbiamo analizzato con maggiore cura la struttura gerarchica del file system di Linux conforme alle specifiche FHS (vedi seminari 1 e 4). Dalla sezione **Download** è possibile scaricare i lucidi proiettati.
- Ci siamo soffermati su alcuni comandi utili per la gestione di file e directory. Maggiori dettagli nella sezione Comandi Base (parte 3) in questo stesso seminario.
- E' stata infine ribadita l'importanza di sapere effettuare le varie operazioni di sistema in due modi: con l'**interfaccia grafica** e con la **linea di comando**.
  Non ci soffermeremo a lungo sulla parte grafica data la forte analogia con altri sistemi operativi: copia/incolla, clicca e trascina, uso del pulsante destro del mouse, etc. sono ormai abilità quasi innate di qualsiasi utente di computer.
  Con l'uso della linea di comando cercheremo di recuperare una abilità che molti credono di potere relegare nel passato remoto e che consente, invece, il pieno controllo del sistema, anche quando per svariati motivi non sia disponibile l'interfaccia grafica.

Il filesystem: gerarchia

Per file system si intende l'astrazione (metodo e protocolli) con cui un sistema operativo organizza i file su un supporto fisico di memorizzazione ad accesso casuale (floppy, cdrom, memoria, hard disk...). I sistemi operativi moderni tipicamente utilizzano un sistema gerarchico (diviso in directory e sottodirectory) e possono supportare nativamente uno o più diversi file system.
*Linux vede ogni cosa come se fosse un file; spesso addirittura come un file di testo che, generalmente, può essere aperto, letto e modificato, se lo si ritiene opportuno. Alcuni esempi sono i file contenuti in /dev (devices) e /proc (process). La directory **/dev** è la posizione dove risiedono le periferiche collegate al Pc (l'hardware), mentre la directory **/proc** rappresenta letteralmente il contenuto della memoria di sistema, dagli irq al bus Pci, fino ai file temporanei. Non è assolutamente un buona idea apportare modifiche in /dev o /proc ma osservarne il contenuto può aiutare a capire il concetto che tutto in linux è considerato un file, in un modo o nell'altro.*
*I sistemi unix-like (quindi anche linux) esistono fin dagli albori dell'informatica, e il loro file system, anche se nel corso del tempo ha avuto qualche cambiamento, è rimasto sostanzialmente lo stesso. Oggi esiste uno standard chiamato **FHS** a cui normalmente ci si attiene nella strutturazione del file system.*

- Unico albero delle cartelle, a partire dalla radice "/"
- Case sensitive
- Struttura più rigida rispetto ai sistemi Windows
- FHS (Filesystem Hierarchy Standard)
- Standard "dal basso": fa riferimento a pratiche consolidate nel mondo UNIX
- Non è rigido
- Le distribuzioni GNU/Linux hanno col tempo recepito le sue indicazioni

Anche se **il file system di linux** può sembrare complesso all'inizio, esso **è flessibile e logico** una volta appreso il suo funzionamento, **e la sua conoscenza è essenziale** in moltissime circostanze d'uso.

Cartella radice "/"

Il concetto più importante da cogliere al volo è che questa directory contiene tutto: dagli elementi fondamentali del sistema ai punti di innesto (mount) per i dispositivi ed altri eventuali file system.

- /bin
- /opt
- /boot
- /proc (Linux)
- /dev
- /root
- /etc
- /sbin
- /home
- /tmp
- /lib
- /usr
- /mnt
- /var

Cartella: "/bin"

*Contiene tutti i file fondamentali (**bin**aries) per l'uso del sistema, sia da parte dell'utente root sia da parte degli utenti ordinari. Ad esempio, non possono mancare in questa directory i comandi per copiare e rimuovere file, per spostarsi fra le directory ma anche alcuni applicativi di rete.*

- Comandi fondamentali per il funzionamento basilare del sistema.
- Utilizzabili sia dall'amministratore sia dagli utenti.
- sh, ls, mount, ln,chmod, ps, cat, kill, tar, ping, ...
- Non contiene sottocartelle

Cartella: "/boot"

Contiene di norma i file essenziali per l'avvio del sistema: solitamente, al suo interno viene conservata anche l'immagine del kernel, necessaria per il boot del sistema. Da qui il sistema si avvia, viene allocata la memoria, riconosciuto l'hardware e montati i filesystem, e tutte le varie altre cose che devono essere effettuate all'avvio. Non è opportuno, se non si è assolutamente sicuri di ciò che si sta facendo, modificare questi file.

- Contiene i dati necessari per la fase di avvio del sistema (ad es. copie del master boot record).
- Il kernel va posizionato in questa cartella o direttamente nella cartella radice.
- Non contiene i file di configurazione (che vanno in **/etc**), né i programmi necessari per modificare il boot loader (da posizionare in **/sbin**)

Cartella: "/dev"

*Contiene tutta la lista dell'hardware del pc, con alcune ripetizioni necessarie per server e configurazioni hardware speciali, tutte le periferiche a cui si ha accesso sono localizzate in questa posizione. Sono, in effetti, contenuti dei file che identificano i **dev**ice presenti nella macchina.*
Come si ricorderà, abbiamo identificato la prima partizione del primo disco IDE come /dev/hda1: esplorando questa directory, ovviamente, lo ritroveremo assieme a molti altri.

- Contiene i file dei dispositivi
- I device driver mettono a disposizione un'interfaccia standard, che si presenta a tutti gli effetti come un file.
- Esempi di dispositivi:
  hda, hdb, hdc, hdd, sda, sdb,....
  hda1, hda2,....
  ttyS0, ttyS1,....
  video0, video1,....
  random, rtc,

Cartella: "/etc"

E' una directory "contenitore", visto che al suo interno trovano spazio i file di configurazione dei programmi installati nel sistema. Per chiarezza, solitamente, ogni programma crea una propria sottodirectory in /etc nella quale inserisce i propri file di configurazione (a meno che il file di configurazione sia solo uno). Per la configurazione del sistema, quindi, /etc è il primo posto dove andare a guardare.
Contiene, ad esempio, i binari del sistema a finestre X (in /etc/X11), gli script di avvio (/etc/rc.d), la lista dei filesystem e dei punti di mount (fstab e mtab) e vari altri file.

- Contiene i file che definiscono la configurazione (globale) della macchina e dei relativi programmi.

- Alcuni file sicuramente presenti:
  fstab, group, passwd, profile, syslog.conf, ld.so.conf...
  host.conf, hosts, services, ...
  script di avvio della macchina

- **/etc/X11**: file di configurazione relativi al sistema grafico

Cartelle: "/home" e "/root"

/home
E' la "casa" degli utenti ordinari del sistema: al suo interno trovano infatti spazio diverse directory, una per ogni utente e con il nome dell'utente stesso (ad esempio, /home/user1 e /home/user2). Ogni utente, dopo il login, verrà mandato proprio nella sua home directory dove potrà lavorare in libertà, visto che nel resto del sistema difficilmente potrà scrivere dei file.
*Ogni utente è separato dagli altri grazie ai permessi utente. Solo **root** può scavalcare questi permessi in ogni momento ed è in grado di leggere e modificare ogni file del sistema.*

- Normalmente contiene le cartelle personali degli utenti (nella forma **/home/nomeutente**), ma non è obbligatorio.

- Nei sistemi Windows, è stata (solo di recente) assegnata una destinazione d'uso simile per "Documents and Settings".

/root
E' la casa dove l'amministratore (o "superuser" o appunto "root") tiene i suoi file, per un normale utente è impossibile accedere in questa directory, alcuni amministratori rinominano la directory /root e la spostano in qualche altra parte del sistema, per prevenire curiosi ed eventuali hackers che riescano ad accedere al sistema.

- Unica eccezione: la cartella personale dell'amministratore, che normalmente va posizionata in **/root**, per motivi di sicurezza ed affidabilità del sistema.

Cartella: "/lib"

*Contiene soltanto le **lib**rerie dinamiche richieste dalle applicazioni, in modo analogo alle DLL (Dynamic Linking Library) di Windows.*

- Contiene (tutte e sole) le librerie dinamiche (file so) necessarie per l'avvio del sistema e l'esecuzione dei programmi in **/bin** e **/sbin**.

- Sicuramente, libc.so e libm.so, ed il linker dinamico ld.so

- Cartella **modules** contiene i moduli del kernel caricabili a run-time.

Cartelle: "/mnt" e "/opt"

/mnt
Questa è generalmente la directory dove vengono montati (innestati) altri file system, questa non è una regola, ma è più semplice avere il cdrom in /mnt/cdrom e il floppy disk in /mnt/floppy piuttosto che sparsi in altre directory del disco fisso.

- Utilizzata per montare i filesystem temporanei.

- Ad esempio, tutti quelli legati a dispositivi removibili (floppy, cdrom, dispositivi di memorizzazione usb).

/opt
*Sta per **opt**ional. Destinata a contenere software non presente nella distribuzione.*

- Riservata per i pacchetti software aggiuntivi (ad esempio, pacchetti solo binari).

Cartella: "/proc"

*Il contenuto di questa cartella (abbreviazione di **proc**ess) in realtà non esiste, si tratta del contenuto della memoria di sistema, essa non occupa dello spazio sul disco ed il suo contenuto viene ricreato ad ogni riavvio, questa directory è utile per capire cosa linux sta facendo durante il suo funzionamento. Può essere utile per tenere un log dello stato del sistema o come punto di riferimento per informazioni tecniche da utilizzare per l'assistenza.*

- Pseudo filesystem, specifico di Linux
- Accesso a varie informazione del sistema (cpu, moduli, processi, sottosistema di rete).
- È possibile agire su alcuni di questi "file" per modificare dinamicamente lo stato del kernel.
- Con l'introduzione della nuova serie di kernel 2.6,
  varie informazioni relative al sistema ed alla sua rappresentazione interna (dispositivi, bus,...)sono state accessibili tramite il filesystem /sys.
- Vedere ad es.:

`cat /proc/cpuinfo`	*informazioni sul processore*
`cat /proc/version`	*informazioni sulla versione del sistema operativo*
`cat /proc/interrupts`	*mostra quali interrupt vengono usati dal sistema*
`cat /proc/pci`	*informazioni dettagliate sullo stato del bus pci*
`cat /proc/meminfo`	*informazioni contenute nella ram (o "core")*

Cartella: "/sbin"

*Sta per **s**uperuser **bin**aries. Contiene eseguibili di sistema, un insieme di file estremamente importanti come fsck (che controlla lo stato del filesystem), reboot, init, swapon, mount, insmod (che controlla le dipendenze dei moduli e le cambia dinamicamente al riavvio), ifconfig, route.*
Non dovrebbe essere messa in una partizione separata perchè in caso di corruzione del file /etc/fstab o /etc/mtab diventerebbe impossibile usare fsck per riparare il filesystem, tenerla insieme a / (root) è un'ottima soluzione.

- Analogo a /bin: contiene comandi utili per l'amministrazione del sistema (e quindi non utilizzati dagli utenti), necessari all'avvio dello stesso.
- Divisione per motivi logici, non tanto per motivi di sicurezza.
- Alcuni programmi contenuti:
  init, swapon, halt, shutdown
  fdisk, fsck., mkfs.**
  ifconfig, route

Cartella: "/tmp"

Essa generalmente contiene files che vengono usati temporaneamente da un'applicazione, è anche un buon posto per "buttare" qualcosa, dato che il contenuto di questa cartella viente generalmente cancellato ad ogni reboot.

- Utilizzato per contenere i file temporanei eventualmente richiesti dai programmi
- I programmi non possono fare affidamento sul contenuto della cartella: una volta terminato l'uso di un file temporaneo, esso può essere cancellato in qualsiasi momento.
- Alcune implementazioni prevedono la cancellazione del contenuto della cartella /tmp all'avvio del sistema.

Cartella: "/usr"

*Questa directory contiene tutti gli eseguibili degli utenti (più alcuni eseguibili che dovrebbero essere usati solo dall' amministratore, in **/usr/sbin**). Sono conservati qui anche alcuni file di help e c'è anche della documentazione in **/usr/doc** che può tornare utile.*

- Ha una struttura complessa, come la cartella "/". In un certo senso, ne rappresenta un'immagine.
- Contiene informazioni non scrivibili dagli utenti.
- Nessun programma può creare cartelle direttamente in questa cartella.
- Alcune cartelle contenute in /usr:
  bin, sbin, lib:
  corrispondono alle rispettive cartelle del filesystem radice, e al contrario di queste contengono programmi non vitali per il funzionamento del sistema.
  games:
  giochi e programmi educativi
  include:
  i file include utilizzabili dai programmi C/CPP
  local:
  utilizzato per creare un'ulteriore livello di gerarchia. Ad esempio, possono essere posizionati qui i programmi compilati sul sistema, senza utilizzare un sistema di gestione dei programmi (rpm, dpkg).
  share:
  dati vari non dipendenti dall'architettura; a sua volta contiene le cartelle: *doc* con la documentazione; *man* per le pagine di manuale; *pixmaps e icons, ecc.*
  X11R6:
  contiene un'ulteriore immagine della cartella radice (bin, include, lib, man)
  src:
  sorgenti dei programmi. Normalmente, vanno posizionati qui i sorgenti del kernel.
- Esempio di uso: un ipotetico editor di testo di nome "textedit" posiziona:
  i file di programma in /usr/bin;
  le librerie condivise in /usr/lib o /usr/lib/textedit
  le pagine di manuale in /usr/share/man/man1
  la documentazione in /usr/share/doc/textedit
  eventuali include per sviluppare plugin in /usr/include

Cartella: "/var"

Contiene tutte le informazioni variabili, ad esempio le cartelle di spool, i file di log, informazioni sui programmi in esecuzione, gli archivi dei database server, ecc. In sostanza /var contiene tutti quei file sottoposti a cambiamento.

- Alcune cartella contenute:
  cache:
  usata come per la cache delle applicazioni; che possono eventualmente creare sottocartelle per i propri dati.
  games:
  i dati variabili dei giochi (punteggi globali).
  lib:
  informazioni relative ai programmi; anche in questo caso, possono essere create sottocartelle. Ad esempio, gli archivi dei motori di database postgresql e mysql si trovano rispettivamente in **/var/lib/postgres** e **/var/lib/mysql**.
  lock:
  file di lock
  log:
  i file di log (messages, wtmp, lastlog, syslog...)
  mail:
  le mailbox degli utenti (se presente un server di posta)
  run:
  file che descrivono lo stato corrente del sistema. Ad esempio, per ogni processo in esecuzione, in questa cartella si trova un file (.pid) che contiene l'identificativo del processo.
  spool:

dati che aspettano in coda per essere successivamente processati (ad esempio spool/lpd o spool/cups contengono la coda di stampa, e spool/cron i dati relativi al demone cron)

tmp:

file temporanei non eliminati all'avvio del sistema

Comandi Base (parte 3)

Passeremo in rassegna alcuni comandi base per la gestione dei file: copiare, spostare ed eliminare file. Inoltre, vedremo come eseguire le stesse operazioni sulle directory, imparando anche come crearle. Copia (**cp**). Sono necessari due argomenti: il file sorgente, che è il file esistente da copiare, ed il file destinazione, ossia il nome della nuova copia. **cp** crea quindi una copia identica del file sorgente, attribuendogli il nome di destinazione specificato. Se esiste già un file con quel nome di destinazione, **cp** lo sovrascrive senza alterare il file sorgente.

```
cp miofile filecopia
```

copia il file *miofile* in *filecopia*, oppure, per la copia multipla di file, si usa:

```
cp file1 file2 file3 ... fileN destinazione
```

che permetterà di copiare i file file1 .. fileN nella directory di destinazione. Si noti che in questo caso l'ultimo argomento del comando deve essere una directory. Vediamo un esempio:

```
cp miof1 miof2 /tmp
```

che copierà *miof1* e *miof2* nella directory */tmp*. Nel caso in /tmp esistessero già file con lo stesso nome, questi verrebbero sovrascritti dai nuovi file.

Vediamo ora le opzioni più importanti di cp:

- **-b** (backup) Esegue automaticamente una copia di backup di ogni file di destinazione esistente.
- **-f** (force) Forza la sovrascrittura dei file, senza richiedere interazioni con l'utente (cautela!).
- **-i** (interactive) Attiva la modalità interattiva, che chiede conferma prima dell'evanutale sovrascrittura di file di destinazione preesistenti; il suffisso usato è il classico simbolo della tilde, ~.
- **-p** (permission) Mantiene, se possibile, gli attributi (permessi, ownership ecc.) del file.
- **-r** (recurse) Permette di attivare la modalità ricorsiva, che permette la copia di directory.
- **-v** (verbose) Attiva la modalità verbose, che visualizza in output quello che il sistema ha fatto in seguito al nostro comando.

Dopo aver visto come copiare file e directory, dedichiamoci allo spostamento ed al cambio di nome dei file; tutto questo è fatto con un solo comando: **mv**, la cui sintassi è:

```
mv SORGENTE DESTINAZIONE                 per rinominare
mv SORGENTE DIRECTORY                    per spostare un file
mv SORG1 SORG2 ... SORGn DESTINAZIONE    per spostare più file
```

Tramite mv, oltre a cambiare il nome di un file, è anche possibile rinominare una directory o muovere più directory in un'altra directory. Le opzioni di mv sono molto simili a quelle di cp, sebbene in numero minore. Quelle che a noi interessano sono le stesse viste per il comando cp, con l'esclusione di -p che per mv non esiste.

Come ultimo, ci rimane il comando rm, che permette di eliminare file o directory. La sua sintassi è abbastanza semplice:

```
rm FILE (per eliminare un file)
rm FILE1 .. FILE-N (per eliminare più file)
```

Le opzioni di rm sono anch'esse simili a quelle di mv; le più utili sono le stesse che sono state viste per cp, con l'esclusione di "-p" che qui non ha alcun senso pratico. Deve qui essere aperta una parentesi relativa alle directory: di norma, se siete sicuri di poter elimimare una directory non vuota, potete utilizzare il comando

```
rm -rf directory
```

che forza l'eliminazione della directory 'directory' e dei suoi contenuti. Per eliminare una directory vuota, si usa invece il comando:

```
rmdir directory
```

A volte, però, tale comando non sembra sortire gli effetti desiderati: nonostante la directory sembri vuota, quanto scritto sopra ci restituisce un errore, dove il sistema si lamenta perchè esistono dei file all'interno della directory che vogliamo eliminare. Accade spesso, infatti, che per controllare il contenuto di una directory si usi solamente "ls" che, per sua natura, non visualizza i file nascosti! A questo punto si può procedere in due modi: entrare nella directory e cancellare tali file oppure, se si è sicuri che i file nascosti all'interno di tale directory non sono importanti, si può procedere con rm -rf. Come ultima cosa, vediamo come creare le directory: il comando è mkdir, la cu sintassi è:

```
mkdir DIRECTORY (per creare una directory)
mkdir DIRECTORY1 .. DIRECTORY-N (per creare più directory).
```

Vediamo un esempio: nella nostra home directory, supponiamo /home/utente, è presente la directory "dir" e vogliamo creare "sottodir" come sottodirectory della precedente. La prima cosa che viene in mente di fare è di entrare in "dir" e lanciare il comando:

```
mkdir sottodir
```

È però possibile, e anche molto più comodo, utilizzare direttamente il comando:

```
mkdir dir/sottodir
```

con una sola attenzione: il comando

```
mkdir dir/sottodir sottodir2 sottodir3
```

non fa quello che ci si potebbe aspettare, ossia creare sottodir, sottodir2 e sottodir3 come sottodirectory di "dir"; quello che qui accade, invece, è la creazione di "sottodir" come sottodirectory di "dir" e la creazione di "sottodir2" e "sottodir3" allo stesso livello di "dir"! Quindi, per non dover scrivere:

```
mkdir dir/sottodir dir/sottodir2 dir/sottodir3
```

in questo caso è più comodo entrare in dir e lanciare:

```
mkdir sottodir sottodir2 sottodir3
```

Con queste nozioni, potrete certamente iniziare a fare gli esperimenti sulla creazione, rimozione e spostamento di file e directory. Permetteteci però qualche consiglio:

- utilizzate touch per creare file vuoti con cui operare; il comando "touch file" crea il file vuoto "file"; utilizzate invece il comando "mkdir" (mkdir nomedirectory) per la creazione delle directory;
- spostatevi in /tmp per le prove; è più sicuro e vi eviterà di lasciare file e directory inutili sparsi per il filesystem o, peggio, di eliminare file importanti o di sistema per qualche disattenzione;
- non, ripeto, NON utilizzate l'utente root per queste prove: uno stupido errore di digitazione può, come si dice in gergo, "rasare" un intero filesystem!

In conslusione, ricordiamo che tutti i comandi fino a qui visti permettono di specificare caratteri jolly, quali ad esempio l'asterisco o il punto di domanda. Ad esempio:

```
rm ab*
```

rimuoverà tutti i file il cui nome inizia con "ab" seguiti da zero o più caratteri. Invece, il comando:

```
rm ab?
```

eliminerà tutti i file il cui nome inizia con ab seguiti da un qualsiasi carattere.